

MQSeries<sup>®</sup> for OS/390<sup>®</sup>



# System Administration Guide

*Version 5 Release 2*

**Note!**

Before using this information and the product it supports, be sure to read the general information under “Appendix B. Notices” on page 245.

**First edition (November 2000)**

This edition applies to MQSeries for OS/390 Version 5 Release 2, and to all subsequent releases and modifications until otherwise indicated in new editions.

This book is based on parts of the *MQSeries for OS/390 System Management Guide*, SC34-5374-01.

© **Copyright International Business Machines Corporation 1993, 2000. All rights reserved.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

**Figures . . . . . vii**

**Tables . . . . . ix**

**About this book . . . . . xi**

Who this book is for . . . . . xi

What you need to know to understand this book . . . . . xi

Conventions used in this book . . . . . xii

| What's new for this release . . . . . xii

| Queue-sharing groups. . . . . xii

| Channel initiator . . . . . xii

| Commands . . . . . xiii

| System parameters. . . . . xiii

| System object samples . . . . . xiv

| Logs . . . . . xiv

| Security . . . . . xiv

| Statistics and accounting. . . . . xiv

| Operations and control panels . . . . . xv

| Dead-letter queue . . . . . xv

| Application programming . . . . . xv

**Part 1. Operating MQSeries for OS/390 . . . . . 1**

**Chapter 1. Operating MQSeries for OS/390 . . . . . 3**

Issuing commands . . . . . 3

Issuing commands from an OS/390 console or its equivalent . . . . . 3

Issuing commands from the utility program CSQUTIL . . . . . 4

Introducing the operations and control panels . . . . . 5

Invoking the operations and control panels . . . . . 5

Rules for the operations and control panels . . . . . 6

Objects and actions . . . . . 7

| Object dispositions . . . . . 10

| Choosing a queue manager . . . . . 10

Using the function keys . . . . . 11

Using the command line . . . . . 12

Using the operations and control panels. . . . . 13

| Defining objects . . . . . 14

Defining a local queue. . . . . 15

When your local queue definition is complete. . . . . 18

| Defining other types of objects . . . . . 18

Working with object definitions. . . . . 19

Altering an object definition. . . . . 19

Displaying an object definition . . . . . 19

Deleting an object . . . . . 19

Working with namelists . . . . . 20

**Chapter 2. Starting and stopping MQSeries . . . . . 21**

Before you start MQSeries . . . . . 21

Starting MQSeries . . . . . 22

Start options . . . . . 22

Starting after an abnormal termination . . . . . 23

User messages on start-up . . . . . 23

Stopping MQSeries . . . . . 24

Stop messages . . . . . 25

**Chapter 3. Writing programs to administer MQSeries. . . . . 27**

Understanding how it all works . . . . . 27

Before you begin . . . . . 27

Preparing queues for administration programs . . . . . 28

Defining the system-command input queue . . . . . 28

Defining a reply-to queue . . . . . 28

Opening the system-command input queue . . . . . 29

Opening a reply-to queue . . . . . 29

Using the command server . . . . . 30

Identifying the queue manager that processes your commands . . . . . 30

Starting the command server . . . . . 30

Sending commands to the command server . . . . . 30

Putting messages on the system-command input queue . . . . . 32

Retrieving replies to your commands. . . . . 33

Waiting for a reply . . . . . 33

The reply message descriptor . . . . . 34

Interpreting the replies . . . . . 35

Using the DISPLAY commands. . . . . 36

Examples of commands and their replies . . . . . 37

Messages from a DEFINE command . . . . . 37

Messages from a DELETE command . . . . . 37

Messages from DISPLAY commands . . . . . 37

Messages from commands with CMDSCOPE . . . . . 40

Messages from commands that generate commands with CMDSCOPE . . . . . 42

If you do not receive a reply. . . . . 43

Passing commands using MGCR or MGCRE . . . . . 43

**Part 2. MQSeries and CICS . . . . . 45**

**Chapter 4. Operating the CICS adapter 47**

Invoking the adapter's control functions. . . . . 47

From the CICS adapter control panels . . . . . 47

From the CICS command line . . . . . 47

From CICS application programs . . . . . 48

Accessing the CICS adapter control panels . . . . . 50

Starting a connection . . . . . 51

Starting a connection from the CICS adapter control panels . . . . . 51

Starting a connection from the CICS command line . . . . . 52

Starting a connection from a CICS application program . . . . . 53

Stopping a connection. . . . . 54

Stopping a connection from the CICS adapter control panels . . . . .	54
Stopping a connection from the CICS command line . . . . .	55
Stopping a connection from a CICS application program . . . . .	55
Modifying a connection . . . . .	56
Modifying a connection from the CICS adapter control panels . . . . .	56
Modifying a connection from the CICS command line . . . . .	57
Modifying a connection from a CICS application program . . . . .	58
Displaying details of connections and CICS tasks . . . . .	59
Displaying details of a connection from the CICS adapter control panels . . . . .	59
Starting an instance of the task initiator CKTI . . . . .	60
Starting CKTI from the CICS adapter control panels . . . . .	60
Starting CKTI from the CICS command line . . . . .	61
Starting CKTI from a CICS application program . . . . .	61
Starting CKTI automatically . . . . .	61
Stopping an instance of CKTI . . . . .	62
Stopping an instance of CKTI from the CICS adapter control panels . . . . .	62
Stopping an instance of CKTI from the command line . . . . .	63
Stopping an instance of CKTI from an application program . . . . .	63
Displaying the current instances of CKTI . . . . .	64
Displaying the current instances of CKTI from the CICS adapter control panels . . . . .	64
Displaying CICS task information . . . . .	65
Displaying CICS tasks from the CICS adapter control panels . . . . .	65
Displaying connection status and in-flight tasks . . . . .	66
Purging tasks that are using the CICS adapter . . . . .	67
Shutting down a connection between MQSeries and the CICS adapter . . . . .	68
Orderly shutdown . . . . .	68
Forced shutdown . . . . .	69

## Chapter 5. Operating the CICS bridge 71

Starting the CICS bridge . . . . .	71
Shutting down the CICS bridge . . . . .	72
Controlling CICS-bridge throughput . . . . .	72

## Part 3. MQSeries and IMS . . . . . 73

### Chapter 6. Operating the IMS adapter 75

Controlling IMS connections . . . . .	75
Connecting from the IMS control region . . . . .	76
Initializing the adapter and connecting to MQSeries . . . . .	76
Thread attachment . . . . .	77
Displaying in-doubt units of recovery . . . . .	78
Recovering in-doubt units of recovery . . . . .	78
Resolving residual recovery entries . . . . .	79
Controlling IMS dependent region connections . . . . .	80
Connecting from dependent regions . . . . .	80

Region error options . . . . .	80
Monitoring the activity on connections . . . . .	80
Disconnecting from dependent regions . . . . .	81
Disconnecting from IMS . . . . .	82
Controlling the IMS trigger monitor . . . . .	83
Starting CSQQTRMN . . . . .	83
Stopping CSQQTRMN . . . . .	83

### Chapter 7. Controlling the IMS bridge 85

Starting and stopping the IMS bridge . . . . .	85
Controlling IMS connections . . . . .	85
Controlling bridge queues . . . . .	86
Resynchronizing the IMS bridge . . . . .	87
Considerations for Commit mode 1 transactions . . . . .	87
Deleting messages from IMS . . . . .	88
Deleting Tpipes . . . . .	88

## Part 4. Managing MQSeries resources . . . . . 89

### Chapter 8. Managing the logs . . . . . 91

Archiving logs with the ARCHIVE LOG command . . . . .	91
Restarting the log archive process after a failure . . . . .	93
Optimizing archive log reading from tape devices . . . . .	93
Printing log records . . . . .	93
Recovering logs . . . . .	93
Discarding archive log data sets . . . . .	94
Automatic archive log data set deletion . . . . .	94
Manually deleting archive log data sets . . . . .	95

### Chapter 9. Managing the BSDS . . . . . 97

Finding out what the BSDS contains . . . . .	97
Time stamps in the BSDS . . . . .	97
Active log data set status . . . . .	98
Changing the BSDS . . . . .	99
Changes for active logs . . . . .	99
Changes for archive logs . . . . .	100
Recovering the BSDS . . . . .	102

### Chapter 10. Managing page sets . . . . . 105

How to add a page set to a queue manager . . . . .	105
What to do when one of your page sets becomes full . . . . .	106
How to balance loads on page sets . . . . .	107
Moving a non-shared queue . . . . .	107
How to expand a page set . . . . .	109
How to reduce a page set . . . . .	110
How to back up and recover page sets . . . . .	111
Creating a point of recovery . . . . .	111
Recovering page sets . . . . .	113
How to back up and restore queues using CSQUTIL . . . . .	115

### Chapter 11. Managing queue-sharing groups and shared queues . . . . . 117

Managing queue-sharing groups . . . . .	117
Adding a queue-sharing group to the DB2 tables . . . . .	117
Adding a queue manager to a queue-sharing group . . . . .	117

	Removing a queue manager from a queue-sharing group . . . . .	117
	Removing a queue-sharing group from the DB2 tables . . . . .	118
	Managing shared queues . . . . .	118
	Recovering shared queues . . . . .	118
	Moving shared queues . . . . .	118
	Migrating non-shared queues to shared queues	121
	Managing group objects . . . . .	122
	Managing the Coupling Facility . . . . .	122
	Adding a Coupling Facility structure . . . . .	122
	Removing a Coupling Facility structure . . . . .	122

## **Part 5. Recovery and restart. . . . 123**

### **Chapter 12. Restarting MQSeries . . . 125**

	Restarting after a normal shutdown . . . . .	125
	Restarting after an abnormal termination . . . . .	125
	Restarting if you have lost your page sets . . . . .	125
	Restarting if you have lost your log data sets. . . . .	126
	Alternative site recovery. . . . .	127
	Reinitializing MQSeries . . . . .	130
	Reinitializing a queue manager that is not in a queue-sharing group . . . . .	130
	Reinitializing queue managers in a queue-sharing group . . . . .	131

### **Chapter 13. Using the OS/390 Automatic Restart Manager (ARM) . . . 133**

	What is the ARM?. . . . .	133
	ARM couple data sets . . . . .	134
	ARM policies . . . . .	134
	Defining an ARM policy. . . . .	134
	Activating an ARM policy . . . . .	135
	Using ARM in an MQSeries network . . . . .	136
	Restarting on a different OS/390 image with LU 6.2 . . . . .	136
	Restarting on a different OS/390 image with TCP/IP . . . . .	137

### **Chapter 14. Recovering units of work manually . . . . . 139**

	Displaying connections and threads . . . . .	139
	Active threads . . . . .	140
	In-doubt threads . . . . .	140
	Recovering CICS units of recovery manually . . . . .	141
	What happens when the CICS adapter restarts	141
	How to resolve CICS units of recovery manually	143
	Recovering IMS units of recovery manually . . . . .	145
	What happens when the IMS adapter restarts	145
	How to resolve IMS units of recovery manually	145
	Recovering RRS units of recovery manually . . . . .	147
	Recovering units of recovery on another queue manager in the queue-sharing group . . . . .	148

### **Chapter 15. Example recovery scenarios . . . . . 149**

	Shared queue problems . . . . .	150
	Queue is both private and shared . . . . .	150

	Active log problems . . . . .	151
	Dual logging is lost . . . . .	151
	Active log stopped . . . . .	151
	One or both copies of the active log data set are damaged . . . . .	152
	Write I/O errors on an active log data set . . . . .	153
	I/O errors occur while reading the active log	153
	Active log is becoming full or is full. . . . .	155
	Archive log problems. . . . .	157
	Allocation problems . . . . .	157
	Off-load task terminated abnormally . . . . .	157
	Insufficient DASD space to complete off-load processing . . . . .	158
	Read I/O errors on the archive data set while MQSeries is restarting . . . . .	159
	BSDS problems. . . . .	160
	Error occurs while opening the BSDS . . . . .	160
	Log content does not agree with the BSDS information . . . . .	161
	Both copies of the BSDS are damaged . . . . .	161
	Unequal time stamps. . . . .	162
	Out of synchronization . . . . .	162
	I/O error. . . . .	163
	Page set problems. . . . .	164
	Page set I/O errors . . . . .	164
	Page set full. . . . .	165
	Coupling Facility and DB2 problems . . . . .	166
	Storage medium full . . . . .	166
	A DB2 system fails . . . . .	166
	A DB2 data-sharing group fails . . . . .	167
	DB2 and the Coupling Facility fail . . . . .	168
	Problems with long-running units of work . . . . .	169
	Old unit of work found during restart . . . . .	169
	IMS-related problems. . . . .	170
	IMS is unable to connect to MQSeries . . . . .	170
	IMS application problem . . . . .	170
	IMS is not operational . . . . .	171
	Hardware problems . . . . .	172

## **Part 6. Using the MQSeries Utilities . . . . . 173**

### **Chapter 16. Using the MQSeries utilities . . . . . 175**

	How to read syntax diagrams . . . . .	176
--	---------------------------------------	-----

### **Chapter 17. MQSeries utility program (CSQUTIL). . . . . 179**

	Invoking the MQSeries utility program. . . . .	180
	PARM parameters. . . . .	180
	Monitoring the progress of the MQSeries utility program . . . . .	182
	Formatting page sets (FORMAT) . . . . .	183
	Expanding a page set (COPYPAGE) . . . . .	185
	Copying a page set and resetting the log (RESETPAGE) . . . . .	187
	Issuing commands to MQSeries (COMMAND) . . . . .	190
	Producing a list of MQSeries define commands (SDEFS) . . . . .	195

Copying queues into a data set while the queue manager is running (COPY) . . . . .	198
Copying queues into a data set while the queue manager is not running (SCOPY). . . . .	201
Emptying a queue of all messages (EMPTY) . . . . .	204
Restoring messages from a data set to a queue (LOAD) . . . . .	206

**Chapter 18. The change log inventory utility (CSQJU003) . . . . . 209**

Invoking the CSQJU003 utility. . . . .	209
Data definition (DD) statements . . . . .	209
Multiple statement operation . . . . .	210
Adding information about a data set to the BSDS (NEWLOG) . . . . .	211
Deleting information about a data set from the BSDS (DELETE) . . . . .	214
Supplying a password for archive log data sets (ARCHIVE) . . . . .	215
Controlling the next restart (CRESTART) . . . . .	216
Setting checkpoint records (CHECKPT). . . . .	217
Updating the highest written log RBA (HIGHRBA) . . . . .	218

**Chapter 19. The print log map utility (CSQJU004) . . . . . 219**

Invoking the CSQJU004 utility. . . . .	219
Data definition statements . . . . .	219

**Chapter 20. The log print utility (CSQ1LOGP). . . . . 221**

Invoking the CSQ1LOGP utility . . . . .	221
Input control parameters . . . . .	222
Output . . . . .	223

**Chapter 21. The queue-sharing group utility (CSQ5PQSG) . . . . . 225**

Invoking the queue-sharing group utility . . . . .	225
Data definition statements . . . . .	225
Keywords and parameters . . . . .	225
Example . . . . .	226

**Chapter 22. The dead-letter queue handler utility (CSQUDLQH) . . . . . 227**

Invoking the DLQ handler . . . . .	227
------------------------------------	-----

Data definition statements . . . . .	228
Sample JCL . . . . .	228
The DLQ handler rules table . . . . .	228
Control data. . . . .	229
Rules (patterns and actions) . . . . .	230
Rules table conventions . . . . .	233
Processing the rules table . . . . .	235
Ensuring that all DLQ messages are processed . . . . .	236
An example DLQ handler rules table . . . . .	237

**Part 7. Appendixes . . . . . 239**

**Appendix A. User messages on start-up . . . . . 241**

**Appendix B. Notices . . . . . 245**

Programming interface information . . . . .	247
Trademarks . . . . .	248

**Glossary of terms and abbreviations 249**

**Bibliography. . . . . 259**

MQSeries cross-platform publications . . . . .	259
MQSeries platform-specific publications . . . . .	259
Softcopy books . . . . .	260
HTML format . . . . .	260
Portable Document Format (PDF) . . . . .	260
BookManager <sup>®</sup> format . . . . .	261
PostScript format . . . . .	261
Windows Help format . . . . .	261
MQSeries information available on the Internet . . . . .	261
Related publications . . . . .	261
OS/390 . . . . .	261
CICS Transaction Server for OS/390. . . . .	261
CICS for MVS/ESA Version 4 . . . . .	261
DB2 . . . . .	261
IMS . . . . .	261
DFSMS/MVS . . . . .	261

**Index . . . . . 263**

**Sending your comments to IBM . . . . . 271**

## Figures

1.	Issuing a DISPLAY command from the OS/390 console . . . . .	4
2.	The MQSeries operations and control initial panel. . . . .	13
3.	Listing queues . . . . .	14
4.	Defining a local queue - first panel. . . . .	15
5.	Defining a local queue - second panel. . . . .	16
6.	Defining a local queue - trigger conditions . . . . .	17
7.	Defining a local queue - event control. . . . .	17
8.	Defining a local queue - backout reporting . . . . .	18
9.	Starting the MQSeries subsystem from an OS/390 console . . . . .	22
10.	Sample start-up procedure . . . . .	23
11.	Stopping MQSeries . . . . .	24
12.	Padding adapter commands . . . . .	48
13.	The CICS adapter control initial panel . . . . .	50
14.	Starting a connection . . . . .	51
15.	Starting a connection from the command line . . . . .	52
16.	Starting a connection from the command line specifying parameters . . . . .	52
17.	Specifying lowercase queue names. . . . .	52
18.	Linking to the adapter connect program, CSQCQCON, from a CICS program . . . . .	53
19.	Stopping a connection from the CKQC initial panel. . . . .	54
20.	Stopping a connection from the command line—a quiesced shutdown . . . . .	55
21.	Stopping a connection from the command line—a forced shutdown . . . . .	55
22.	Stopping a connection from a CICS application program—a quiesced shutdown. . . . .	55
23.	Stopping a connection from a CICS application program—a forced shutdown . . . . .	55
24.	Modifying a connection . . . . .	56
25.	Format of command to modify connection parameters from the command line . . . . .	57
26.	Resetting connection statistics from the command line. . . . .	57
27.	Changing the adapter's trace number and disabling the API-crossing exit from the command line. . . . .	57
28.	Format of the MODIFY command issued from a CICS adapter application program . . . . .	58
29.	Resetting connection statistics from a CICS program . . . . .	58
30.	Linking to the adapter reset program, CSQCRST, from a CICS program . . . . .	58
31.	The display connection panel . . . . .	59
32.	Starting an instance of CKTI . . . . .	60
33.	Starting an instance of CKTI—for the default initiation queue . . . . .	61
34.	Starting an instance of CKTI—for a specified initiation queue . . . . .	61
35.	Linking to the adapter task-initiator program CSQCSSQ from CICS . . . . .	61
36.	Linking to the adapter task-initiator program CSQCSSQ from CICS . . . . .	61
37.	Stopping an instance of the task initiator CKTI . . . . .	62
38.	Stopping an instance of CKTI from the command line—for the default initiation queue . . . . .	63
39.	Stopping an instance of CKTI from the command line—for a specified initiation queue . . . . .	63
40.	Stopping an instance of CKTI from a program—for the default initiation queue from CICS. . . . .	63
41.	Stopping an instance of CKTI from a program—for a specified initiation queue from CICS. . . . .	63
42.	The CKQC Display CKTI panel . . . . .	64
43.	The CKQC Display Task panel . . . . .	65
44.	Message showing the status of a connection . . . . .	66
45.	Displaying the status of a connection . . . . .	66
46.	Linking to the adapter program CSQCDSPL from a CICS program . . . . .	66
47.	Extract from a load balancing job for a page set . . . . .	108
48.	Extract from a load balancing job for a Coupling Facility structure . . . . .	119
49.	Sample job for moving a non-shared queue to a shared queue . . . . .	120
50.	Sample job for moving a shared queue to a non-shared queue . . . . .	120
51.	Sample job for moving a non-shared queue without messages to a shared queue . . . . .	121
52.	Moving messages from a non-shared queue to an existing shared queue. . . . .	121
53.	Sample input statements for CSQJU003 . . . . .	128
54.	Sample ARM policy . . . . .	134
55.	Example restart messages . . . . .	142
56.	How to invoke the CSQUTIL utility program . . . . .	180
57.	Sample JCL for the FORMAT function of CSQUTIL . . . . .	184
58.	Sample JCL showing the use of the COPYPAGE function . . . . .	186
59.	Sample JCL showing the use of the RESETPAGE function. . . . .	188
60.	Sample JCL for issuing MQSeries commands using CSQUTIL. . . . .	192
61.	Sample JCL for using the MAKEDEF option of the COMMAND function . . . . .	192
62.	Sample JCL for using the MAKECLNT option of the COMMAND function . . . . .	193
63.	Sample JCL for the SDEFS function of CSQUTIL . . . . .	196
64.	Sample JCL for the SDEFS function of CSQUTIL for objects in the DB2 shared repository. . . . .	197
65.	Sample JCL for the CSQUTIL COPY functions . . . . .	199
66.	Sample JCL for the CSQUTIL SCOPY functions . . . . .	202

67. Sample JCL for the CSQUTIL EMPTY function . . . . .	204		76. Specifying the queue manager and dead-letter queue names for the dead-letter queue handler in the JCL . . . . .	227
68. Sample JCL for the CSQUTIL LOAD function	207			
69. Sample JCL to invoke the CSQJU003 utility	209		77. Specifying the queue manager and dead-letter queue names for the dead-letter queue handler in the rules table . . . . .	227
70. Sample JCL to invoke the CSQJU004 utility	219			
71. Sample JCL to invoke the CSQ1LOGP utility using a BSDS . . . . .	221		78. Sample JCL to invoke the CSQUDLQH utility	228
72. Sample JCL to invoke the CSQ1LOGP utility using active log data sets . . . . .	222		79. An example rule from a DLQ handler rules table . . . . .	230
73. Sample JCL to invoke the CSQ1LOGP utility using archive logs . . . . .	222			
74. Sample JCL to invoke the CSQ5PQSG utility	225		80. MQSeries startup messages for subsystem CSQ1 . . . . .	241
75. Using the queue-sharing group utility to add a queue manager into a queue-sharing group .	226			



---

## Tables

1. Valid operations and control panel actions for MQSeries objects . . . . .	9	4. A summary of MQSeries utilities . . . . .	175
2. Shutting down a CICS adapter connection . . . . .	68	5. How to read syntax diagrams . . . . .	176
3. Example recovery scenarios . . . . .	149	6. SDEFS QSGDISP parameters and their actions . . . . .	195



---

## About this book

This book describes how to operate MQSeries® for OS/390® using commands, panels, and utilities, and how to write applications to administer MQSeries. The latter part of the book deals with termination, recovery, and restart. Read these sections when you need to perform such tasks.

This book is based on parts of the *MQSeries for OS/390 Version 2.1 System Management Guide*.

The *System Management Guide* has been replaced by the following three books:

- *MQSeries for OS/390 Concepts and Planning Guide*. This book describes the concepts of MQSeries for OS/390; it does not describe the concepts of MQSeries messaging and queueing. If you are not familiar with these concepts, you should read *MQSeries: An Introduction to Messaging and Queueing*. It also describes how to plan your MQSeries for OS/390 systems.
- *MQSeries for OS/390 System Setup Guide*. This book describes the tasks that you have to perform to customize MQSeries after you have installed it. It also describes how to monitor system use and performance, and how to set up security.
- *MQSeries for OS/390 System Administration Guide*. (This book.)

Changes to the information in this book since the last edition of the *System Management Guide* are marked with vertical bars in the left-hand margin.

---

## Who this book is for

This book is intended for system programmers and system administrators.

---

## What you need to know to understand this book

This book assumes you are familiar with the basic concepts of:

- CICS®
- DB2® (if you intend to use queue-sharing groups)
- IMS™
- OS/390 job control language (JCL)
- OS/390 Time Sharing Option (TSO)
- The OS/390 Coupling Facility (if you intend to use queue-sharing groups)

If you want to write programs to administer MQSeries, this book assumes that you can write programs in one of the supported languages:

- COBOL
- C
- C++
- Assembler
- PL/I

You do not need to have written message-queuing programs previously.

### Conventions used in this book

- Throughout this book, the term *object* refers to any MQSeries queue manager, queue, namelist, channel, storage class, or process.
- The examples in this book are taken from a queue manager with a command prefix string (CPF) of +CSQ1. The commands are shown in UPPERCASE.
- CICS means both CICS Transaction Server for OS/390 and CICS for MVS/ESA™, IMS means IMS/ESA®.
- Throughout this book, the default value thlqual is used to indicate the target library high-level qualifier for MQSeries data sets in your installation.

---

### What's new for this release

This section describes the new function that has been added for this release of MQSeries for OS/390.

#### Queue-sharing groups

- You can group your queue managers into a queue-sharing group. These queue managers can access the same set of shared queues. This is described in the *MQSeries for OS/390 Concepts and Planning Guide*.
- A new system parameter (QSGDATA) sets the queue-sharing group parameters for your queue manager. This is described in the *MQSeries for OS/390 System Setup Guide*.
- You can now define an object once on one queue manager and then use the object definition on other queue managers in the queue-sharing group. This is described in the *MQSeries for OS/390 Concepts and Planning Guide*.
- You can now send commands to all queue managers in a queue-sharing group by issuing the command on one member of the group. This is described in the *MQSeries for OS/390 Concepts and Planning Guide*.
- You can now use intra-group queuing to send messages between queue managers in a queue-sharing group, without setting up channels. This is described in the *MQSeries for OS/390 Concepts and Planning Guide*.

#### Channel initiator

- You can now use shared channels. This is described in the *MQSeries for OS/390 Concepts and Planning Guide*.
- You can now stop a receiver channel automatically and start a new one in its place when a request to start a duplicate receiver channel is received by using the ADOPTMCA and ADOPTCHK parameters of the CSQ6CHIP channel initiator parameter macro. This is described in the *MQSeries Intercommunication manual*; CSQ6CHIP is described in the *MQSeries for OS/390 System Setup Guide*. (This function was available as a PTF for previous releases.)
- You can now specify a range of port numbers to be used when binding outbound channels by using the OPORTMIN and OPORTMAX parameters of the CSQ6CHIP channel initiator parameter macro. This is described in the *MQSeries for OS/390 System Setup Guide*. (This function was available as a PTF for previous releases.)
- You can now specify a single IP address upon which the TCP/IP listener accepts inbound connection requests. This is described in the *MQSeries MQSC Command Reference*.
- You can now specify that a TCP listener should register with Workload Manager for Dynamic Domain Name Services by using the DNSWLM and DNSGROUP

parameters of the CSQ6CHIP channel initiator parameter macro. This is described in the *MQSeries for OS/390 System Setup Guide*.

- You can now specify an LU name that is defined as part of a generic resource for an LU 6.2 listener by using the LUGROUP parameter of the CSQ6CHIP channel initiator parameter macro. This is described in the *MQSeries for OS/390 System Setup Guide*.

## Commands

- The following commands have been added:

<b>CLEAR QLOCAL</b>	Clear messages from a local queue.
<b>DISPLAY GROUP</b>	Display information about the queue-sharing group to which the queue manager is connected.
<b>DISPLAY QSTATUS</b>	Display queue status information.
<b>MOVE QLOCAL</b>	Move messages from one queue to another.
<b>RESET QSTATS</b>	Display information about how many messages have been put to and retrieved from a queue.

These commands are described in the *MQSeries MQSC Command Reference*.

- The QSGDISP attribute has been added to many commands to specify the object disposition (described in the *MQSeries for OS/390 Concepts and Planning Guide*).
- The CHLDISP attribute has been added to the channel commands to specify the channel disposition (described in the *MQSeries MQSC Command Reference*).
- The CMDSCOPE attribute has been added to most commands to specify the command scope (described in the *MQSeries for OS/390 Concepts and Planning Guide*).
- The responses to many commands have changed.

## System parameters

- The following new system parameters have been added (they are all described elsewhere in this chapter):

<b>CSQ6SYSP</b>	QSGDATA, RESAUDIT
<b>CSQ6LOGP</b>	DEALLCT, MAXRTU
<b>CSQ6ARVP</b>	UNIT2

- The default settings for the following system parameters have changed:

<b>CSQ6SYSP</b>	The default for LOGLOAD is now 500 000.
<b>CSQ6LOGP</b>	The default for INBUFF is now 60 KB and the default for OUTBUFF is now 4 000 KB.

These are described in the *MQSeries for OS/390 System Setup Guide*.

- The MAXALLC parameter of CSQ6LOGP is no longer used.

## About this book

### System object samples

- A new system object sample (CSQ4INSS) that defines the objects required for queue-sharing groups has been added (described in the *MQSeries for OS/390 Concepts and Planning Guide*).
- The default buffer pool, storage class and page set definitions have been changed. These are explained in the CSQ4INP1 and CSQ4INYG sample data sets (described in the *MQSeries for OS/390 Concepts and Planning Guide*).
- The sample object data set for the basic IVP has been renamed to CSQ4IVPQ, and a new sample object data set called CSQ4IVPG has been added for the queue-sharing group IVP.
- The default region sizes have been increased to 0M for the queue manager and channel initiator address spaces.

### Logs

- The default settings for log placement and size have been changed. These changes are described in the CSQ4BSDS sample data set.
- You can now specify that you want both copies of the archive log to be written to different device types by using the UNIT and UNIT2 parameters of the CSQ6ARVP system parameter macro. This is described in the *MQSeries for OS/390 System Setup Guide*.
- You can now restart the log archive process after a failure by using the ARCHIVE LOG CANCEL OFFLOAD command. This command is described in the *MQSeries MQSC Command Reference*.
- You can now optimize archive log reading from tape devices using the MAXRTU and DEALLCT parameters of the CSQ6LOGP system parameter macro. You can display and reset these parameters using the DISPLAY LOG and SET LOG commands. CSQ6LOGP is described in the *MQSeries for OS/390 System Setup Guide*; the commands are described in the *MQSeries MQSC Command Reference*.

### Security

- You can now use one set of security profiles to control security for all the queue managers in a queue-sharing group. This is described in the *MQSeries for OS/390 Concepts and Planning Guide*.
- You can now control whether RACF<sup>®</sup> audit records are written for RESLEVEL security checks using the RESAUDIT parameter of the CSQ6SYSP system parameter module. This is described in the *MQSeries for OS/390 System Setup Guide*.

### Statistics and accounting

- You can now use the SMF data collection broadcast to gather MQSeries statistics and accounting records. This is described in the *MQSeries for OS/390 System Setup Guide*.
- You can now gather performance statistics for the Coupling Facility manager and DB2 manager. This is described in the *MQSeries for OS/390 System Setup Guide*.
- You can now gather accounting data at queue and thread level. This is described in the *MQSeries for OS/390 System Setup Guide*.

## Operations and control panels

- The operations and control panels have been changed extensively to include the new function. For example, you are now asked to enter the disposition of objects that you are working with, and this information is included when you use the panels to display an object. This is described in “Introducing the operations and control panels” on page 5 and in the help supplied with the operations and control panels.
- The DEFINE action has been changed to DEFINE LIKE.
- You can now get and collate information for the whole queue-sharing group.
- You can now display queue status information.

## Dead-letter queue

There is a new utility program (CSQUDLQH) which processes messages on the dead-letter queue. This is described in “Chapter 22. The dead-letter queue handler utility (CSQUDLQH)” on page 227.

## Application programming

- MQSeries messages can now be up to 100 MB in length. (This function was available as a PTF for previous releases.)
- The Application Messaging Interface (AMI) provides a simple interface to the Message Queue Interface (MQI). Application programmers can use this interface to write applications without needing to understand all the functions available in the MQI. The functions that are required in a particular installation are defined by a system administrator. This function is described in the *MQSeries Application Messaging Interface* manual.
- The MQRC\_STORAGE\_MEDIUM\_FULL return code has been added. This is described in the *MQSeries Application Programming Reference*.
- New code pages have been added, including one for UNICODE. These are described in the *MQSeries Application Programming Reference*.
- You can now specify options when you connect to a queue manager using the **MQCONN** call. This is described in the *MQSeries Application Programming Reference*.

## About this book



# Part 1. Operating MQSeries for OS/390

<b>Chapter 1. Operating MQSeries for OS/390</b> . . . . .	3	Defining a reply-to queue . . . . .	28
Issuing commands . . . . .	3	Opening the system-command input queue . . . . .	29
Issuing commands from an OS/390 console or its equivalent . . . . .	3	Opening a reply-to queue . . . . .	29
Command prefix strings . . . . .	4	Using the command server . . . . .	30
Using the OS/390 console to issue commands . . . . .	4	Identifying the queue manager that processes your commands . . . . .	30
Command responses. . . . .	4	Starting the command server . . . . .	30
Issuing commands from the utility program CSQUTIL . . . . .	4	Sending commands to the command server . . . . .	30
Introducing the operations and control panels . . . . .	5	Building a message that includes MQSeries commands. . . . .	30
Invoking the operations and control panels . . . . .	5	Command attributes . . . . .	31
Rules for the operations and control panels . . . . .	6	Putting messages on the system-command input queue . . . . .	32
Blank fields. . . . .	6	Using MQPUT1 and the system-command input queue . . . . .	32
Objects and actions . . . . .	7	Retrieving replies to your commands. . . . .	33
Queues, processes, namelists, and storage classes . . . . .	7	Waiting for a reply . . . . .	33
Channels . . . . .	7	Discarded messages . . . . .	34
Cluster objects. . . . .	8	The reply message descriptor . . . . .	34
Queue manager and security . . . . .	8	Interpreting the replies . . . . .	35
System . . . . .	8	Using the DISPLAY commands. . . . .	36
Actions . . . . .	9	Examples of commands and their replies . . . . .	37
Object dispositions . . . . .	10	Messages from a DEFINE command . . . . .	37
Choosing a queue manager . . . . .	10	Messages from a DELETE command . . . . .	37
Queue manager defaults . . . . .	11	Messages from DISPLAY commands . . . . .	37
Using the function keys . . . . .	11	Finding out the name of the dead-letter queue . . . . .	37
Getting things done. . . . .	11	Messages from the DISPLAY THREAD command . . . . .	38
Displaying MQSeries user messages . . . . .	11	Messages from the DISPLAY QUEUE command . . . . .	38
Ignoring what you have done . . . . .	11	Messages from the DISPLAY NAMELIST command . . . . .	39
Getting help . . . . .	12	Messages from commands with CMDSCOPE . . . . .	40
Using the command line . . . . .	12	Messages from the ALTER PROCESS command . . . . .	40
Using the operations and control panels. . . . .	13	Messages from the DISPLAY PROCESS command . . . . .	40
Defining objects . . . . .	14	Messages from the DISPLAY CHSTATUS command . . . . .	41
Defining a local queue. . . . .	15	Messages from the STOP CHANNEL command . . . . .	41
When your local queue definition is complete . . . . .	18	Messages from commands that generate commands with CMDSCOPE . . . . .	42
Defining other types of objects . . . . .	18	If you do not receive a reply. . . . .	43
Working with object definitions. . . . .	19	Passing commands using MGCR or MGCRE . . . . .	43
Altering an object definition. . . . .	19		
Displaying an object definition . . . . .	19		
Deleting an object . . . . .	19		
Working with namelists . . . . .	20		
<b>Chapter 2. Starting and stopping MQSeries.</b> . . . .	21		
Before you start MQSeries . . . . .	21		
Starting MQSeries . . . . .	22		
Start options . . . . .	22		
Starting after an abnormal termination . . . . .	23		
User messages on start-up . . . . .	23		
Stopping MQSeries . . . . .	24		
Stop messages . . . . .	25		
<b>Chapter 3. Writing programs to administer MQSeries.</b> . . . .	27		
Understanding how it all works . . . . .	27		
Before you begin . . . . .	27		
Preparing queues for administration programs . . . . .	28		
Defining the system-command input queue . . . . .	28		



---

## Chapter 1. Operating MQSeries for OS/390

This chapter describes the basic procedures you can use to operate MQSeries for OS/390. It discusses the following topics:

- “Issuing commands”
- “Introducing the operations and control panels” on page 5
- “Using the operations and control panels” on page 13
- “Defining objects” on page 14
- “Defining a local queue” on page 15
- “Defining other types of objects” on page 18
- “Working with object definitions” on page 19
- “Working with namelists” on page 20

---

### Issuing commands

You can control most of the operational environment of MQSeries using the MQSeries commands. For details of the syntax of the MQSeries commands, see the *MQSeries MQSC Command Reference* manual. If you are a suitably authorized user, you can issue MQSeries commands from:

- The initialization input data sets (described in the *MQSeries for OS/390 System Setup Guide*).
- An OS/390 console, or equivalent, such as SDSF
- The OS/390 master get command routine, MGCR and MGCRE (SVC 34)
- The MQSeries utility, CSQUTIL (described in “Chapter 17. MQSeries utility program (CSQUTIL)” on page 179.)
- A user application, which can be:
  - A CICS program
  - A TSO program
  - An OS/390 batch program
  - An IMS program

See “Chapter 3. Writing programs to administer MQSeries” on page 27 for information about this.

Much of the functionality of these commands is provided in a user-friendly way by the operations and control panels, accessible from TSO and ISPF, and described in “Introducing the operations and control panels” on page 5.

### Issuing commands from an OS/390 console or its equivalent

You can issue all MQSeries commands from an OS/390 console or its equivalent. This means you can also issue MQSeries commands from anywhere where you can issue OS/390 commands, such as SDSF or by a program using the MGCR macro.

The maximum amount of data that can be displayed as a result of a command typed in at the console is 32 KB.

#### Notes:

1. You cannot issue MQSeries commands using the IMS /SSR command format from an IMS terminal. This function is not supported by the IMS adapter.
2. The input field provided by SDSF might not be long enough for some commands, particularly those for channels.

## Issuing commands

### Command prefix strings

Each MQSeries command must be prefixed with a command prefix string (CPF), as shown in Figure 1.

Because more than one MQSeries subsystem can run under OS/390, the CPF is used to indicate which MQSeries subsystem processes the command. For example, to start a subsystem called CSQ1, whose CPF is '+CSQ1', you issue the command +CSQ1 START QMGR from the operator console. This CPF must be defined in the subsystem name table (for the subsystem CSQ1). This is described in the *MQSeries for OS/390 System Setup Guide*. In the examples, the string '+CSQ1' is used as the command prefix.

### Using the OS/390 console to issue commands

You can type simple commands from the OS/390 console, for example, the DISPLAY command in Figure 1. However, for complex commands or for sets of commands that you issue frequently, the other methods of issuing commands are better.

```
+CSQ1 DISPLAY QUEUE(TRANSMIT.QUEUE.PROD) TYPE(QLOCAL)
```

Figure 1. Issuing a DISPLAY command from the OS/390 console

### Command responses

Direct responses to commands are sent to the console that issued the command. MQSeries supports the *Extended Console Support (EMCS)* function available in OS/390, and therefore consoles with 4-byte IDs can be used. Additionally, all commands except START QMGR and STOP QMGR support the use of Command and Response Tokens (CARTs) when the command is issued by a program using the MGCRC macro.

## Issuing commands from the utility program CSQUTIL

You can issue commands from a sequential data set using the COMMAND function of the utility program CSQUTIL. This utility transfers the commands to the *system-command input queue* and waits for the response, which is printed together with the original commands in SYSPRINT. For details of this, see the "Chapter 17. MQSeries utility program (CSQUTIL)" on page 179.

---

## Introducing the operations and control panels

You can use the MQSeries operations and control panels to perform administration tasks on MQSeries objects. You use these panels to run commands for defining, displaying, altering, or deleting MQSeries objects. Use the panels for day-to-day administration and for making small changes to objects. If you are setting up or changing many objects, you should use the COMMAND function of the CSQUTIL utility program.

The operations and control panels support the system control commands for the channel initiator (for example, to start a channel or a TCP/IP listener), for clustering, and for security. They also enable you to display information about threads and page set usage.

### Notes:

1. A small number of system control commands are not available through the panels. These commands must be issued explicitly using one of the other methods, see “Issuing commands” on page 3
2. You cannot issue the MQSeries commands directly from the command line in the panels.
3. To use the operations and control panels, you must have the correct security authorization; this is described in the *MQSeries for OS/390 System Setup Guide*.

## Invoking the operations and control panels

If the ISPF/PDF primary options menu has been updated for MQSeries, you can access the MQSeries operations and control panels from that menu. For details about updating the menu, see the *MQSeries for OS/390 System Setup Guide*.

You can access the MQSeries operations and control panels from the TSO command processor panel (usually option 6 on the ISPF/PDF primary options menu). The name of the exec that you run to do this is CSQOREXX. It has two parameters; `thlqual` is the high-level qualifier for the MQSeries libraries to be used, and `langletter` is the letter identifying the national language libraries to be used (for example, E for U.S. English). The parameters can be omitted if the MQSeries libraries are permanently installed in your ISPF setup. Alternatively, you can issue CSQOREXX from the TSO command line.

These panels are designed to be used by operators and administrators with a minimum of formal training. Read these instructions with the panels running and try out the different tasks suggested.

**Note:** While using the panels, temporary dynamic queues with names of the form `SYSTEM.CSQOREXX.*` will be created.

## Operations and control panels

### Rules for the operations and control panels

The *MQSeries MQSC Command Reference* manual defines the general rules for MQSeries character strings and names. However, there are some rules that apply only to the operations and control panels:

- Do not enclose strings, for example descriptions, in single or double quotes.
- If you need to use a quote mark in a description or other text field, for example:

This is Maria's queue

use just one quote. The panel processor doubles them for you to pass them to MQSeries. However, if it has to truncate your data to do this, it will do so.

- You can use uppercase or lowercase characters in most fields, and they are translated to uppercase characters when you press Enter. The exceptions are:
  - Storage class names and Coupling Facility structure names, which must start with uppercase A through Z and be followed by uppercase A through Z or numeric characters.
  - Certain fields that are not translated. These include:
    - Application ID
    - Description
    - Environment data
    - Object names (but if you use a lowercase object name, you might not be able to enter it at an OS/390 console)
    - Remote system name
    - Trigger data
    - User data
- In names, leading blanks and leading underscores are ignored. Therefore, you cannot have object names beginning with blanks or underscores.
- Underscores are used to show the extent of blank fields. When you press Enter, trailing underscores are replaced by blanks.
- Many description and text fields are presented in multiple parts, each part being handled by MQSeries independently. This means that trailing blanks *are retained* and the text is not contiguous.

#### Blank fields

When you specify the **define** action for an MQSeries object, each field on the define panel contains a value. See the general help (extended help) for the display panels for information on where MQSeries gets the values. If you type over a field with blanks, and blanks are not allowed, MQSeries puts the installation default value in the field or prompts you to enter the required value.

When you specify the **Alter** action for an MQSeries object, each field on the alter panel contains the current value for that field. If you type over a field with blanks, and blanks are not allowed, the ALTER command fails and an error message is displayed, prompting you to enter the required value.

## Objects and actions

The operations and control panels offer you many different types of object and a number of actions that you can perform on them. The actions are listed on the initial panel and enable you to manipulate the objects and display information about them. These objects include all the MQSeries objects, together with some extra ones. The objects fall into five categories.

- Queues, processes, namelists, and storage classes
- Channels
- Cluster objects
- Queue manager and security
- System

### Queues, processes, namelists, and storage classes

These are the basic MQSeries objects. There can be many of each type. They can be defined and deleted, and have attributes that can be displayed and altered, using the DEFINE LIKE, MANAGE, DISPLAY, and ALTER actions.

This category consists of the following objects:

QLOCAL	Local queue
QREMOTE	Remote queue
QALIAS	Alias queue for indirect reference to a queue
QMODEL	Model queue for defining queues dynamically
QUEUE	Any type of queue
PROCESS	Information about an application to be started when a trigger event occurs
NAMELIST	List of names, such as queues or clusters
STGCLASS	Storage class
QSTATUS	Status of a local queue

### Channels

Channels are used for distributed queuing (not for the CICS mover). There can be many of each type, and they can be defined, deleted, displayed, and altered. They also have other functions available using the START, STOP and PERFORM actions. PERFORM provides reset, ping, and resolve channel functions.

This category consists of the following objects:

CHANNEL	Any type of channel
SENDER	Sender channel
SERVER	Server channel
RECEIVER	Receiver channel
REQUESTER	Requester channel
CLUSRCVR	Cluster-receiver channel
CLUSDR	Cluster-sender channel
SVRCONN	Server-connection channel
CLNTCONN	Client-connection channel

## Operations and control panels

### Cluster objects

Cluster objects are created automatically for queues and channels that belong to a cluster. The base queue and channel definitions can be on another queue manager. There can be many of each type, and names can be duplicated. They can only be displayed, using the DISPLAY action.

This category consists of the following objects:

CLUSQ	Cluster queue, created for a queue that belongs to a cluster
CLUSCHL	Cluster channel, created for a channel that belongs to a cluster
CLUSQMGR	Cluster queue manager, the same as a cluster channel but identified by its queue manager name

Cluster channels and cluster queue managers do have the PERFORM, START and STOP actions, but only indirectly through the DISPLAY action.

### Queue manager and security

These have a single instance. They have attributes that can be displayed and altered (using the DISPLAY and ALTER actions), and have other functions available using the PERFORM action.

This category consists of the following objects:

MANAGER	Queue manager – the PERFORM action provides suspend and resume cluster functions
SECURITY	Security functions – the PERFORM action provides refresh and reverify functions

### System

A collection of other functions. This category consists of the following objects:

SYSTEM	System functions
CONTROL	Synonym for SYSTEM

The functions available are:

DISPLAY	Display queue-sharing group, distributed queuing, thread, or page set usage information.
PERFORM	Refresh or reset clustering
START	Start the channel initiator or listeners
STOP	Stop the channel initiator or listeners



**Actions**

The actions that you can perform for each type of object are shown in the following table:

Table 1. Valid operations and control panel actions for MQSeries objects

Object	Alter	Define like	Manage (1)	Display	Perform	Start	Stop
CHANNEL	✓	✓	✓	✓	✓	✓	✓
CLNTCONN	✓	✓	✓	✓			
CLUSCHL				✓	✓(2)	✓(2)	✓(2)
CLUSQ				✓			
CLUSQMGR				✓	✓(2)	✓(2)	✓(2)
CLUSRCVR	✓	✓	✓	✓	✓	✓	✓
CLUSSDR	✓	✓	✓	✓	✓	✓	✓
CONTROL				✓	✓	✓	✓
MANAGER	✓			✓	✓		
NAMELIST	✓	✓	✓	✓			
PROCESS	✓	✓	✓	✓			
QALIAS	✓	✓	✓	✓			
QLOCAL	✓	✓	✓	✓			
QMODEL	✓	✓	✓	✓			
QREMOTE	✓	✓	✓	✓			
QSTATUS				✓			
QUEUE	✓	✓	✓	✓			
RECEIVER	✓	✓	✓	✓	✓	✓	✓
REQUESTER	✓	✓	✓	✓	✓	✓	✓
SECURITY	✓			✓	✓		
SENDER	✓	✓	✓	✓	✓	✓	✓
SERVER	✓	✓	✓	✓	✓	✓	✓
SVRCONN	✓	✓	✓	✓		✓	✓
STGCLASS	✓	✓	✓	✓			
SYSTEM				✓	✓	✓	✓

**Note:**

1. Provides **Delete** and other functions
2. Via **Display**

## Operations and control panels

### Object dispositions

You can specify the *disposition* of the object with which you need to work. The disposition signifies where the object **definition** is kept, and how the object behaves.

The disposition is significant only if you are working with any of the following object types:

- Queues
- Channels
- Processes
- Namelists
- Storage classes

If you are working with other object types, it is disregarded.

Permitted values are:

- Q** QMGR. The object definitions are on the page set of the queue manager and are accessible only by the queue manager.
- C** COPY. The object definitions are on the page set of the queue manager and are accessible only by the queue manager. They are local copies of objects defined as having a disposition of GROUP.
- P** PRIVATE. The object definitions are on the page set of the queue manager and are accessible only by the queue manager. The objects have been defined as having a disposition of QMGR or COPY.
- G** GROUP. The object definitions are in the shared repository, and are accessible by all queue managers in the queue-sharing group.
- S** SHARED. This disposition applies only to local queues. The queue definitions are in the shared repository, and are accessible by all queue managers in the queue-sharing group.
- A** ALL. If the action queue manager is either the target queue manager, or \*, objects of **all** dispositions are included; otherwise, objects of QMGR and COPY dispositions only are included. This is the default.

### Choosing a queue manager

While you are viewing the initial panel, you are not connected to any queue manager. However, as soon as you press Enter, you are connected to the queue manager or a queue manager in the queue-sharing group named in the “**Connect name**” field. You can leave this field blank; this means you are using the default queue manager for batch applications. This is defined in CSQBDEFV (see the *MQSeries for OS/390 System Setup Guide* for information about this).

Use the “**Target queue manager**” field to specify the queue manager where the actions you request are to be performed. If you leave this field blank, it defaults to the queue manager specified in the “**Connect name**” field. You can specify a target queue manager that is not the one you connect to. In this case, you would normally specify the name of a remote queue manager object that provides a queue manager alias definition (the name is used as the *ObjectQMgrName* when opening the command input queue). To do this, you must have suitable queues and channels set up to access the remote queue manager.

The “**Action queue manager**” allows you to specify a queue manager that is in the same queue-sharing group as the queue manager specified in the “**Target queue**”

`manager` field to be the queue manager where the actions you request are to be performed. If you specify `q` in this field, the actions you request are performed on all queue managers in the queue-sharing group. If you leave this field blank, it defaults to the value specified in the `Target queue manager` field. The `Action queue manager` field corresponds to using the CMDSCOPE command modifier described in the *MQSeries MQSC Command Reference*.

### Queue manager defaults

If you leave any queue manager field blank, or choose to connect to a queue-sharing group, a secondary window appears when you press Enter. This window confirms the names of the queue managers you will be using. Press Enter to continue. When you return to the initial panel after having made some requests, you find fields filled in with the actual names.

## Using the function keys

To use the panels, you must use the function keys or enter the equivalent commands in the command area. The function keys have special settings for MQSeries. (This means that you cannot use the ISPF default values for the function keys; if you have previously used the KEYLIST OFF ISPF command anywhere, you must type KEYLIST ON in the command area of any operations and control panel and then press Enter to enable the MQSeries settings.)

These function key settings can be displayed on the panels, as shown in Figure 2 on page 13. If the settings are not shown, type PFSHOW in the command area of any operations and control panel and then press Enter. To remove the display of the settings, use the command PFSHOW OFF.

The function key settings in the operations and control panels conform to CUA<sup>®</sup> standards. Although you can change the key setting through normal ISPF procedures (such as the KEYLIST utility) you are not recommended to do so.

**Note:** Using the PFSHOW and KEYLIST commands affects any other logical ISPF screens that you have, and their settings remain when you leave the operations and control panels.

### Getting things done

Press Enter to carry out the action requested on a panel. The information from the panel is sent to the queue manager for processing.

Each time you press Enter in the panels, MQSeries generates one or more operator messages. If the operation was successful, you get confirmation message CSQ9022I, otherwise you get some error messages.

### Displaying MQSeries user messages

Press function key F10 in any panel to see the MQSeries user messages.

### Ignoring what you have done

On the initial panel, both F3 and F12 exit the operations and control panels and return you to ISPF. No information is sent to the queue manager.

On any other panel, press function keys F3 or F12 to leave the current panel *ignoring any data you have typed since last pressing Enter*. Again, no information is sent to the queue manager.

- F3 takes you straight back to the initial panel.
- F12 takes you back to the previous panel.

## Operations and control panels

### Getting help

Each panel has help panels associated with it. The help panels use the ISPF protocols:

- Press function key F1 on any panel to see general help (extended help) about the task.
- Press function key F1 with the cursor on any field to see specific help about that field.
- Press function key F5 from any field help panel to get the general help.
- Press function key F3 to return to the base panel, that is, the panel from which you pressed function key F1.
- Press function key F6 from any help panel to get help about the function keys.

If the help information carries on into a second or subsequent pages, a **More** indicator is displayed in top right of the panel. Use these function keys to navigate through the help pages:

- F11 to get to the next help page (if there is one).
- F10 to get back to the previous help page (if there is one).

### Using the command line

| You never need to use the command line to issue the commands used by the  
| operations and control panels because they are available from function keys, as  
| described above. It is provided to allow you to enter normal ISPF commands (like  
| PFSHOW).

The command line is initially displayed at the bottom of the panels, regardless of what ISPF settings you have. If you prefer it to be at the top, use the **SETTINGS** ISPF command from any of the operations and control panels to change it. The settings will be remembered for subsequent sessions with the operations and control panels.

## Using the operations and control panels

Figure 2 shows the panel that is displayed when you start a panel session.

```

IBM MQSeries for OS/390 - Main Menu

Complete fields. Then press Enter.

Action . . . . . _      1.Display   4.Manage   6.Start
                        2.Define like 5.Perform  7.Stop
                        3.Alter

Object type . . . . . _____ +
Name . . . . . _____
Disposition . . . . . _  Q=Qmgr,C=Copy,P=Private
                        G=Group,S=Shared,A=All
Connect name . . . . . ____ - local queue manager or group
Target queue manager . . . _____
                        - connected or remote queue manager for command input
Action queue manager . . . ____ - command scope in group
Response wait time . . . . ____ 5 - 999 seconds

(C) Copyright IBM Corporation 1993,2000. All rights reserved.

Command ==> _____
F1=Help      F2=Split   F3=Exit     F4=Prompt   F9=Swap     F10=Messages
F12=Cancel

```

Figure 2. The MQSeries operations and control initial panel

From this panel you can:

- Choose the local queue manager you want and whether you want the commands issued on that queue manager, on a remote queue manager, or on another queue manager in the same queue-sharing group as the local queue manager. Overtyping the queue manager name if you need to change it.
- Select the action you want to perform by typing in the appropriate number in the **Action** field.
- Specify the object type that you want to work with. Press function key F1 for help about the object types if you are not sure what they are.
- Specify the disposition of the object type that you want to work with.
- Display a list of objects of the type specified. Type in an asterisk (\*) in the **Name** field and press Enter to display a list of objects (of the type specified) that have already been defined on the action queue manager. You can then select one or more objects to work with in sequence. Figure 3 on page 14 shows a list of queues produced in this way. All the actions are available from the list.
- Perform other actions.

## Operations and control panels

```

                                List Queues                                ROW 1 OF 12
Type action codes. Then press Enter.
1=Display 2=Define like 3=Alter 4=Manage

Name                               Type      Disposition
- CICS01.INITQ                      QLOCAL   QMGR   QM83
- PROTO.APPL                        QLOCAL   QMGR   QM83
- PROTO.TRIG                        QLOCAL   QMGR   QM83
- LOCAL.QUEUE                       QLOCAL   QMGR   QM83
- SYSTEM.CHANNEL.SEQNO              QLOCAL   QMGR   QM83
- SYSTEM.COMMAND.INPUT              QLOCAL   QMGR   QM83
- SYSTEM.COMMAND.REPLY.MODEL        QMODEL   QMGR   QM83
- SYSTEM.DEFAULT.ALIAS.QUEUE        QALIAS   QMGR   QM83
- SYSTEM.DEFAULT.LOCAL.QUEUE        QLOCAL   QMGR   QM83
- SYSTEM.DEFAULT.MODEL.QUEUE        QMODEL   QMGR   QM83
- SYSTEM.DEFAULT.REMOTE.QUEUE       QREMOTE  QMGR   QM83
- TRANSMIT.QUEUE.PROD               QLOCAL   QMGR   QM83
                                ***** End of list *****

Command ==>
F1=Help      F2=Split    F3=Exit     F5=Refresh  F6=Clnsinfo F7=Bkwd
F8=Fwd       F9=Swap     F10=Messages F11=Status  F12=Cancel
```

Figure 3. Listing queues

## Defining objects

To define a new object, use an existing definition as the basis for it. You can do this in one of three ways:

- By selecting an object that is a member of a list displayed as a result of options selected on the initial panel. You then enter action type 2 (Define like) in the action field to the left of the selected object. Your new object has the attributes of the selected object, except the disposition. You can then change any attributes in your new object as you require.
- On the initial panel, select the “Define like” action type, enter the type of object that you are defining in the “Object type” field, and enter the name of a specific existing object in the “Name” field. Your new object has the same attributes as the object you named in the “Name” field, except the disposition. You can then change any attributes in your new object definition as you require.
- By selecting the “Define like” action type, specifying an object type and then leaving the “Name” field blank. You can then define your new object and it has the default attributes defined for your installation. You can then change any attributes in your new object definition as you require.

**Note:** You do not enter the name of the object you are defining on the initial panel, but on the “Define” panel you are presented with.

## Defining a local queue

To define a local queue object from the operations and control panels, use an existing queue definition as the basis for your new definition. There are several panels to complete. When you have completed *all* the panels and you are satisfied that the attributes are correct, press Enter to send your definition to the queue manager, which then creates the actual queue.

Use the **Define like** action either on the initial panel or against an object entry in a list displayed as a result of options selected on the initial panel.

For example, starting from the initial panel, complete these fields:

```
Action          2 (Define like)
Object type     QLOCAL
Name           QUEUE.YOU.LIKE. This is the name of the queue which provides the
              attributes for your new queue.
```

Press Enter to display the **Define a Local Queue** panel as shown in Figure 4. The queue name field is blank so that you can supply the name for the new queue. The description is that of the queue upon which you are basing this new definition. Overtyping this field with your own description for the new queue.

The values in the other fields are those of the queue upon which you are basing this new queue, except the disposition. You can overtype these fields as you require. For example, type Y in the Put enabled field (if it is not already Y) if suitably authorized applications can put messages on this queue.

Define a Local Queue

Complete fields, then press F8 for further fields, or Enter to define queue.

More: +

```
Queue name . . . . . _____
Disposition . . . . . Q G=Group,S=Shared,Q=Qmgr on QM83
Description . . . . . Queue upon which the new one is based
              _____

Put enabled . . . . . Y Y=Yes,N=No
Get enabled . . . . . Y Y=Yes,N=No
Usage . . . . . N N=Normal,X=XmitQ
Storage class . . . . . SYSTEM
CF structure name . . . . . _____
```

Command ==> \_\_\_\_\_

F1=Help    F2=Split    F3=Exit    F7=Bkwd    F8=Fwd    F9=Swap  
F10=Messages    F12=Cancel

Figure 4. Defining a local queue - first panel

## Operations and control panels

1 You get field help by moving the cursor into a field and pressing function key F1. Field help provides information about the values that can be used for each attribute.

When you have completed the first panel, press function key F8 to display the second panel, see Figure 5.

### Hints:

1. Do *not* press Enter at this stage, otherwise the queue will be created before you have a chance to complete the remaining fields. (If you do press Enter prematurely, do not worry; you can always alter your definition later on.)
2. Do not press function key F3 or F12 either, or the data you typed will be lost.

Press function key F8 repeatedly to see and complete the remaining panels, including the trigger definition, event control, and backout reporting panels.

```

                                Define a Local Queue

Press F7 or F8 to see other fields, or Enter to define queue.

                                                                    More: - +

Default persistence . . . . . N Y=Yes,N=No
Default priority . . . . . 5 0 - 9
Message delivery sequence . . P P=Priority,F=FIFO
Permit shared access . . . . Y Y=Yes,N=No
Default share option . . . . S E=Exclusive,S=Shared
Index type . . . . . N N=None,M=MsgId,C=CorrelId,T=MsgToken
Maximum queue depth . . . . . 10000 0 - 999999999
Maximum message length . . . 1000000 0 - 104857600

Cluster name . . . . . _____
Cluster namelist name . . . . _____
Default bind . . . . . 0 0=Open,N=Notfixed

Command ==> _____
F1=Help F2=Split F3=Exit F7=Bkwd F8=Fwd F9=Swap
F10=Messages F12=Cancel
```

Figure 5. Defining a local queue - second panel



```

Define a Local Queue
Press F7 or F8 to see other fields, or Enter to define queue.

More: - +

Trigger Definition
Trigger type . . . . . F F=First,E=Every,D=Depth,N=None
Trigger set . . . . . N Y=Yes,N=No
Trigger message priority . 0 0 - 9
Trigger depth . . . . . 1 1 - 99999999
Trigger data . . . . . _____
Process name . . . . . _____
Initiation queue . . . . . _____

Command ==> _____
F1=Help F2=Split F3=Exit F7=Bkwd F8=Fwd F9=Swap
F10=Messages F12=Cancel
    
```

Figure 6. Defining a local queue - trigger conditions

```

Define a Local Queue
Press F7 or F8 to see other fields, or Enter to define queue.

More: - +

Event Control
Queue full . . . . . E E=Enabled,D=Disabled
Upper queue depth . . . . D E=Enabled,D=Disabled
Threshold . . . . . 80 0 - 100 %
Lower queue depth . . . . D E=Enabled,D=Disabled
Threshold . . . . . 40 0 - 100 %
Service interval . . . . . N H=High,0=OK,N=None
Interval . . . . . 999999999 0 - 999999999 milliseconds

Command ==> _____
F1=Help F2=Split F3=Exit F7=Bkwd F8=Fwd F9=Swap
F10=Messages F12=Cancel
    
```

Figure 7. Defining a local queue - event control

## Operations and control panels

```

                                Define a Local Queue
Press F7 to see previous fields, or Enter to define queue.

                                More: -
Backout Reporting

Backout threshold . . . . . 0          0=No backout reporting
    Harden backout counter . . N  Y=Yes,N=No
    Backout requeue name . . . _____

Retention interval . . . . . 999999999 0 - 999999999 hours

Command ==> _____
F1=Help      F2=Split    F3=Exit      F7=Bkwd      F8=Fwd      F9=Swap
F10=Messages F12=Cancel

```

Figure 8. Defining a local queue - backout reporting

### When your local queue definition is complete

When your definition is complete, press Enter to send the information to the queue manager for processing. The queue manager creates the queue according to the definition you have supplied. If you do not want the queue to be created, press function key F3 to exit and cancel the definition.

---

### Defining other types of objects

To define other types of object, use an existing definition as the base for your new definition as explained in “Defining a local queue” on page 15.

Use the **Define like** action either on the initial panel or against an object entry in a list displayed as a result of options selected on the initial panel.

For example, starting from the initial panel, complete these fields:

Action	2 (Define like)
Object type	QALIAS, NAMELIST, PROCESS, CHANNEL, and so on.
Name	Leave blank or enter the name of an existing object of the same type.

Press Enter to display the corresponding DEFINE panels. Complete the fields as required and then press Enter again to send the information to the queue manager.

Like defining a local queue, defining another type of object generally requires several panels to be completed. Defining a namelist requires some additional work, as described in “Working with namelists” on page 20.

---

## Working with object definitions

When an object has been defined, you can specify an action in the **Action** field, to alter, display, or manage it.

In each case, you can either:

- Select the object you want to work with from a list displayed as a result of options selected on the initial panel. For example, having entered 1 in the “Action” field to display objects, “Queue” in the “Object type” field, and \* in the “Name” field, you are presented with a list of all queues defined in the system. You can then select from this list the queue with which you need to work.
- Start from the initial panel, where you specify the object you are working with by completing the **Object type** and **Name** fields.

### Altering an object definition

To alter an object definition, specify action 3 and press Enter to see the ALTER panels. These panels are very similar to the DEFINE panels. You can alter the values you want. When your changes are complete, press Enter to send the information to the queue manager.

### Displaying an object definition

If you want to see the details of an object without being able to change them, specify action 1 and press Enter to see the DISPLAY panels. Again, these panels are similar to the DEFINE panels except that you cannot change any of the fields. Change the object name to display details of another object.

### Deleting an object

To delete an object, specify action 4 (Manage) and the Delete action is one of the actions presented on the resulting menu. Select the Delete action.

You are asked to confirm your request. If you press function key F3 or F12, the request is cancelled. If you press Enter, the request is confirmed and passed to the queue manager. The object you specified is then deleted.

**Note:** You cannot delete most types of channel object unless the channel initiator is started.



---

## Chapter 2. Starting and stopping MQSeries

This chapter describes how to start and stop MQSeries. It discusses the following topics:

- “Before you start MQSeries”
- “Starting MQSeries” on page 22
- “Stopping MQSeries” on page 24

Starting and stopping MQSeries is relatively straightforward. When MQSeries stops under normal conditions, its last action is to take a termination checkpoint. This checkpoint, and the logs, give MQSeries the information it needs to restart.

This section discusses the START and STOP commands, and contains a brief overview of start up after an abnormal termination has occurred.

---

### Before you start MQSeries

After you have installed MQSeries, it is defined as a formal OS/390 subsystem. This message appears during any initial program load (IPL) of OS/390:

```
CSQ3110I +CSQ1 CSQ3UR00 - SUBSYSTEM ssnm READY FOR START COMMAND
```

where *ssnm* is the MQSeries subsystem name.

From now on, you can start MQSeries *from any OS/390 console that has been authorized to issue system control commands*; that is, an OS/390 SYS command group. The START command must be issued from the authorized console, and cannot be submitted through JES or TSO.

If you are using queue-sharing groups you must start RRS first, and then DB2, before you start MQSeries.

### Starting MQSeries

You start MQSeries by issuing a START QMGR command. However, you cannot successfully use the START command unless you have appropriate authority. See the *MQSeries for OS/390 System Setup Guide* for information about MQSeries security. Figure 9 shows examples of the START command. (Remember that you must prefix an MQSeries command with a command prefix string (CPF).)

```
+CSQ1 START QMGR
+CSQ1 START QMGR PARM(NEWLOG)
```

*Figure 9. Starting the MQSeries subsystem from an OS/390 console. The second example specifies a system parameter module name.*

See the *MQSeries MQSC Command Reference* manual for information about the syntax of this command.

You cannot run the MQSeries subsystem as a batch job or start it using an OS/390 START command. These methods are likely to start an address space for MQSeries that then abends. You also cannot start MQSeries from the CSQUTIL utility program or a similar user application.

You can, however, start MQSeries from an APF-authorized program by passing a START QMGR command to the OS/390 MGCR or MGCRC (SVC 34) service.

### Start options

When you start a queue manager, a special routine called the system parameter module is invoked. You can specify the name of a system parameter module if you use the PARM keyword. A system parameter module provides information specified when the queue manager was customized. In Figure 80 on page 241, the user message CSQY001I indicates the name of the system parameter module that was used, in this case, CSQ1ZPRM. For more information about this, see the *MQSeries for OS/390 System Setup Guide*.

You can also use the ENVPARM option to substitute one or more parameters in the JCL procedure for the queue manager.

For example, you can update your MQSeries startup procedure, so that the DDname CSQINP2 is a variable. This means that you can change the CSQINP2 DDname without changing the startup procedure. This is very useful for implementing changes, providing backouts for operators, and so on.

Suppose your start-up procedure for queue manager CSQ1 looked like Figure 10 on page 23.

```
//CSQ1MSTR PROC INP2=NORM
//MQMESA EXEC PGM=CSQYASCP
//STEPLIB DD DISP=SHR,DSN=thlqua1.SCSQANLE
// DD DISP=SHR,DSN=thlqua1.SCSQAUTH
// DD DISP=SHR,DSN=db2qua1.SDSNLOAD
//BSDS1 DD DISP=SHR,DSN=myqua1.BSDS01
//BSDS2 DD DISP=SHR,DSN=myqua1.BSDS02
//CSQP0000 DD DISP=SHR,DSN=myqua1.PSID00
//CSQP0001 DD DISP=SHR,DSN=myqua1.PSID01
//CSQP0002 DD DISP=SHR,DSN=myqua1.PSID02
//CSQP0003 DD DISP=SHR,DSN=myqua1.PSID03
//CSQINP1 DD DISP=SHR,DSN=myqua1.CSQINP(CSQ1INP1)
//CSQINP2 DD DISP=SHR,DSN=myqua1.CSQINP(CSQ1&INP2.)
//CSQOUT1 DD SYSOUT=*
//CSQOUT2 DD SYSOUT=*
```

Figure 10. Sample start-up procedure

If you then start the your queue manager with the command:

```
+CSQ1 START QMGR
```

the CSQINP2 actually used is a member called CSQ1NORM.

However, suppose you are putting a new suite of programs into production so that the next time you start queue manager CSQ1, the CSQINP2 definitions are to be taken from member CSQ1NEW. To do this, you would start MQSeries with this command:

```
+CSQ1 START QMGR ENVPARM('INP2=NEW')
```

and CSQ1NEW would be used instead of CSQ1NORM. Note that OS/390 limits the KEYWORD=value specifications for symbolic parameters (as in INP2=NEW) to 48 characters.

## Starting after an abnormal termination

MQSeries automatically detects whether restart follows a normal shutdown or an abnormal termination.

Starting MQSeries after it abends is different from starting it after the +CSQ1 STOP QMGR command has been issued. After +CSQ1 STOP QMGR, the system finishes its work in an orderly way and takes a termination checkpoint before stopping. When you restart MQSeries, it uses information from the system checkpoint and recovery log to determine the system status at shutdown.

However, if MQSeries abends, it terminates without being able to finish its work or take a termination checkpoint. When you restart MQSeries after an abend, it refreshes its knowledge of its status at termination using information in the log, and notifies you of the status of various tasks. Normally, the restart process resolves all inconsistent states. But, in some cases, you must take specific steps to resolve inconsistencies.

## User messages on start-up

When you start MQSeries successfully, it produces a set of start up messages similar to the ones in "Appendix A. User messages on start-up" on page 241.

### Stopping MQSeries

Before stopping MQSeries, all MQSeries-related write-to-operator-with-reply (WTOR) messages must receive replies, for example, getting log requests. Each of the commands in Figure 11 terminates a running MQSeries subsystem.

```
+CSQ1 STOP QMGR
+CSQ1 STOP QMGR MODE(QUIESCE)
+CSQ1 STOP QMGR MODE(FORCE)
+CSQ1 STOP QMGR MODE(RESTART)
```

Figure 11. Stopping MQSeries

The command `+CSQ1 STOP QMGR` defaults to `+CSQ1 STOP QMGR MODE(QUIESCE)`.

In QUIESCE mode, MQSeries does not allow any new connection threads to be created, but allows existing threads to continue; it terminates only when all threads have ended. Applications can request to be notified in the event of the queue manager quiescing. Therefore, use the QUIESCE mode where possible so that applications that have requested notification have the opportunity to disconnect. See the *MQSeries Application Programming Guide* for details.

If MQSeries does not terminate in a reasonable time in response to a `+CSQ1 STOP QMGR MODE(QUIESCE)` command, use the `+CSQ1 DISPLAY THREAD(*) TYPE(ACTIVE)` command to determine whether any connection threads exist, and take the necessary steps to terminate the associated applications. If there are no threads then issue a `+CSQ1 STOP QMGR MODE(FORCE)` command.

The `+CSQ1 STOP QMGR MODE(QUIESCE)` and `+CSQ1 STOP QMGR MODE(FORCE)` commands deregister MQSeries from the MVS™ Automatic Restart Manager (ARM), preventing ARM from restarting the queue manager automatically. The `+CSQ1 STOP QMGR MODE(RESTART)` command works in the same way as the `+CSQ1 STOP QMGR MODE(FORCE)` command, except that it does not deregister MQSeries from ARM. This means that the queue manager is eligible for immediate automatic restart.

If the MQSeries subsystem is not registered with ARM, the `STOP QMGR MODE(RESTART)` command is rejected and the following message sent to the OS/390 console:

```
CSQY205I  ARM element arm-element is not registered
```

If this message is not issued, the queue manager is restarted automatically. For more information about ARM, see “Chapter 13. Using the OS/390 Automatic Restart Manager (ARM)” on page 133.

**Do not cancel the MQSeries address space unless `+CSQ1 STOP QMGR MODE(FORCE)` does not cause MQSeries to terminate.**

If MQSeries is stopped by either canceling the address space or by using the command `+CSQ1 STOP QMGR MODE(FORCE)`, consistency is maintained with



## Starting and stopping MQSeries

connected CICS or IMS systems. Resynchronization of resources is started when MQSeries restarts and is completed when the connection to the CICS or IMS system is established.

**Note:** When you stop your MQSeries subsystem, you might find message IEF352I is issued. OS/390 issues this message if it detects that failing to mark the address space as unusable would lead to an integrity exposure. You can ignore this message.

### Stop messages

After issuing a +CSQ1 STOP QMGR command, you get the messages CSQY009I and CSQY002I, for example:

```
CSQY009I +CSQ1 ' STOP QMGR' COMMAND ACCEPTED FROM  
USER(userid), STOP MODE(FORCE)  
CSQY002I +CSQ1 SUBSYSTEM STOPPING
```

Where *userid* is the user ID that issued the +CSQ1 STOP QMGR command, and the MODE parameter depends on that specified in command.

When the STOP command has completed successfully, these messages are displayed on the OS/390 console:

```
CSQ9022I +CSQ1 CSQYASCP ' STOP QMGR' NORMAL COMPLETION  
CSQ3104I +CSQ1 CSQ3EC0X - TERMINATION COMPLETE
```

If you are using ARM, the following message is also displayed if you did not specify MODE(RESTART):

```
CSQY204I +CSQ1 ARM DEREGISTER for element arm-element type  
arm-element-type successful
```

You cannot restart MQSeries until the following message has been generated:

```
CSQ3100I +CSQ1 CSQ3EC0X - SUBSYSTEM ssnm READY FOR START COMMAND
```



---

## Chapter 3. Writing programs to administer MQSeries

### Start of General-use programming interface information

This chapter contains hints and guidance to enable you to issue MQSeries commands from an MQSeries application program.

It contains these sections:

- “Understanding how it all works”
- “Preparing queues for administration programs” on page 28
- “Using the command server” on page 30
- “Retrieving replies to your commands” on page 33
- “Interpreting the replies” on page 35
- “Using the DISPLAY commands” on page 36
- “Examples of commands and their replies” on page 37
- “If you do not receive a reply” on page 43
- “Passing commands using MGCR or MGCRE” on page 43

**Note:** In this chapter, the MQI calls are described using C-language notation. For typical invocations of the calls in the COBOL, PL/I, and assembler languages, see the *MQSeries Application Programming Reference* manual.

---

### Understanding how it all works

In outline, the procedure for issuing commands from an application program is quite simple:

1. You build an MQSeries command into a type of MQSeries message called a *request message*.
2. You put (**MQPUT**) this message onto a special queue called the system-command input queue. The MQSeries command processor runs the command.
3. You retrieve (**MQGET**) the results of the command as *reply messages* on the reply-to queue. These messages contain the user messages that you need to determine whether your command was successful and, if it was, what the results were.

Then it is up to your application program to process the results.

### Before you begin

Before you can write an application program to issue MQSeries commands, you must be familiar with:

1. Issuing MQSeries commands and the command syntax. See the *MQSeries MQSC Command Reference* manual for more information.
2. Writing application programs that use the MQI.

This includes:

- Connecting to a queue manager using the **MQCONN** or **MQCONNX** call.
- Opening a queue using **MQOPEN**.
- Opening a dynamic queue using **MQOPEN** and specifying the name of a model queue.
- Putting messages on a queue using **MQPUT** and **MQPUT1**.

## Writing administration programs

- Getting messages from a queue using **MQGET**.

You need to know about the messages including:

- The message descriptor structure.
- What the persistence attribute of a message means.
- The types of MQSeries messages, in particular, *request messages* and the *reply messages* they generate.

You can find all this information in the *MQSeries Application Programming Guide* and the *MQSeries Application Programming Reference* manual.

### 3. User messages.

These messages are generated by MQSeries to show the success or failure of, and the responses to, MQSeries commands. Each message is identified by an ID that contains the characters CSQ, for example, CSQN205I. For more information, see the *MQSeries for OS/390 Messages and Codes* manual.

If you want your MQSeries commands to be run on a remote queue manager, see the *MQSeries Intercommunication* manual.

MQSeries can also be set up to perform security checks. For example, to ensure that a user is authorized to issue a particular command for a particular resource. For more information, see the *MQSeries for OS/390 System Setup Guide*.

---

## Preparing queues for administration programs

Before you can issue any **MQPUT** or **MQGET** calls, you must first define, and then open, the queues you are going to use.

### Defining the system-command input queue

The system-command input queue is a local queue called `SYSTEM.COMMAND.INPUT`. The supplied `CSQINP2` initialization data set, `thlqual.SCSQPROC(CSQ4INSG)`, contains a default definition for the system-command input queue. See the *MQSeries for OS/390 System Setup Guide* for more information.

### Defining a reply-to queue

You must define a reply-to queue to receive reply messages from the MQSeries command processor. It can be any queue whose attributes allow reply messages to be put on it. However, for normal operation, specify these attributes:

- `MAXSMSGL(13000)`
- `USAGE(NORMAL)`
- `NOTRIGGER` (unless your application uses triggering)

You should not normally use persistent messages for commands, but if you choose to do so, the reply-to queue must not be a temporary dynamic queue.

The supplied `CSQINP2` initialization data set, `thlqual.SCSQPROC(CSQ4INSG)`, contains a definition for a model queue called `SYSTEM.COMMAND.REPLY.MODEL`. You can use this model to create a dynamic reply-to queue.

**Note:** Replies generated by the command processor can be up to 13 000 bytes in length.

## Opening the system-command input queue

Before you can open the system-command input queue, your application program must be connected to your MQSeries subsystem. Use the MQI call **MQCONN** or **MQCONNX** to do this.

Then use the MQI call **MQOPEN** to open the system-command input queue. To use this call:

1. Set the *Options* parameter to **MQOO\_OUTPUT**
2. Set the MQOD object descriptor fields as follows:

*ObjectType*

MQOT\_Q (the object is a queue)

*ObjectName*

SYSTEM.COMMAND.INPUT

*ObjectQMgrName*

Leave blank if you want to send your request messages to your local queue manager. This means that your commands are processed locally.

If you want your MQSeries commands to be processed on a remote queue manager, put its name here. You must also have set up the correct queues and links, as described in the *MQSeries Intercommunication* manual.

## Opening a reply-to queue

To be able to retrieve the replies from an MQSeries command, you must open a reply-to queue. One way of doing this is to specify the model queue, **SYSTEM.COMMAND.REPLY.MODEL**, in an **MQOPEN** call to create a permanent dynamic queue as your reply-to queue. To use this call:

1. Set the *Options* parameter to **MQOO\_INPUT\_SHARED**
2. Set the MQOD object descriptor fields as follows:

*ObjectType*

MQOT\_Q (the object is a queue)

*ObjectName*

The name of your reply-to queue. If the queue name you specify is the name of a model queue object, the queue manager creates a dynamic queue.

*ObjectQMgrName*

To receive replies on your local queue manager, leave this field blank.

*DynamicQName*

Specify the name of the dynamic queue to be created.

### Using the command server

The command server is an MQSeries component that works with the command processor component. The command server reads request messages from the system-command input queue, verifies them, and passes the valid ones as commands to the command processor. The command processor processes the commands and puts any replies as reply messages on to the reply-to queue that you specify. The first reply message contains the user message CSQN205I. See “Interpreting the replies” on page 35 for more information.

### Identifying the queue manager that processes your commands

The queue manager that processes the commands you issue from an administration program is the queue manager that owns the system-command input queue that the message is put onto.

### Starting the command server

Normally, the command server is started automatically when the queue manager is started. It becomes available as soon as the message CSQ9022I ‘START QMGR’ NORMAL COMPLETION is returned from the START QMGR command. The command server is stopped when all the connected tasks have been disconnected during the system termination phase.

You can control the command server yourself using the START CMDSERV and STOP CMDSERV commands. To prevent the command server starting automatically when MQSeries is restarted, you can add a STOP CMDSERV command to your CSQINP1 or CSQINP2 initialization data sets.

The STOP CMDSERV command stops the command server as soon as it has finished processing the current message or immediately, if no messages are being processed.

If the command server has been stopped by a STOP CMDSERV command in the program, no other commands from the program can be processed. To restart the command server, you must issue a START CMDSERV command from the OS/390 console.

If you stop and restart the command server while MQSeries is running, all the messages that are on the system-command input queue when the command server stops are processed when the command server is restarted. However, if you stop and restart MQSeries after the command server is stopped, only the persistent messages on the system-command input queue are processed when the command server is restarted. All nonpersistent messages on the system-command input queue are lost.

### Sending commands to the command server

For each command, you build a message containing the command and then you put it onto the system-command input queue.

#### Building a message that includes MQSeries commands

You can incorporate MQSeries commands in an application program by building request messages that include the required commands. For each such command you:

1. Create a buffer containing a character string representing the command.

## Writing administration programs

2. Issue an **MQPUT** call specifying the buffer name in the *buffer* parameter of the call.

The simplest way to do this in C is to define a buffer using 'char'. For example:

```
char message_buffer[ ] = "ALTER QLOCAL(SALES) PUT(ENABLED)";
```

When you build a command, use a null-terminated character string. Do not specify a command prefix string (CPF) at the start of a command defined in this way. This means that you do not have to alter your command scripts if you want to run them on another queue manager. However, you must take into account that a CPF is included in any response messages that are put onto the reply-to queue.

The command server translates all characters to uppercase unless they are inside single quotes.

Commands can be any length up to a maximum 32 762 characters.

### Command attributes

When using commands in administration programs, you should consider the following:

1. Not all attributes have associated values.
2. Each attribute or attribute and value pair is separated by one or more blanks.
3. Do not make any assumptions about the order in which attributes are returned.
4. The attribute values returned are fixed length and surrounded by parentheses. Integer values are ten characters long, right justified, and padded with blanks. Character values are left justified and padded with blanks. Their lengths are as follows:
  - a. Character string lengths are the same as those given in the *MQSeries Application Programming Reference* manual.
  - b. Attributes that return a keyword (for example, **DEFSOPT** returns **EXCL** or **SHARED**) are 10 characters long, left justified, and padded with blanks.
  - c. Some attribute keywords can take negated values, for example, **NOTRIGGER**, **NOSHARE**, and **NOHARDENBO**. The attribute keywords that can have negated values take their length from the negated value. For example, the negated equivalent of **SHARE** is **NOSHARE**; it has a length of 7. These attributes are left justified and padded with blanks.
5. The number of attributes returned depends on what attributes are requested by the command.
6. The **NAMES** attribute of a namelist returns multiple values. This attribute returns a list of names, each of fixed length, separated by commas. Use the **NAMCOUNT** attribute to discover the number of names in the list. If there are no names in the list, the **NAMES** attribute is returned as **NAMES()**.
7. Attributes that normally require quotes around the string because they contain embedded blanks, lowercase characters or special characters, are returned without the quotes.
8. When you want to use the reply to a **DISPLAY** command as input to another command, put single quotes ( ' ) around each attribute. For example, if you define this queue:

```
+CSQ1 DEFINE QLOCAL(SALES) DESCR('Sales enquiries queue')
```

You can display it using the command:

```
+CSQ1 DISPLAY QUEUE(SALES) DESCR
```

## Writing administration programs

The DESCR attribute is displayed as:

```
DESCR(Sales enquiries queue)
```

To use this description in another command you must add the quotes as follows:

```
DESCR('Sales enquiries queue')
```

If the attribute itself contains any quotes, you must double them.

## Putting messages on the system-command input queue

Use the **MQPUT** call to put request messages containing commands on the system-command input queue. In this call you specify the name of the reply-to queue that you have already opened.

To use the **MQPUT** call:

1. Set these **MQPUT** parameters:

*Hconn* The connection handle returned by the **MQCONN** or **MQCONNEX** call.

*Hobj* The object handle returned by the **MQOPEN** call for the system-command input queue.

*BufferLength*

The length of the formatted command.

*Buffer* The name of the buffer containing the command.

2. Set these MQMD fields:

*MsgType*

MQMT\_REQUEST

*ReplyToQ*

Name of your reply-to queue.

*ReplyToQMgr*

Leave blank if you want replies sent to your local queue manager. If you want your MQSeries commands to be sent to a remote queue manager, put its name here. You must also have set up the correct queues and links, as described in the *MQSeries Intercommunication* manual.

3. Set any other MQMD fields, as required. If you are not using the same code page as the queue manager, set *CodedCharSetId* as appropriate, and set *Format* to MQFMT\_STRING, so that the command server can convert the message. You should normally use nonpersistent messages for commands.
4. Set any *PutMsgOpts* options, as required.

If you specify MQPMO\_SYNCPOINT (the default), you must follow the **MQPUT** call with a syncpoint call.

## Using MQPUT1 and the system-command input queue

If you want to put just one message on the system-command input queue, you can use the **MQPUT1** call. This call combines the functions of an **MQOPEN**, followed by an **MQPUT** of one message, followed by an **MQCLOSE**, all in one call. If you use this call, modify the parameters accordingly. See the *MQSeries Application Programming Guide* for details.



## Retrieving replies to your commands

When the command processor processes your commands, any reply messages are put onto the reply-to queue specified in the **MQPUT** call. The command server sends the reply messages with the same persistence as the command message it received.

### Waiting for a reply

Use the **MQGET** call to retrieve a reply from your request message. One request message can produce several reply messages. For details, see “Interpreting the replies” on page 35.

You can specify a time interval that an **MQGET** call waits for a reply message to be generated. If you do not get a reply, use the checklist beginning on page 43.

To use the **MQGET** call:

1. Set these parameters:

*Hconn* The connection handle returned by the **MQCONN** or **MQCONNX** call.

*Hobj* The object handle returned by the **MQOPEN** call for the reply-to queue.

*Buffer* The name of the area to receive the reply.

*BufferLength*

The length of the buffer to receive the reply. This must be a minimum of 80 bytes.

2. To ensure that you only get the responses from the command that you issued, you must specify the appropriate *MsgId* and *CorrelId* fields. These depend on the report options, **MQMD\_REPORT**, you specified in the **MQPUT** call:

**MQRO\_NONE**

Binary zero, '00...00' (24 nulls).

**MQRO\_NEW\_MSG\_ID**

Binary zero, '00...00' (24 nulls).

This is the default if none of these options has been specified.

**MQRO\_PASS\_MSG\_ID**

The *MsgId* from the **MQPUT**.

**MQRO\_NONE**

The *MsgId* from the **MQPUT** call.

**MQRO\_COPY\_MSG\_ID\_TO\_CORREL\_ID**

The *MsgId* from the **MQPUT** call.

This is the default if none of these options has been specified.

**MQRO\_PASS\_CORREL\_ID**

The *CorrelId* from the **MQPUT** call.

For more details on report options, see the *MQSeries Application Programming Reference* manual.

## Writing administration programs

3. Set the following *GetMsgOpts* fields:

### *Options*

MQGMO\_WAIT

If you are not using the same code page as the queue manager, set MQGMO\_CONVERT, and set *CodedCharSetId* as appropriate in the MQMD.

### *WaitInterval*

For replies from the local queue manager, try 5 seconds. Coded in milliseconds, this becomes 5 000. For replies from a remote queue manager, and channel control and status commands, try 30 seconds. Coded in milliseconds, this becomes 30 000.

## Discarded messages

If the command server finds that a request message is not valid, it discards this message and writes the message CSQN205I to the named reply-to queue. If there is no reply-to queue, the CSQN205I message is put onto the dead-letter queue. The return code in this message shows why the original request message was not valid:

00D5020F	It is not of type MQMT_REQUEST.
00D50210	It has zero length.
00D50212	It is longer than 32 762 bytes.
00D50211	It contains all blanks.
00D5483E	It needed converting, but <i>Format</i> was not MQFMT_STRING.
Other	See the <i>MQSeries for OS/390 Messages and Codes</i> manual.

## The reply message descriptor

For any reply message, the following MQMD message descriptor fields are set:

<i>MsgType</i>	MQMT_REPLY
<i>Feedback</i>	MQFB_NONE
<i>Encoding</i>	MQENC_NATIVE
<i>Priority</i>	As for the MQMD in the message you issued.
<i>Persistence</i>	As for the MQMD in the message you issued.
<i>CorrelId</i>	Depends on the MQPUT report options.
<i>ReplyToQ</i>	None.

The command server sets the *Options* field of the MQPMO structure to MQPMO\_NO\_SYNCPOINT. This means that you can retrieve the replies as they are created, rather than as a group at the next syncpoint.

## End of General-use programming interface information

## Interpreting the replies

### Start of Product-sensitive programming interface information

Each request message correctly processed by MQSeries produces at least two reply messages. Each reply message contains a single MQSeries user message.

The length of a reply depends on the command that was issued. The longest reply you can get is from a DISPLAY NAMELIST, and that can be up to 13 000 bytes long.

The first user message, CSQN205I, always contains:

- A count of the replies (in decimal), which you can use as a counter in a loop to get the rest of the replies. The count includes this first message.
- The return code from the command preprocessor.
- A reason code, which is the return code from the command processor.

This message does not contain a CPF.

For example:

```
CSQN205I  COUNT=      4, RETURN=0000000C, REASON=00000008
```

The COUNT field is 8 bytes long and is right-justified. It always starts at position 18, that is, immediately after 'COUNT='. The RETURN field is 8 bytes long in character hexadecimal and is immediately after 'RETURN=' at position 35. The REASON field is 8 bytes long in character hexadecimal and is immediately after 'REASON=' at position 52.

If the RETURN= value is 00000000 and the REASON= value is 00000004, the set of reply messages is incomplete. After retrieving the replies indicated by the CSQN205I message, issue a further **MQGET** call to wait for a further set of replies. The first message in the next set of replies will again be CSQN205I, indicating how many replies there are, and whether there are still more to come.

See the *MQSeries for OS/390 Messages and Codes* manual for more details about the individual messages.

If you are using a non-English language feature, the text and layout of the replies are different from those shown here. However, the size and position of the count and return codes in message CSQN205I are the same.

### Using the DISPLAY commands

To obtain information about MQSeries, use the MQSeries DISPLAY commands. Use these commands rather than **MQINQ** if you want:

- Information about objects on a remote queue manager. (**MQINQ** only returns information from the local queue manager.)
- Reply messages ready-formatted for printing. (**MQINQ** returns information that is not formatted.)
- Other information that **MQINQ** does not provide.

The format of the replies from these commands:

```
DISPLAY CMDSERV
DISPLAY DQM
DISPLAY GROUP
DISPLAY LOG
DISPLAY MAXSMGS
DISPLAY SECURITY
DISPLAY THREAD
DISPLAY TRACE
DISPLAY USAGE
```

is the same, regardless of whether you issue the command from an application program or from an OS/390 console. However, if you issue DISPLAY commands listed below for MQSeries objects or object status, the format is different when they are issued from an application program.

```
DISPLAY CHANNEL/CHSTATUS
DISPLAY CLUSQMGR
DISPLAY NAMELIST
DISPLAY PROCESS
DISPLAY QMGR
DISPLAY QUEUE/QSTATUS
DISPLAY STGCLASS
```

The user messages in the replies are still in the form of character strings, however, the attribute values in a message have the fixed positions relative to the attribute name.

The format of the reply is:

```
msg_no +CSQ1 attr_name(value) attr_name attr_name(value)
```

where:

msg_no	An 8 character message number
+CSQ1	The command prefix string
attr_name	The attribute or keyword name
value	The attribute value

## Examples of commands and their replies

Here are some examples of commands that could be built into MQSeries messages, and the user messages that are the replies. Unless otherwise stated, each line of the reply is a separate message.

### Messages from a DEFINE command

The following command:

```
DEFINE QLOCAL(Q1)
```

Produces these messages:

```
CSQN205I  COUNT=      2, RETURN=00000000, REASON=00000000
CSQ9022I +CSQ1 CSQMMSGP ' DEFINE QLOCAL' NORMAL COMPLETION
```

These reply messages are produced on normal completion.

### Messages from a DELETE command

The following command:

```
DELETE QLOCAL(Q2)
```

Produces these messages:

```
CSQN205I  COUNT=      4, RETURN=00000000, REASON=00000008
CSQM125I +CSQ1 CSQMUQLC QLOCAL (Q2) QSGDISP(QMGR) WAS NOT FOUND
CSQM090E +CSQ1 CSQMUQLC FAILURE REASON CODE X'00D44002'
CSQ9023E +CSQ1 CSQMUQLC ' DELETE QLOCAL' ABNORMAL COMPLETION
```

These messages indicate that a local queue called Q2 does not exist.

### Messages from DISPLAY commands

The following examples show the replies from some DISPLAY commands.

#### Finding out the name of the dead-letter queue

If you want to find out the name of the dead-letter queue for a queue manager, issue this command from an application program:

```
DISPLAY QMGR DEADQ
```

The following three user messages are returned, from which you can extract the required name:

```
CSQN205I  COUNT=      3, RETURN=00000000, REASON=00000000
CSQM409I +CSQ1 QMNAME(CSQ1) DEADQ(SYSTEM.DEAD.QUEUE      )
CSQ9022I +CSQ1 CSQMDRTS ' DISPLAY QMGR' NORMAL COMPLETION
```

## Writing administration programs

### Messages from the DISPLAY THREAD command

The following command:

```
DISPLAY THREAD(*) TYPE(*)
```

Produces these messages:

```
CSQN205I  COUNT=      20, RETURN=00000000, REASON=00000000.
CSQV401I +CSQ1 DISPLAY THREAD REPORT FOLLOWS -
CSQV402I +CSQ1 ACTIVE THREADS - 668
NAME      STA      REQ  THREAD-XREF          USERID  ASID  URID
ABCDEF G  T         3                               ABCDEF G  002D  000000000000
MQSCIC1  T         4                               MQSCIC1  0034  000000000000
MQSCIC1  T         0 00011128C3D2C1D40000023C MQSCIC1  0034  000000000000
MQSCIC1  T         3                               MQSCIC1  0034  000000000000
MQSCIC1  T         1                               0034  000000000000
MQSCIC1  T         1                               0034  000000000000
MQSCIC1  T         1                               0034  000000000000
MQSCIC1  T         1                               0034  000000000000
MQSCIC1  T         1                               0034  000000000000
MQSCIC1  T         1                               0034  000000000000
MQSCIC1  T         1                               0034  000000000000
MQSCIC1  T         0 00012020C3D2E3C90000039C MQSCIC1  0034  000000000000
DISPLAY ACTIVE REPORT COMPLETE
CSQV412I +CSQ1 CSQVDT NO INDOUBT THREADS FOUND FOR NAME=MQSCIC1
CSQV412I +CSQ1 CSQVDT NO INDOUBT THREADS FOUND FOR NAME=ABCDEF G
CSQ9022I +CSQ1 CSQVDT ' DISPLAY THREAD ' NORMAL COMPLETION
```

The actual number and content of the messages depend on what is running in your queue manager.

### Messages from the DISPLAY QUEUE command

The following examples show how the results from a command depend on the attributes specified in that command.

**Example 1:** You define a local queue using the command:

```
DEFINE QLOCAL(Q1) DESCR('A sample queue') GET(ENABLED) SHARE
```

If you issue the following command from an application program:

```
DISPLAY QUEUE(Q1) SHARE GET DESCR
```

These three user messages are returned:

```
CSQN205I  COUNT=      3, RETURN=00000000, REASON=00000000
CSQM401I +CSQ1 QUEUE(Q1
QLOCAL ) QSGDISP(QMGR ) ) TYPE(
DESCR(A sample queue
) SHARE GET(ENABLED )
CSQ9022I +CSQ1 CSQMMSG ' DISPLAY QUEUE ' NORMAL COMPLETION
```

**Note:** The second message, CSQM401I, is shown here occupying four lines.

## Writing administration programs

**Example 2:** Two queues have names beginning with the letter “A”:

A1 is a local queue with its PUT attribute set to DISABLED.

A2 is a remote queue with its PUT attribute set to ENABLED.

If you issue the following command from an application program:

```
DISPLAY QUEUE(A*) PUT
```

These four user messages are returned:

```
CSQN205I  COUNT=      4, RETURN=00000000, REASON=00000000
CSQM401I +CSQ1 QUEUE(A1                                ) TYPE(
QLOCAL  ) QSGDISP(QMGR      )
          PUT(DISABLED  )
CSQM406I +CSQ1 QUEUE(A2                                ) TYPE(
QREMOTE ) PUT(ENABLED   )
CSQ9022I +CSQ1 CSQMDMSG ' DISPLAY QUEUE' NORMAL COMPLETION
```

**Note:** The second and third messages, CSQM401I and CSQM406I, are shown here occupying three and two lines respectively.

### Messages from the DISPLAY NAMELIST command

A namelist is defined by the command:

```
DEFINE NAMELIST(N1) NAMES(Q1,SAMPLE_QUEUE)
```

If you issue the following command from an application program:

```
DISPLAY NAMELIST(N1) NAMES NAMCOUNT
```

The following three user messages are returned:

```
CSQN205I  COUNT=      3, RETURN=00000000, REASON=00000000
CSQM407I +CSQ1 NAMELIST(N1                                ) QS
GDISP(QMGR      ) NAMCOUNT(      2) NAMES(Q1
,SAMPLE_QUEUE
)
CSQ9022I +CSQ1 CSQMDMSG ' DISPLAY NAMELIST' NORMAL COMPLETION
```

**Note:** The third message, CSQM407I, is shown here occupying three lines.

## Writing administration programs

### Messages from commands with CMDSCOPE

The following examples show the replies from commands that have been entered with the CMDSCOPE attribute.

#### Messages from the ALTER PROCESS command

The following command:

```
ALT PRO(V4) CMDSCOPE(*)
```

Produces the following messages:

```
CSQN205I  COUNT=      2, RETURN=00000000, REASON=00000004
CSQN137I !MQ25 'ALT PRO' command accepted for CMDSCOPE(*), sent to 2
CSQN205I  COUNT=      5, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'ALT PRO' command responses from MQ26
CSQM125I !MQ26 CSQMMSGP PROCESS(V4) QSGDISP(QMGR) WAS NOT FOUND
CSQM090E !MQ26 CSQMMSGP FAILURE REASON CODE X'00D44002'
CSQ9023E !MQ26 CSQMMSGP ' ALT PRO' ABNORMAL COMPLETION
CSQN205I  COUNT=      3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'ALT PRO' command responses from MQ25
CSQ9022I !MQ25 CSQMMSGP ' ALT PRO' NORMAL COMPLETION
CSQN205I  COUNT=      2, RETURN=0000000C, REASON=00000008
CSQN123E !MQ25 'ALT PRO' command for CMDSCOPE(*) abnormal completion
```

The command was entered on queue manager MQ25 and sent to two queue managers (MQ25 and MQ26). The command was successful on MQ25 but the process definition did not exist on MQ26, so the command failed on that queue manager.

#### Messages from the DISPLAY PROCESS command

The following command:

```
DIS PRO(V*) CMDSCOPE(*)
```

Produces the following messages:

```
CSQN205I  COUNT=      2, RETURN=00000000, REASON=00000004
CSQN137I !MQ25 'DIS PRO' command accepted for CMDSCOPE(*), sent to 2
CSQN205I  COUNT=      5, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DIS PRO' command responses from MQ26
CSQM408I !MQ26 PROCESS(V2) QSGDISP(COPY)
CSQM408I !MQ26 PROCESS(V3) QSGDISP(QMGR)
CSQ9022I !MQ26 CSQMDRTS ' DIS PROCESS' NORMAL COMPLETION
CSQN205I  COUNT=      7, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DIS PRO' command responses from MQ25
CSQM408I !MQ25 PROCESS(V2) QSGDISP(COPY)
CSQM408I !MQ25 PROCESS(V2) QSGDISP(GROUP)
CSQM408I !MQ25 PROCESS(V3) QSGDISP(QMGR)
CSQM408I !MQ25 PROCESS(V4) QSGDISP(QMGR)
CSQ9022I !MQ25 CSQMDRTS ' DIS PROCESS' NORMAL COMPLETION
CSQN205I  COUNT=      2, RETURN=00000000, REASON=00000000
CSQN122I !MQ25 'DIS PRO' command for CMDSCOPE(*) normal completion
```

The command was entered on queue manager MQ25 and sent to two queue managers (MQ25 and MQ26). Information is displayed about all the processes on each queue manager with names starting with the letter V.



**Messages from the DISPLAY CHSTATUS command**

The following command:

```
DIS CHS(VT) CMDSCOPE(*)
```

Produces the following messages:

```
CSQN205I  COUNT=      2, RETURN=00000000, REASON=00000004
CSQN137I !MQ25 'DIS CHS' command accepted for CMDSCOPE(*), sent to 2
CSQN205I  COUNT=      3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DIS CHS' command responses from MQ25
CSQM134I !MQ25 CSQMDCST DIS CHS(VT) COMMAND ACCEPTED
CSQN205I  COUNT=      3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DIS CHS' command responses from MQ26
CSQM134I !MQ26 CSQMDCST DIS CHS(VT) COMMAND ACCEPTED
CSQN205I  COUNT=      4, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DIS CHS' command responses from MQ25
CSQM422I !MQ25 CHSTATUS(VT) CHLDISP(PRIVATE) CONNAME( ) CURRENT STATUS(STOPPED)
CSQ9022I !MQ25 CSQXDRTS ' DIS CHS' NORMAL COMPLETION
CSQN205I  COUNT=      4, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DIS CHS' command responses from MQ26
CSQM422I !MQ26 CHSTATUS(VT) CHLDISP(PRIVATE) CONNAME( ) CURRENT STATUS(STOPPED)
CSQ9022I !MQ26 CSQXDRTS ' DIS CHS' NORMAL COMPLETION
CSQN205I  COUNT=      2, RETURN=00000000, REASON=00000000
CSQN122I !MQ25 'DIS CHS' command for CMDSCOPE(*) normal completion
```

The command was entered on queue manager MQ25 and sent to two queue managers (MQ25 and MQ26). Information is displayed about channel status on each queue manager.

**Messages from the STOP CHANNEL command**

The following command:

```
STOP CHL(VT) CMDSCOPE(*)
```

Produces these messages:

```
CSQN205I  COUNT=      2, RETURN=00000000, REASON=00000004
CSQN137I !MQ25 'STOP CHL' command accepted for CMDSCOPE(*), sent to 2
CSQN205I  COUNT=      3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'STOP CHL' command responses from MQ25
CSQM134I !MQ25 CSQMTCHL STOP CHL(VT) COMMAND ACCEPTED
CSQN205I  COUNT=      3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'STOP CHL' command responses from MQ26
CSQM134I !MQ26 CSQMTCHL STOP CHL(VT) COMMAND ACCEPTED
CSQN205I  COUNT=      3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'STOP CHL' command responses from MQ26
CSQ9022I !MQ26 CSQXCRPS ' STOP CHL' NORMAL COMPLETION
CSQN205I  COUNT=      3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'STOP CHL' command responses from MQ25
CSQ9022I !MQ25 CSQXCRPS ' STOP CHL' NORMAL COMPLETION
CSQN205I  COUNT=      2, RETURN=00000000, REASON=00000000
CSQN122I !MQ25 'STOP CHL' command for CMDSCOPE(*) normal completion
```

The command was entered on queue manager MQ25 and sent to two queue managers (MQ25 and MQ26). Channel VT was stopped on each queue manager.

## Writing administration programs

### Messages from commands that generate commands with CMDSCOPE

The following command:

```
DEF PRO(V2) QSGDISP(GROUP)
```

Produces these messages:

```
CSQN205I  COUNT=      3, RETURN=00000000, REASON=00000004
CSQM122I !MQ25 CSQMMSGP ' DEF PRO' COMPLETED FOR QSGDISP(GROUP)
CSQN138I !MQ25 'DEFINE PRO' command generated for CMDSCOPE(*), sent to 2
CSQN205I  COUNT=      3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DEFINE PRO' command responses from MQ25
CSQ9022I !MQ25 CSQMMSGP ' DEFINE PROCESS' NORMAL COMPLETION
CSQN205I  COUNT=      3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DEFINE PRO' command responses from MQ26
CSQ9022I !MQ26 CSQMMSGP ' DEFINE PROCESS' NORMAL COMPLETION
CSQN205I  COUNT=      2, RETURN=00000000, REASON=00000000
CSQN122I !MQ25 'DEFINE PRO' command for CMDSCOPE(*) normal completion
```

The command was entered on queue manager MQ25. When the object was created on the shared repository, another command was generated and sent to all the active queue managers in the queue-sharing group (MQ25 and MQ26).

**End of Product-sensitive programming interface information**

---

## If you do not receive a reply

If you do not receive a reply to your request message, work through this checklist:

- Is the command server running?
- Is the *WaitInterval* long enough?
- Are the system-command input and reply-to queues correctly defined?
- Were the **MQOPEN** calls to these queues successful?
- Are both the system-command input and reply-to queues enabled for **MQPUT** and **MQGET** calls?
- Have you considered increasing the **MAXDEPTH** and **MAXMSGL** attributes of your queues?
- Are you are using the *CorrelId* and *MsgId* fields correctly?
- Is the MQSeries subsystem still running?
- Was the command built correctly?
- Are all your remote links defined and operating correctly?
- Were the **MQPUT** calls correctly defined?
- Has the reply-to queue been defined as a temporary dynamic queue instead of a permanent dynamic queue? (If the request message is persistent, you must use a permanent dynamic queue for the reply.)

When the command server generates replies but cannot write them to the reply-to queue that you specify, it tries to write them to the system dead-letter queue.

---

## Passing commands using MGCR or MGCRE

If you have the correct authorization, you can pass MQSeries commands from your program to multiple MQSeries subsystems by the MGCR or MGCRE (SVC 34) OS/390 service. The value of the CPF identifies the particular MQSeries subsystem to which the command is directed. For information about CPFs, see the *MQSeries for OS/390 System Setup Guide*.

If you use MGCRE, you can use a Command and Response Token (CART) to get the direct responses to the command.



---

## Part 2. MQSeries and CICS

<b>Chapter 4. Operating the CICS adapter</b> . . . . .	47
Invoking the adapter's control functions. . . . .	47
From the CICS adapter control panels . . . . .	47
From the CICS command line . . . . .	47
From CICS application programs . . . . .	48
Command syntax in application programs . . . . .	48
Passing parameters from a CICS transaction	48
EXEC CICS LINK interface messages. . . . .	49
Accessing the CICS adapter control panels . . . . .	50
Starting a connection . . . . .	51
Starting a connection from the CICS adapter	
control panels . . . . .	51
Starting a connection from the CICS command	
line . . . . .	52
Specifying lowercase queue names . . . . .	52
Starting a connection from a CICS application	
program . . . . .	53
Stopping a connection. . . . .	54
Stopping a connection from the CICS adapter	
control panels . . . . .	54
Stopping a connection from the CICS command	
line . . . . .	55
Stopping a connection from a CICS application	
program . . . . .	55
Modifying a connection . . . . .	56
Modifying a connection from the CICS adapter	
control panels . . . . .	56
Modifying a connection from the CICS command	
line . . . . .	57
Modifying a connection from a CICS application	
program . . . . .	58
Displaying details of connections and CICS tasks. . . . .	59
Displaying details of a connection from the CICS	
adapter control panels. . . . .	59
Starting an instance of the task initiator CKTI . . . . .	60
Starting CKTI from the CICS adapter control	
panels . . . . .	60
Starting CKTI from the CICS command line . . . . .	61
Starting CKTI from a CICS application program . . . . .	61
Starting CKTI automatically . . . . .	61
Stopping an instance of CKTI . . . . .	62
Stopping an instance of CKTI from the CICS	
adapter control panels. . . . .	62
Stopping an instance of CKTI from the command	
line . . . . .	63
Stopping an instance of CKTI from an	
application program . . . . .	63
Displaying the current instances of CKTI . . . . .	64
Displaying the current instances of CKTI from	
the CICS adapter control panels . . . . .	64
Displaying CICS task information . . . . .	65
Displaying CICS tasks from the CICS adapter	
control panels . . . . .	65
Displaying connection status and in-flight tasks	
From the CICS command line . . . . .	66
From a CICS application program . . . . .	66

Purging tasks that are using the CICS adapter. . . . .	67
Shutting down a connection between MQSeries and	
the CICS adapter . . . . .	68
Orderly shutdown . . . . .	68
Forced shutdown . . . . .	69

<b>Chapter 5. Operating the CICS bridge.</b> . . . . .	71
Starting the CICS bridge . . . . .	71
Shutting down the CICS bridge. . . . .	72
Controlling CICS-bridge throughput . . . . .	72



---

## Chapter 4. Operating the CICS adapter

This chapter describes how you can use the CICS adapter control functions to initiate and manage connections between MQSeries and CICS. It describes these tasks:

- “Invoking the adapter’s control functions”
- “Accessing the CICS adapter control panels” on page 50
- “Starting a connection” on page 51
- “Stopping a connection” on page 54
- “Modifying a connection” on page 56
- “Displaying details of connections and CICS tasks” on page 59
- “Starting an instance of the task initiator CKTI” on page 60
- “Stopping an instance of CKTI” on page 62
- “Displaying the current instances of CKTI” on page 64
- “Displaying CICS task information” on page 65
- “Purging tasks that are using the CICS adapter” on page 67
- “Shutting down a connection between MQSeries and the CICS adapter” on page 68

Before you can use the CICS adapter for messaging, you must start the MQSeries subsystem.

---

### Invoking the adapter’s control functions

You can invoke the control functions of the CICS adapter in three different ways:

1. From the CICS adapter control panels.
2. From the CICS command line.
3. From an application program.

#### From the CICS adapter control panels

You can use the CICS adapter control panels to monitor and control connections between MQSeries and CICS.

From the initial panel, you first select an item from the menu bar at the top of the panel, and then select an action from one of the pull-down menus. In the displayed panel or secondary window, you can then type new values in the fields, as required.

#### From the CICS command line

You can take a “fast-path” approach and bypass the CICS adapter control panels, by specifying command line parameters on the CKQC transaction. The syntax of these command parameters, and examples of them, are given for each of the tasks described later in this chapter.

**Note:** You can also issue these commands from the console using OS/390 commands. Commands take this form:

```
MODIFY CICS-job-name CKQC command-line-command
```

## Operating the CICS adapter

### From CICS application programs

You can use the EXEC CICS LINK command to invoke most adapter control functions from CICS application programs. The syntax of the EXEC CICS LINK commands you need, and examples, are given for each of the tasks described later in this chapter.

#### Command syntax in application programs

Some commands issued in this way must be padded with trailing spaces to make the length of the command 10 characters. When an argument follows the command, an extra space character must be added as a separator. See Figure 12. The commands affected by this restriction and the number of trailing spaces required for each command are:

Command	Number of trailing spaces (not including the separator)
START	5
MODIFY	4
STARTCKTI	1
STOPCKT	1

With all other commands the padding is optional.

```
EXEC CICS LINK PROGRAM('CSQCRST ')
      INPUTMSG('CKQC MODIFY      Y')
                ↑                ↑
                .....
                1                12
```

Figure 12. *Padding adapter commands.* The MODIFY command must be padded with 4 trailing spaces plus another space as a separator. Starting at the 'M' in MODIFY, the argument 'Y' is the twelfth character.

**Note:** This restriction applies only to commands issued from an application program; it does not apply to commands issued from the command line.

#### Passing parameters from a CICS transaction

Use the following rules to determine how to pass the parameters:

- The CICS transaction must be running on an attached terminal. If it is not, all MQSeries commands are ignored.
- If a CICS application program on an attached terminal is connected to MQSeries, you must use the INPUTMSG option with EXEC CICS LINK to pass parameters, except at PLTPI time.
- If you connect to MQSeries at PLTPI time, you must use the COMMAREA option to pass parameters. If you use the INPUTMSG option, the command is ignored.

However, the adapter STOP commands:

```
CKQC STOP
CKQC STOP FORCE
```

cannot be run at PLTPI time, regardless of whether you use the INPUTMSG option or the COMMAREA option.



### **EXEC CICS LINK interface messages**

If you invoke the adapter operation functions `START` and `STOP` from an application program using `EXEC CICS LINK`, the resultant messages are written to both the system console and a transient data queue (TDQ) named `CKQQ`. When the application program returns from the `LINK`, it can read back the messages by repeating `EXEC CICS READQ TD QUEUE(CKQQ)` until the queue is empty. The following restrictions apply:

- The TDQ queue name is `CKQQ` and cannot be changed. A sample TDQ definition is provided (in `CSQ4DCT2`), which defines `CKQQ` as an intra-partition TDQ.
- The queue is not cleared before it is written to.
- The messages are not time-stamped.
- If you have more than one application writing to the TDQ, the messages are not serialized. It is the responsibility of the invoking programs to serialize themselves.
- The same set of messages also appear on the system console.
- The server subtask messages are not written to `CKQQ`.

## Accessing the CICS adapter control panels

To access the adapter control panels, use the CICS transaction CKQC:

1. Type CKQC and press Enter.  
The CICS adapter control initial panel, shown in Figure 13, is displayed.
2. In the menu bar at the top of the screen, use the TAB key to move between the three options **Connection**, **CKTI**, and **Task**.
3. Press Enter to select your choice.
4. Select the required option from one of the pull-down menus by typing the number of your choice and then pressing Enter to confirm or function key F12 to cancel.
5. Press function key F1 to get help on any panel or window.

```

      Connection      CKTI      Task
-----
CKQCM0  IBM MQSeries for OS/390 - CICS adapter control initial panel

Select menu bar item using Tab key. Then press Enter.

#          #  #####  #####
##         ## ###   ## ##   ##
###        ### ##   ## ##   ##
####       #### ##   ## #####  #####  ## #####  #####  #####
## ##    ## ## ##   ##   #####  #   #  #####  ## #   #   #
## ## ##  ## ##   ## ##   ## #####  ##  ## #####  #####
##   ##  ## ##   ## ##   ## #   ##   ##  ## #   ##   #
##  #   ##  #####  ## #####  #####  ##   ##  #####  #####

                                                    for OS/390

(C) Copyright IBM Corporation 1993, 2000. All rights reserved.

F1=Help  F3=Exit
    
```

Figure 13. The CICS adapter control initial panel

**Note:** You can access the adapter control panels *without* starting the MQSeries subsystem. You can also start a connection but it will not be active until MQSeries is started.

## Starting a connection

You can start a connection from:

- The CICS adapter control panels
- The CICS command line
- A CICS application program
- A PLTPI program
- The CICS MQCONN SIT parameter

### Starting a connection from the CICS adapter control panels

To start a connection from the CICS adapter control initial panel:

1. Select **Connection** from the menu bar.
2. Select the **Start** action from the pull-down menu. See Figure 14.
3. Modify the connection values displayed in the **Start a Connection** secondary parameter window. Alternatively, use the defaults derived from the INITPARM settings, if defined.
4. Press Enter to confirm.

Messages indicating the success or failure of the attempt to start the connection are displayed on the CICS adapter messages panel, CKQCM1.

```

Connection      CKTI      Task
+-----+-----+-----+
| Select an action. | for OS/390 - CICS adapter control initial panel |
| 1. Start...      | sing Tab key. Then press Enter.                  |
| 2. Stop...       |                                                     |
| 3. Modify...     |                                                     |
| 4. Display       |                                                     |
+-----+-----+-----+
| F1=Help F12=Cancel |                                                     |
+-----+-----+-----+
## ## ## ## ## #
## ## ## ## ## #
## ### ## ### #
## # ## ##### #
+-----+-----+-----+
|                                     | Start a Connection |
|                                     | Type parameters. Then press Enter. |
|                                     | 1. Queue Manager Name (SN) . . . QMGR      ##
|                                     | 2. Initiation Queue Name (IQ) . . . . . ##
|                                     | CICS.INITIATION.QUEUE1                    #
|                                     | 3. Trace Number (TN) . . . . . 123        #
|                                     | F1=Help F12=Cancel                        |0
+-----+-----+-----+

(C) Copyright IBM Corporation 1993, 2000. All rights reserved.

F1=Help F3=Exit

```

Figure 14. Starting a connection

## Operating the CICS adapter

### Starting a connection from the CICS command line

The example shown in Figure 15 starts a connection, using the default connection values set at system initialization.

```
CKQC START
```

Figure 15. Starting a connection from the command line

The command shown in Figure 16 starts a connection, using the explicitly defined connection parameter values. The parameters are positional—every field must be entered to its maximum length if you want to override the default.

```
CKQC START Y|N <subsystem ID> <trace number> <initiation queue name>
```

Figure 16. Starting a connection from the command line specifying parameters

Where:

Y|N Specify either:

'Y' Use the default values, that is, substitute default values for any blank arguments.

'N' Do not use the default values.

<subsystem ID>

Name of the queue manager to connect to.

<trace number>

The trace number. It must be in the range 0 through 199.

<initiation queue name>

The name of the default initiation queue.

### Specifying lowercase queue names

By default, CICS translates lowercase input, for both keywords and parameters, to uppercase. Therefore, by default, these commands are equivalent:

```
CKQC START Y CSQ1 199 CICS01.INITQ  
ckqc start y csq1 199 cics01.initq
```

Figure 17. Specifying lowercase queue names

If you want to use lowercase queue names, you must:

1. Specify UCTRAN(TRANID) on the TYPETERM definition of terminals that start adapter control functions.
2. Specify UCTRAN(NO) on the transaction profile used by all “CKxx” transactions.

Thereafter, the adapter translates all lowercase arguments, *except queue names*, to uppercase.

For details of TYPETERM and PROFILE definitions, see the *CICS Resource Definition Guide*.

## Starting a connection from a CICS application program

You can start a connection by linking the adapter connect program, CSQCQCON, from a CICS application program. Your program, which can be written in C, COBOL, PL/I, or assembler language, must pass a parameter list that specifies the connection values to be used. The parameter list is:

### CKQC

4-character transaction ID—must be 'CKQC'.

### DISPMODE

1-byte field—must contain a blank.

### CONNREQ

10-character field—must contain 'START '.

### DELIM1

1-byte delimiter field—must contain a blank.

**INITP** 1-character field that specifies whether this connection is to use the default parameters set by INITPARM. The possible values are:

'Y' Use the default values, that is, substitute default values for any blank arguments.

'N' Do not use the default values. If you code 'N', you must supply all the new connection values, to override the INITPARM settings, in the CONNSSN, CONNTN, and CONNIQ fields.

' ' Equivalent to 'Y'.

### DELIM2

1-byte delimiter field—must contain a blank.

### CONNSSN

4-character field used to specify the OS/390 subsystem name of the target queue manager.

This must be the name of a queue manager, not a queue-sharing group.

### DELIM3

1-byte delimiter field—must contain a blank.

### CONNTN

3-character trace number. If supplied, it must be in the range 0 through 199.

### DELIM4

1-byte delimiter field—must contain a blank.

### CONNIQ

48-character field that specifies the name of the default initiation queue.

Figure 18 shows the LINK command that your CICS program must issue.

```
EXEC CICS LINK PROGRAM('CSQCQCON')
        INPUTMSG(CONNPL) INPUTMSGLEN(length of CONNPL)
```

Figure 18. Linking to the adapter connect program, CSQCQCON, from a CICS program. In this example, the name of the parameter list is CONNPL.

Output messages from CSQCQCON are displayed on the system console.

## Stopping a connection

You can stop a connection from:

- The CICS adapter control panels
- The CICS command line
- A CICS application program

### Stopping a connection from the CICS adapter control panels

From the initial panel:

1. Select **Connection** from the menu bar.
2. Select the **Stop** action from the pull-down menu.
3. Use the **Stop Connection** secondary parameter window to select the type of shutdown that you require. Methods of shutting down the CICS adapter are summarized in Table 2 on page 68.

```

Connection      CKTI      Task
-----+-----+-----
Select an action.  for OS/390 - CICS adapter control initial panel
2 1. Start...    sing Tab key. Then press Enter
  2. Stop...
  3. Modify...
  4. Display
-----+-----+-----
| F1=Help F12=Cancel |
-----+-----+-----
## ## ## ## ##
## ## ## ## ## #
## ## ## ## ##
## # ## #####
-----+-----+-----
| Stop Connection |
| Select stop type. |
| Then press Enter |
| | | | | | | |
| 1 1. Quiesce |
| 2. Force |
| | | | | | | |
| F1=Help F12=Cancel |
-----+-----+-----
for OS/390

(C) Copyright IBM Corporation 1993, 2000. All rights reserved.

F1=Help F3=Exit
    
```

Figure 19. Stopping a connection from the CKQC initial panel

The messages associated with stopping a connection are displayed on the system console.

## Stopping a connection from the CICS command line

The command shown in Figure 20 initiates a *quiesced* shutdown. The connection shuts down only after the last task has completed its work.

```
CKQC STOP
```

Figure 20. Stopping a connection from the command line—a quiesced shutdown

The command shown in Figure 21 initiates a *forced* shutdown. The connection shuts down immediately, regardless of the state of any in-flight tasks.

```
CKQC STOP FORCE
```

Figure 21. Stopping a connection from the command line—a forced shutdown

## Stopping a connection from a CICS application program

To stop a connection from a CICS program, the program must link to the adapter shutdown program, CSQCDSC. Figures 22 and 23 show examples of LINK commands initiating quiesced and forced shutdowns. When you do an EXEC CICS LINK to CSQCDSC, the program requires a terminal associated task.

```
EXEC CICS LINK PROGRAM('CSQCDSC ')
        INPUTMSG('CKQC STOP')
```

Figure 22. Stopping a connection from a CICS application program—a quiesced shutdown. The QUIESCE parameter is optional.

```
EXEC CICS LINK PROGRAM('CSQCDSC ')
        INPUTMSG('CKQC STOP FORCE')
```

Figure 23. Stopping a connection from a CICS application program—a forced shutdown

Output messages from CSQCDSC are displayed on the system console.

## Modifying a connection

You can modify a connection to reset the connection statistics, enable or disable the API-crossing exit, or change the adapter's trace number. You can do this from:

- The CICS adapter control panels
- The CICS command line
- A CICS application program

### Modifying a connection from the CICS adapter control panels

From the initial panel:

1. Select **Connection** from the menu bar.
2. Select the **Modify** action from the pull-down menu.
3. Use the **Modification Options** secondary parameter window to specify the option you require.

To change the trace number:

- Enter 4 in the options selection field
  - Enter a number, in the range 0 through 199, in the trace number field. Do not change the 4 in the options selection field before you press Enter. If you do, the trace number will not be changed.
4. Press Enter to confirm your choice.
  5. Repeat steps 1 through 4, as required.

```

Connection      CKTI      Task
+-----+-----+-----+
| Select an action. | for OS/390 - CICS adapter control initial panel
|                 |
| 3 1. Start...   | sing Tab key. Then press Enter.
| 2. Stop...     |
| 3. Modify...   |
| 4. Display     |
+-----+-----+-----+
| F1=Help F12=Cancel |
+-----+-----+-----+
## ## ## ## ##
## ## ## ## ## #
## ### ## ###
## # ## #####
|                 |
| Modification Options
| Select modify option. Then
| press Enter.    | ##
|                 | ## ##### #####
| 4 1. Reset statistics
| 2. Enable API Exit
| 3. Disable API Exit
| 4. Change Trace Number 123
|                 | ## ##### #####
| F1=Help F12=Cancel | for OS/390
+-----+-----+-----+

(C) Copyright IBM Corporation 1993, 2000. All rights reserved.

F1=Help F3=Exit
    
```

Figure 24. Modifying a connection



## Modifying a connection from the CICS command line

You can use the CKQC MODIFY command to modify a connection.

```
CKQC MODIFY Y|N E|D <trace-number>
```

Figure 25. Format of command to modify connection parameters from the command line

The command syntax is shown in Figure 25, where:

**Y|N** Specify one of:  
**Y** Reset connection statistics.  
**N** Do not reset connection statistics.

This parameter is required.

**E|D** Specify one of:  
**E** Enable the API-crossing exit.  
**D** Disable the API-crossing exit.

This parameter is optional, the default is to disable the API-crossing exit.

**<trace number>**

Specify a valid trace number in the range 0 through 199. This parameter is optional. If it is not specified, the trace number is not changed.

The command shown in Figure 26 resets the connection statistics only. The command shown in Figure 27 disables the API-crossing exit and changes the trace number to 121.

```
CKQC MODIFY Y
```

Figure 26. Resetting connection statistics from the command line

```
CKQC MODIFY N D 121
```

Figure 27. Changing the adapter's trace number and disabling the API-crossing exit from the command line

## Operating the CICS adapter

### Modifying a connection from a CICS application program

To modify a connection from a CICS program, the program must link to the adapter reset program, CSQCRST.

Figure 28 shows the format of the LINK command. It has the same effect as the command-line requests described in “Modifying a connection from the CICS command line” on page 57.

```
EXEC CICS LINK PROGRAM('CSQCRST ')
          INPUTMSG('CKQC MODIFY    Y E <trace-number>')
```

Figure 28. Format of the MODIFY command issued from a CICS adapter application program

The command shown in Figure 29 resets the connection statistics only.

```
EXEC CICS LINK PROGRAM('CSQCRST ')
          INPUTMSG('CKQC MODIFY    Y')
```

Figure 29. Resetting connection statistics from a CICS program

The command shown in Figure 30 disables the API-crossing exit and changes the trace number to 121.

```
EXEC CICS LINK PROGRAM('CSQCRST ')
          INPUTMSG('CKQC MODIFY    N D 121')
```

Figure 30. Linking to the adapter reset program, CSQCRST, from a CICS program

**Note:** The MODIFY command must be padded to 10 characters, see “Command syntax in application programs” on page 48.

## Displaying details of connections and CICS tasks

You can use the CICS adapter control panels to display details of the current connection. The equivalent functionality is not available from the CICS command line or from a CICS application program. However, you can obtain some status information using the CKQC DISPLAY command, see “Displaying connection status and in-flight tasks” on page 66.

### Displaying details of a connection from the CICS adapter control panels

From the initial panel:

1. Select **Connection** from the menu bar.
2. Select the **Display** action from the pull-down menu.

Figure 31 shows the details provided:

```

CKQCM2                Display Connection panel

Read connection information. Then press F12 to cancel.

CICS Applid = VICIC14  Connection Status = Connected  Qmgrname = VCA
Trace No.   = 124      Tracing           = On         API Exit = Off
Initiation Queue Name = VICIC14.INITIATION.QUEUE
----- S T A T I S T I C S -----
Number of in-flight tasks = 1          Total No. of API calls = 43912
Number of running CKTI   = 1
          APIs and flows analysis                Syncpoint                Recovery
-----
Run OK      43874  MQINQ      6806  Tasks      26  Indoubt    0
Futile     0      MQSET      0      Backout    0  UnResol   0
MQOPEN     6833  ----- Flows -----  Commit     10  Commit    0
MQCLOSE    6823  Calls      43952  S-Phase    10  Backout   0
MQGET     10032  SyncComp   43922  2-Phase    0
GETWAIT    3399  SuspReqd   0
MQPUT     13399  MsgWait    7      InitTCBs  8  StrtTCBs  8  BusyTCBs  0
MQPUT1     5      Switched   43940
-----
F1=Help  F12=Cancel  Enter=Refresh
    
```

Figure 31. The display connection panel

The display is organized into three areas:

- Top: parameters used for the connection, and current status.
- Middle: connection statistics. These are totals for the current connection, since statistics were last reset.
- Bottom: statistics produced by the adapter.

For an explanation of specific fields on this screen, view the online help panels by pressing function key F1.

## Starting an instance of the task initiator CKTI

CKTI is the MQSeries-supplied task initiator<sup>1</sup> used in a CICS environment to start a transaction when the trigger conditions on any of its associated MQSeries queues are met.

You can start a CKTI instance from:

- The CICS adapter control panels
- The CICS command line
- A CICS application program
- From emulated terminals automatically

### Starting CKTI from the CICS adapter control panels

From the initial panel:

1. Select **CKTI** from the menu bar.
2. Select the **Start** action from the pull-down menu.
3. In the **Start Task Initiator** secondary window, use the **Initiation Queue Name** field to specify the name of the initiation queue to be serviced by this CKTI instance. If you leave this field blank, the default initiation queue is used, if defined.

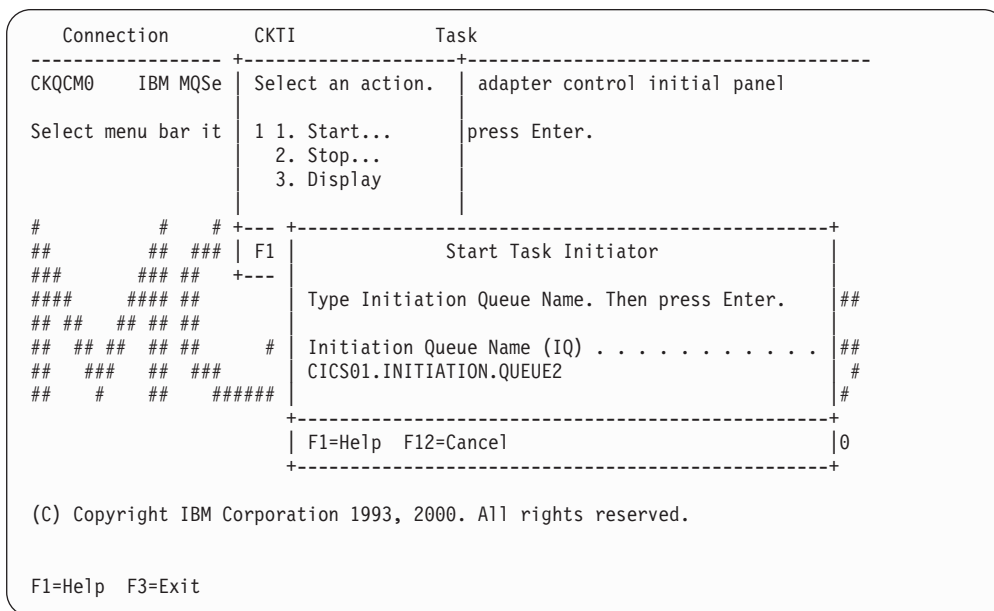


Figure 32. Starting an instance of CKTI

1. Trigger monitor in MQSeries terminology.

## Starting CKTI from the CICS command line

The command shown in Figure 33 starts an instance of CKTI to serve the default initiation queue, if defined.

```
CKQC STARTCKTI
```

Figure 33. Starting an instance of CKTI—for the default initiation queue

The command shown in Figure 34 starts an instance of CKTI to serve a specified initiation queue.

```
CKQC STARTCKTI CICS01.INITIATION.QUEUE2
```

Figure 34. Starting an instance of CKTI—for a specified initiation queue

## Starting CKTI from a CICS application program

To start an instance of CKTI from a CICS program, the program must link to the adapter task initiation program, CSQCSSQ. Figures 35 through 36 show suitable LINK commands. When you do an EXEC CICS LINK to CSQCSSQ, the program requires a terminal associated task.

```
EXEC CICS LINK PROGRAM('CSQCSSQ ')
        INPUTMSG('CKQC STARTCKTI ')
```

Figure 35. Linking to the adapter task-initiator program CSQCSSQ from CICS. This starts a CKTI that uses the default initiation queue.

```
EXEC CICS LINK PROGRAM('CSQCSSQ ')
        INPUTMSG('CKQC STARTCKTI CICS01.INITIATION.QUEUE2')
```

Figure 36. Linking to the adapter task-initiator program CSQCSSQ from CICS. This starts a CKTI that uses a named initiation queue.

Output messages from CSQCSSQ are displayed on the system console.

**Note:** The STARTCKTI command must be padded to 10 characters; see “Command syntax in application programs” on page 48.

## Starting CKTI automatically

To automate the starting of CKTIs under a specific user ID, you can use an automation product, for example, NetView. You can use this to sign on a CICS console and issue the STARTCKTI command.

You can also use preset security sequential terminals, which have been defined to emulate a CRLP terminal, with the sequential terminal input containing the CKQC STARTCKTI command.

However, when the CICS adapter alert monitor reconnects CICS to MQSeries after, for example, an MQSeries restart, only the CKTI specified at the initial MQSeries connection is restarted. You must automate starting any extra CKTIs yourself.

## Stopping an instance of CKTI

You can stop an instance of CKTI by using:

- The CICS adapter control panels
- The CICS command line
- A CICS application program

### Stopping an instance of CKTI from the CICS adapter control panels

From the initial panel:

1. Select **CKTI** from the menu bar.
2. Select the **Stop** action from the pull-down menu.
3. Use the **Stop Task Initiator** secondary window to specify the name of the initiation queue serviced by this instance of CKTI. If you leave the name blank, the default initiation queue, if defined, is used.

```

      Connection      CKTI      Task
-----+-----+-----
CKQCM0  IBM  MQSe | Select an action. | adapter control initial panel
Select menu bar it | 2 1. Start...    | press Enter.
                | 2. Stop...      |
                | 3. Display      |
#           #       +---+-----+
##          ## ### | F1 | Stop Task Initiator
###         ### ## | +---+
#####     ##### ## | Type Initiation Queue Name. Then press Enter. ##
## ##     ## ## ## |
## ## ##   ## ##   # | Initiation Queue Name (IQ) . . . . . ##
##   ###   ##   ### | CICS01.INITIATION.QUEUE2 #
##    #    ##   ##### |
                | F1=Help  F12=Cancel |0
                +-----+
(C) Copyright IBM Corporation 1993, 2000. All rights reserved.

F1=Help  F3=Exit
    
```

Figure 37. Stopping an instance of the task initiator CKTI

## Stopping an instance of CKTI from the command line

The command shown in Figure 38 stops an instance of CKTI that is serving the default initiation queue, if there is one.

```
CKQC STOPCKTI
```

Figure 38. Stopping an instance of CKTI from the command line—for the default initiation queue

The command shown in Figure 39 stops the instance of CKTI that is serving a specified initiation queue.

```
CKQC STOPCKTI CICS01.INITIATION.QUEUE2
```

Figure 39. Stopping an instance of CKTI from the command line—for a specified initiation queue

## Stopping an instance of CKTI from an application program

You can stop an instance of CKTI by linking to the adapter task-initiator program, CSQCSSQ. Figures 40 through 41 show alternative LINK commands to stop an instance of CKTI from a CICS program. The first command stops the CKTI that is serving the default initiation queue; the second stops the CKTI serving a specified initiation queue.

```
EXEC CICS LINK PROGRAM('CSQCSSQ ')
        INPUTMSG('CKQC STOPCKTI  ')
```

Figure 40. Stopping an instance of CKTI from a program—for the default initiation queue from CICS

```
EXEC CICS LINK PROGRAM('CSQCSSQ ')
        INPUTMSG('CKQC STOPCKTI  CICS01.INITIATION.QUEUE2')
```

Figure 41. Stopping an instance of CKTI from a program—for a specified initiation queue from CICS

**Note:** The STOPCKTI command must be padded to 10 characters; see “Command syntax in application programs” on page 48.

## Displaying the current instances of CKTI

You can use the CICS adapter control panels to display details of the current instances of CKTI. The equivalent functionality is not available from the CICS command line or from a CICS application program.

### Displaying the current instances of CKTI from the CICS adapter control panels

From the initial panel:

1. Select **CKTI** from the menu bar
2. Select the **Display** action from the pull-down menu

Figure 42 shows the details provided for each instance of CKTI:

- CICS task number
- Task status
- Thread status
- Number of API calls it has issued
- Most recent API call it has issued
- Name of the initiation queue it is serving

Press function key F1 to display help information about each of the fields in this panel.

```
CKQCM4                      Display CKTI panel
Read CKTI status information. Then press F12 to cancel.

CKTI  1 to  1 of  1

Task No.   Task Status   Thread Status   No-of-APIs   Last API
-----
0000123    Normal             Msg Wait        2            MQGET
Initiation Queue Name: CICS01.INITIATION.QUEUE1

F1=Help  F7=Backward  F8=Forward  F12=Cancel  Enter=Refresh
```

Figure 42. The CKQC Display CKTI panel



## Displaying CICS task information

You can use the CICS adapter control panels to display information about CICS tasks using MQI calls. The equivalent functionality is not available from the CICS command line or from a CICS application program. However, you can obtain some status information using the CKQC DISPLAY command, see “Displaying connection status and in-flight tasks” on page 66.

### Displaying CICS tasks from the CICS adapter control panels

You can display information about the CICS tasks that are currently using MQI calls. From the initial panel:

1. Select **Task** from the menu bar.
2. Select an action from the pull-down menu.
  - Select option 1, **List all tasks** to obtain information about all tasks that are currently active. To limit the scope of the display, select option 2, **List from task**, to specify the starting number of the first task to be displayed.

CKQCM3 Display Task panel

Read task status information. Then press F12 to cancel.

Tasks 1 to 3 of 3

Txn Id	User Id	Task No.	Task Status	Thread Status	Total APIs	Res Sec	In API-X	Last MQ-Call	Thread ID
PUTQ	CICSUSER	00065	Normal	InQueue	102	No	No	MQPUT1	00012420
GETQ	CICSUSER	00067	Normal	BtnCalls	22	No	No	MQOPEN	00012620
CKTI	CICSUSER	00123	Normal	Msg Wait	2	No	No	MQGET	00012C20

F1=Help F7=Backward F8=Forward F12=Cancel Enter=Refresh

Figure 43. The CKQC Display Task panel

Figure 43 shows the details provided for each CICS task:

- Transaction ID (name)
- User ID
- CICS task number
- Task status
- Thread status
- Total number of API calls issued by this task
- Whether resource security checking is active for this task
- Whether this task is currently in the API-crossing exit
- Most recent API call issued by this task
- Thread ID used by MQSeries

## Operating the CICS adapter

### Displaying connection status and in-flight tasks

You can use the CKQC DISPLAY command to display limited information about the current connection and CICS tasks from either the CICS command line or from a CICS application program. The information from this command is returned in a message CSQC453I, see Figure 44. This message contains:

- The name of the MQSeries subsystem.
- The status of the connection.
- The number of in-flight tasks that are still using the connection.

```
CSQC453I VICY06 CSQCDSPL Status of connection to JAC2 is Connected. 2
tasks are in-flight
```

Figure 44. Message showing the status of a connection

To obtain more detailed information, use the CICS adapter control panels. See “Displaying details of a connection from the CICS adapter control panels” on page 59 and “Displaying CICS tasks from the CICS adapter control panels” on page 65, respectively.

#### From the CICS command line

You can use the CKQC DISPLAY command, shown in Figure 45, from the CICS command line.

```
CKQC DISPLAY
```

Figure 45. Displaying the status of a connection

Figure 44 shows a typical response to this command. The response messages are sent to your CICS terminal.

#### From a CICS application program

Figure 46 shows the LINK command for displaying the status of a connection from a CICS application program.

```
EXEC CICS LINK PROGRAM('CSQCDSPL') INPUTMSG('CKQC DISPLAY')
```

Figure 46. Linking to the adapter program CSQCDSPL from a CICS program

Figure 44 shows a typical output from this command. The response messages are sent to the CKQQ queue (the transient data queue).

The COMMAREA option can be used instead of INPUTMSG but only when the program is run at PLT time.

---

## Purging tasks that are using the CICS adapter

You can use the CICS CEMT transaction to purge user tasks that are using the CICS adapter. Tasks that are waiting on the adapter respond only to CEMT SET TASK FORCEPURGE commands—CEMT SET TASK PURGE commands are ignored. The way the adapter handles a FORCEPURGE command depends on the kind of wait state that the task is in:

- If a task is waiting for a message to arrive, for example, the application has issued an **MQGET WAIT** call, the task is abended with code AEXY immediately.
- If the task is waiting for an MQI request to be completed by MQSeries, message CSQC413I is displayed on the system console.

The adapter waits for the request to complete, and then checks whether it is suitable to abend the task:

- If the task is in a critical state, the CICS adapter lets the task continue and ignores the attempt to purge it. This is done to preserve data and system integrity. Message CSQC415I is displayed.

A task is in a critical state is when, for example, it is in the process of completing phase 2 of a two-phase commit sequence.

- If the task is not in a critical state, the adapter abends it with code AEXY. Message CSQC414I is displayed.

For information about CEMT commands, see the *CICS-Supplied Transactions* manual.

## Shutting down a connection between MQSeries and the CICS adapter

You can shut down a connection between MQSeries and the CICS adapter by using the CKQC transaction or an application program. There are two types of shutdown:

- Forced
- Quiesced

Other forms of connection shutdown result from a termination of CICS or MQSeries. Table 2 summarizes how the adapter handles different forms of connection shutdown.

Table 2. Shutting down a CICS adapter connection

Method of shutdown	How this is handled by the adapter
CKQC STOP ( <i>A quiesced shutdown</i> )	Mark the status of the adapter as <i>Quiescing</i> . Allow both active and waiting tasks to complete. Allow syncpoint. Do not allow calls from a new task. The last task initiates disconnection from MQSeries.
CKQC STOP FORCE	Mark the status of the adapter as <i>StoppingForce</i> . Disconnect from MQSeries. Resume waiting tasks. Fail any in-flight or following MQI calls.
CICS warm shutdown	Issue message CSQC411I. Initiate a quiesced shutdown of the connection; see CKQC STOP, above.
CICS immediate shutdown	Issue message CSQC410I. Any in-flight tasks using MQSeries are backed out.
CICS abend	Issue message CSQC412I.
MQSeries quiesced	Initiate a quiesced shutdown of the connection; see CKQC STOP, above.
MQSeries abend or forced shutdown	Initiate a forced shutdown of connection; see CKQC STOP FORCE, above.
<p><b>Notes:</b></p> <ol style="list-style-type: none"> <li>1. If the connection is not active (for example, quiesced) when CICS or MQSeries shuts down, no action is taken and no messages are issued.</li> <li>2. "Waiting tasks" includes instances of CKTI, which you must stop before shutdown completes.</li> </ol>	

### Orderly shutdown

An orderly shutdown of the connection lets each CICS transaction terminate before thread subtasks are detached. When you use this method, there should be no in-doubt units of work when you reconnect CICS. An orderly termination occurs in each of the following situations:

- The CICS terminal operator issues a CKQC STOP command. CICS and MQSeries remain active. The command can be issued from the command line, from a terminal using the CKQC panels, or from a program, see page 55.
- The CICS terminal operator issues the CICS command:

```
CEMT PERFORM SHUTDOWN
```

For information about the CEMT PERFORM SHUTDOWN command, see the *CICS-Supplied Transactions* manual.

- MQSeries is quiesced by the command:

```
+CSQ1 STOP QMGR MODE(QUIESCE)
```

This stops the MQSeries subsystem, allows the currently identified tasks to continue normal execution, and does not allow new tasks to identify themselves to MQSeries. CICS remains active.

### Forced shutdown

A forced shutdown of the connection can abend CICS transactions connected to MQSeries. Therefore, there might be in-doubt units of work when the system is reconnected. A forced shutdown occurs in each of these situations:

- The CICS terminal operator issues the CKQC STOP FORCE command. The command can be issued from the command line, from a terminal using the CKQC panels, or from a program, see page 55.
- The CICS terminal operator issues the CICS immediate termination command: For information about this command, see the *CICS-Supplied Transactions* manual.

```
CEMT PERFORM SHUTDOWN IMMEDIATE
```

MQSeries remains active.

- The MQSeries forced termination command is issued:

```
+CSQ1 STOP QMGR MODE(FORCE) or +CSQ1 STOP QMGR MODE(RESTART)
```

CICS remains active.

- An MQSeries abend occurs. CICS remains active.
- CICS abend occurs. MQSeries remains active.

## Operating the CICS adapter

---

## Chapter 5. Operating the CICS bridge

This chapter describes how to operate the MQSeries CICS bridge. It discusses the following topics:

- “Starting the CICS bridge”
- “Shutting down the CICS bridge” on page 72
- “Controlling CICS-bridge throughput” on page 72

---

### Starting the CICS bridge

To start the bridge, you need to run the CKBR transaction providing a maximum of three parameters:

- `Q=qqq`, where `qqq` is the name of the queue holding requests. If you don't specify one, the default is `SYSTEM.CICS.BRIDGE.QUEUE`.

Remember that names of objects within MQSeries are case-sensitive.

- `WAIT=nnn`, where `nnn` is the number of seconds that you want the bridge task to wait for second and subsequent requests before timing out when processing a unit of work that runs many user programs.

The default wait time is unlimited.

You are recommended to specify a wait time. If you don't, the CICS bridge might inhibit CICS or MQSeries shut down.

- `AUTH=xxx`, where `xxx` is the security option. The default is `LOCAL`.

Start the CKBR task running by using one of the following methods:

- Input a single line from a terminal (3270 or other). Note that the terminal is not freed until the monitor ends. The format is:

```
CKBR Q = <queue name>, AUTH = <auth option>, WAIT = nnn
```

For example:

```
CKBR Q = MyQueue, AUTH = IDENTIFY, WAIT = 30
```

- Issue an EXEC CICS START for the CKBR program with the parameters as data.
- Issue an EXEC CICS LINK to the program CSQCBR00 with the parameters as data in the commarea.
- Use TRIGGER TRIGTYPE(FIRST) from the bridge request queue to a process specifying APPLICID(CKBR), with any parameters for the AUTH and WAIT options in USERDATA.

The level of security you want to use will influence how you start the monitor task. See the *MQSeries for OS/390 System Setup Guide* for information on the security options available to you.

### Shutting down the CICS bridge

There are various ways in which you can shut down the CICS bridge:

- By altering the attributes of the request queue by setting GET(DISABLED)
- By shutting CICS down
- By shutting MQSeries down

Whichever method you choose, it will attempt to allow all the requests in progress to complete first.

However in the event that this is not possible, the problems encountered are reported on the CICS CSMT log.

**Note:** The CICS bridge does not stop CICS or MQSeries if either of them are in the process of shutting down, unless bridge tasks started with WAIT\_UNLIMITED have **MQGET** calls outstanding for second or subsequent messages in a unit of work.

#### **Restarting the monitor**

The monitor requires exclusive use of the request queue during its initialization, so the monitor cannot be restarted until all bridge tasks for the queue have terminated.

---

### Controlling CICS-bridge throughput

You can control the throughput of the bridge by putting the bridge transaction, CKBP, in a class of its own and setting the CLASSMAXTASK to suit your requirements.

If a high volume of requests is expected, you could consider starting a second or subsequent monitor task. To do this, you must create another request queue for the sole use of this monitor (and the bridge tasks it starts).



---

## Part 3. MQSeries and IMS

<b>Chapter 6. Operating the IMS adapter</b> . . . . .	75
Controlling IMS connections. . . . .	75
Connecting from the IMS control region. . . . .	76
Initializing the adapter and connecting to	
MQSeries . . . . .	76
Thread attachment . . . . .	77
Displaying in-doubt units of recovery . . . . .	78
Recovering in-doubt units of recovery . . . . .	78
Resolving residual recovery entries . . . . .	79
Controlling IMS dependent region connections . . . . .	80
Connecting from dependent regions . . . . .	80
Region error options . . . . .	80
Monitoring the activity on connections . . . . .	80
Disconnecting from dependent regions . . . . .	81
Disconnecting from IMS . . . . .	82
Controlling the IMS trigger monitor . . . . .	83
Starting CSQQTRMN . . . . .	83
Stopping CSQQTRMN. . . . .	83
<b>Chapter 7. Controlling the IMS bridge</b> . . . . .	85
Starting and stopping the IMS bridge. . . . .	85
Controlling IMS connections. . . . .	85
Controlling bridge queues . . . . .	86
Resynchronizing the IMS bridge . . . . .	87
Considerations for Commit mode 1 transactions	87
Deleting messages from IMS. . . . .	88
Deleting Tpipes . . . . .	88



---

## Chapter 6. Operating the IMS adapter

This chapter describes how to operate the IMS adapter, which connects MQSeries to IMS systems.

**Note:** The IMS adapter does not incorporate any operations and control panels.

This chapter contains the following sections:

- “Controlling IMS connections”
- “Connecting from the IMS control region” on page 76
- “Displaying in-doubt units of recovery” on page 78
- “Controlling IMS dependent region connections” on page 80
- “Disconnecting from IMS” on page 82
- “Controlling the IMS trigger monitor” on page 83

---

### Controlling IMS connections

IMS provides these operator commands to control and monitor the connection to MQSeries:

**/CHANGE SUBSYS**

Deletes an in-doubt unit of recovery from IMS.

**/DISPLAY OASN SUBSYS**

Displays outstanding recovery elements.

**/DISPLAY SUBSYS**

Displays connection status and thread activity.

**/START SUBSYS**

Connects the IMS control region to an MQSeries subsystem.

**/STOP SUBSYS**

Disconnects IMS from an MQSeries subsystem.

**/TRACE**

Controls the IMS trace.

For more information about these commands, see the *IMS/ESA Operator's Reference* manual for the level of IMS that you are using.

IMS command responses are sent to the terminal from which the command was issued. Authorization to issue IMS commands is based on IMS security.

### Connecting from the IMS control region

IMS makes one connection from its control region to each MQSeries subsystem. IMS can make the connection in these ways:

- Automatically during either:
  - A cold-start initialization.
  - A warm start of IMS, if the MQSeries connection was active when IMS was shut down.
- In response to the IMS command:  
`/START SUBSYS sysid`

where *sysid* is the MQSeries subsystem name.

This command causes the following message to be displayed at the logical terminal (LTERM):

```
DFS058  START COMMAND COMPLETED
```

The command is issued regardless of whether MQSeries is active or not, and does not imply that the connection has been established.

The order in which you start IMS and MQSeries is not significant. If you start IMS first, then, when MQSeries starts, MQSeries posts the control region modify task, and IMS again tries to reconnect.

IMS cannot reconnect to MQSeries automatically if MQSeries is stopped with a STOP QMGR command, the /STOP SUBSYS IMS command, or an abnormal end.

Therefore, you must make the connection by using the /START SUBSYS IMS command.

### Initializing the adapter and connecting to MQSeries

The adapter is a set of modules loaded into the IMS control and dependent regions, using the IMS external subsystem attach facility.

This procedure initializes the adapter and connects to MQSeries:

1. Read the subsystem member (SSM) from IMS.PROCLIB. The SSM chosen is an IMS EXEC parameter. There is one entry in the member for each MQSeries subsystem to which IMS can connect. Each entry contains control information about an MQSeries adapter.
2. Load the IMS adapter.

**Note:** IMS loads one copy of the adapter modules for each MQSeries instance that is defined in the SSM member.

3. Attach the external subsystem task for MQSeries.
4. Run the adapter with the CTL EXEC parameter (IMSID) as the connection name.

The process is the same whether the connection is part of initialization or a result of the /START SUBSYS IMS command.

If MQSeries is active when IMS tries to make the connection, the following messages are sent:

- To the OS/390 console:  
DFS3613I ESS TCB INITIALIZATION COMPLETE
- To the IMS master terminal:  
CSQQ000I IMS/TM *imsid* connected to queue manager *ssnm*

When IMS tries to make the connection and *MQSeries is not active*, the following messages are sent to the IMS master terminal each time an application makes an MQI call:

```
CSQQ001I IMS/TM imsid not connected to queue manager ssnm.  
Notify message accepted  
DFS3607I MQM1 SUBSYSTEM ID EXIT FAILURE, FC = 0286, RC = 08,  
JOBNAME = IMSEMPR1
```

If you get DFS3607I messages when you start the connection to IMS or on system startup, this indicates that MQSeries is not available. To prevent a large numbers of messages being generated, you must do one of the following:

1. Start the relevant MQSeries subsystem.
2. Issue a /STOP SUBSYS IMS command so that IMS does not expect to connect to the MQSeries subsystem.

If you do neither, a DFS2607I message and the associated CSQQ001I message are issued each time a job is scheduled in the region and each time a connection request to MQSeries is made by an application.

## Thread attachment

In an MPP or IFP region, IMS makes a thread connection when the first application program is scheduled into that region, even if that application program does not make an MQSeries call. In a BMP region, the thread connection is made when the application makes its first MQSeries call (**MQCONN** or **MQCONN**X). This thread is retained for the duration of the region or until the connection is stopped.

For both the message driven and non-message driven regions, the recovery thread cross-reference identifier, *Thread-xref*, associated with the thread is:

```
PSTid + PSBname
```

where:

**PSTid** Partition specification table region identifier

**PSBname**

Program specification block name

You can use connection IDs as unique identifiers in MQSeries commands; if you do, MQSeries automatically inserts these IDs into any operator message that it generates.

### Displaying in-doubt units of recovery

The operational steps used to list and recover in-doubt units of recovery are discussed here for relatively simple cases only.

If MQSeries ends abnormally while connected to IMS, it is possible for IMS to commit or back out work without MQSeries being aware of it. When MQSeries restarts, that work is termed *in doubt*. A decision must be made about the status of the work.

To display a list of in-doubt units of recovery, issue the command:

```
+CSQ1 DISPLAY THREAD(*) TYPE(INDOUBT)
```

MQSeries responds with the following messages:

```
CSQV401I +CSQ1 DISPLAY THREAD REPORT FOLLOWS -
CSQV406I +CSQ1 INDOUBT THREADS - 154
NAME      THREAD-XREF  URID  NID
IMSJ      0002MQSPRG1      IMSJ.5600000000
IMSJ      0001MQSINQ      IMSJ.5700000000
DISPLAY THREAD REPORT COMPLETE
CSQ9022I +CSQ1 CSQVDT ' DISPLAY THREAD' NORMAL COMPLETION
```

where:

#### NAME

The connection name, which is the IMS system ID (the IMSID parameter from the region JCL).

#### THREAD-XREF

The associated thread cross-reference, see “Thread attachment” on page 77.

**NID** The associated *net-node.number* taken from the IMS recovery token, where *net-node* is the IMS system ID (with trailing blanks suppressed), *number* is the OASN and commit number (leading zeros suppressed).

For a formal explanation of the displayed list, see the description of message CSQV406I in the *MQSeries for OS/390 Messages and Codes* manual.

### Recovering in-doubt units of recovery

To recover in-doubt units of recovery, issue this command:

```
+CSQ1 RESOLVE INDOUBT(connection-name) ACTION(COMMIT|BACKOUT)
NID(net-node.number)
```

where:

*connection-name*

The IMS system ID.

#### ACTION

Indicates whether to commit (COMMIT) or back out (BACKOUT) this unit of recovery.

*net-node.number*

The associated net-node.number.

One of the following messages is generated after the RESOLVE INDOUBT command:

```
CSQV414I +CSQ1 THREAD network-id COMMIT SCHEDULED
CSQV415I +CSQ1 THREAD network-id BACKOUT SCHEDULED
```

## Resolving residual recovery entries

At given times, IMS builds a list of residual recovery entries (RREs). RREs are units of recovery about which MQSeries could be in doubt. They arise in several situations:

- If MQSeries is not operational, IMS has RREs that cannot be resolved until MQSeries is operational. These RREs are not a problem.
- If MQSeries is operational and connected to IMS, and if IMS backs out the work that MQSeries has committed, the IMS adapter issues message CSQQ010E. If the data in the two systems must be consistent, there is a problem. Resolving it is discussed in “Recovering IMS units of recovery manually” on page 145.
- If MQSeries is operational and connected to IMS, there might still be RREs even though no messages have informed you of this problem. After the MQSeries connection to IMS has been established, you can issue the following IMS command to find out if there is a problem:

```
/DISPLAY OASN SUBSYS sysid
```

To purge the RRE, issue one of the following IMS commands:

```
/CHANGE SUBSYS sysid RESET
/CHANGE SUBSYS sysid RESET OASN nnnn
```

where *nnnn* is the originating application sequence number listed in response to your +CSQ1 DISPLAY command. This is the schedule number of the program instance, giving its place in the sequence of invocations of that program since the last IMS cold start. IMS cannot have two in-doubt units of recovery with the same schedule number.

These commands reset the status of IMS; they do not result in any communication with MQSeries.

### Controlling IMS dependent region connections

Controlling IMS dependent region connections involves these activities:

- Connecting from dependent regions
- Region error options
- Monitoring the activity on connections
- Disconnecting from dependent regions

#### Connecting from dependent regions

The IMS adapter used in the control region is also loaded into dependent regions. A connection is made from each dependent region to MQSeries. This connection is used to coordinate the commitment of MQSeries and IMS work. To initialize and make the connection, IMS does the following:

1. It reads the subsystem member (SSM) from IMS.PROCLIB.

A subsystem member can be specified on the dependent region EXEC parameter. If it is not specified, the control region SSM is used. If the region is never likely to connect to MQSeries, to avoid loading the adapter, specify a member with no entries.

2. It loads the MQSeries adapter.

For a batch message program, the load is not done until the application issues its first messaging command. At that time, IMS tries to make the connection.

For a message-processing program region or IMS fast-path region, the attempt is made when the region is initialized.

#### Region error options

If MQSeries is not active, or if resources are not available when the first messaging command is sent from application programs, the action taken depends on the error option specified on the SSM entry. The options are:

- R** The appropriate return code is sent to the application.
- Q** The application ends abnormally with abend code U3051. The input message is re-queued.
- A** The application ends abnormally with abend code U3047. The input message is discarded.

#### Monitoring the activity on connections

A thread is established from a dependent region when an application makes its first successful MQSeries request. Information on connections and the applications currently using them can be displayed by issuing the following command from MQSeries:

```
+CSQ1 DISPLAY THREAD (connection-name)
```



The command produces the following messages:

```

CSQV401I +CSQ1 DISPLAY THREAD REPORT FOLLOWS -
CSQV402I +CSQ1 ACTIVE THREADS -
NAME  ST  A  REQ  THREAD-XREF  USERID  ASID  URID
name  s   *  ct   thread-xref  auth-id  asid  urid
name  s   *  ct   thread-xref  auth-id  asid  urid
DISPLAY ACTIVE REPORT COMPLETE
CSQ9022I +CSQ1 CSQVDT 'DIS THREAD' NORMAL COMPLETION
    
```

For the control region, *thread-xref* is the special value CONTROL. For dependent regions, it is the PSTid concatenated with the PSBname. *auth-id* is either the user field from the job card, or the ID from the OS/390 started procedures table.

For an explanation of the displayed list, see the description of message CSQV402I in the *MQSeries for OS/390 Messages and Codes* manual.

IMS provides a display command to monitor the connection to MQSeries. It shows which program is active on each dependent region connection, the LTERM user name, and the control region connection status. The command is:

```
/DISPLAY SUBSYS name
```

The status of the connection between IMS and MQSeries is shown as one of:

```

CONNECTED
NOT CONNECTED
CONNECT IN PROGRESS
STOPPED
STOP IN PROGRESS
INVALID SUBSYSTEM NAME=name
SUBSYSTEM name NOT DEFINED BUT RECOVERY OUTSTANDING
    
```

The thread status from each dependent region one of the following:

```

CONN
CONN, ACTIVE (includes LTERM of user)
    
```

## Disconnecting from dependent regions

To change values in the SSM member of IMS.PROCLIB, you disconnect a dependent region. To do this, you must:

1. Issue the /STOP REGION IMS command
2. Update the SSM member
3. Issue the /START REGION IMS command

### Disconnecting from IMS

The connection is ended when either IMS or MQSeries terminates. Alternatively, the IMS master terminal operator can explicitly break the connection by issuing the following IMS command:

```
/STOP SUBSYS sysid
```

The command sends the following message to the terminal that issued it, usually the master terminal operator (MTO):

```
DFS058I STOP COMMAND IN PROGRESS
```

The `/START SUBSYS sysid` IMS command is required to re-establish the connection.

**Note:** The `/STOP SUBSYS` IMS command will not be completed if an IMS trigger monitor is running.

## Controlling the IMS trigger monitor

The IMS trigger monitor (the CSQQTRMN transaction) is described in the *MQSeries for OS/390 Concepts and Planning Guide*.

### Starting CSQQTRMN

1. Start a batch oriented BMP running the program CSQQTRMN for each initiation queue you want to monitor.
2. Modify your batch JCL (described in the *MQSeries for OS/390 System Setup Guide*) to add a DDname of CSQQUT1 that points to a data set containing the following information:  
where:

QMGRNAME=q_manager_name	Comment: queue manager name
INITQUEUEUENAME=init_q_name	Comment: initiation queue name
LTERM=lterm	Comment: LTERM to remove error messages
CONSOLEMESSAGES=YES	Comment: Send error messages to console

q_manager_name	The name of the queue manager (if this is blank, the default nominated in CSQQDEFV is assumed)
init_q_name	The name of the initiation queue to be monitored
lterm	The IMS LTERM name for the destination of error messages (if this is blank, the default value is MASTER).
CONSOLEMESSAGES=YES	Requests that messages sent to the nominated IMS LTERM are also sent to the OS/390 console. If this parameter is omitted or misspelled then the default is NOT to send messages to the console.

3. Add a DD name of CSQQUT2 if you want a printed report of the processing of CSQQUT1 input.

#### Notes:

1. The data set CSQQUT1 is defined with LRECL=80. Other DCB information is taken from the data set. The DCB for data set CSQQUT2 is RECFM=VBA and LRECL=125.
2. You can put only one keyword on each record. The keyword value is delimited by the first blank following the keyword; this means that you can include comments. An asterisk in column 1 means that the whole input record is a comment.
3. If you misspell either of the QMGRNAME or LTERM keywords, CSQQTRMN will use the default for that keyword.
4. Ensure that the subsystem is started in IMS (by the /START SUBSYS command) before submitting the trigger monitor BMP job. If it is not started, your trigger monitor job will terminate, with abend code U3042.

### Stopping CSQQTRMN

Once started, CSQQTRMN runs until either the connection between MQSeries and IMS is broken due to one of the following events:

- MQSeries ending
- IMS ending

or an OS/390 STOP jobname command is entered.



---

## Chapter 7. Controlling the IMS bridge

This chapter describes how to control the IMS bridge. It discusses the following topics:

- “Starting and stopping the IMS bridge”
- “Controlling IMS connections”
- “Controlling bridge queues” on page 86
- “Resynchronizing the IMS bridge” on page 87
- “Deleting messages from IMS” on page 88
- “Deleting Tpipes” on page 88

There are no MQSeries commands to control the MQSeries-IMS bridge.

---

### Starting and stopping the IMS bridge

Start the MQSeries bridge by starting OTMA. Either use the IMS command `/START OTMA`, or start it automatically by specifying `OTMA=YES` in the IMS system parameters. If OTMA is already started, the bridge starts automatically when MQSeries startup has completed. An MQSeries event message is produced when OTMA is started.

Use the IMS command `/STOP OTMA` to stop OTMA communication. When this command is issued, an MQSeries event message is produced.

---

### Controlling IMS connections

IMS provides these operator commands to control and monitor the connection to MQSeries:

**/DEQUEUE TMEMBER** *tmember* **TPIPE** *tpipe*

Removes messages from a Tpipe, specify `PURGE` to remove all messages or `PURGE1` to remove the first message only.

**/DISPLAY OTMA**

Displays summary information about the OTMA server and clients, and client status.

**/DISPLAY TMEMBER** *name*

Displays information about an OTMA client.

**/DISPLAY TRACE TMEMBER** *name*

Displays information about what is being traced.

**/SECURE OTMA**

Sets security options.

**/START OTMA**

Enables communications through OTMA.

**/START TMEMBER** *tmember* **TPIPE** *tpipe*

Starts the named Tpipe.

**/STOP OTMA**

Stops communications through OTMA.

## Controlling the IMS bridge

`/STOP TMEMBER tmember TPIPE tpipe`  
Stops the named Tpipe.

`/TRACE`  
Controls the IMS trace.

For more information about these commands, see the *IMS/ESA Operator's Reference* manual for the level of IMS that you are using.

IMS command responses are sent to the terminal from which the command was issued. Authorization to issue IMS commands is based on IMS security.

---

## Controlling bridge queues

Issue the following IMS command to stop communicating with the MQSeries system with XCF member name *tmember* through the bridge:

```
/STOP TMEMBER tmember TPIPE ALL
```

Issue the following IMS command to resume communication:

```
/START TMEMBER tmember TPIPE ALL
```

To stop communication with the MQSeries system on a single Tpipe, issue the following IMS command:

```
/STOP TMEMBER tmember TPIPE tpipe
```

One or two Tpipes are created for each active bridge queue, so issuing this command stops communication with the MQSeries queue. Use the following IMS command to resume communication:

```
/START TMEMBER tmember TPIPE tpipe
```

Alternatively, you can alter the attributes of the MQSeries queue to make it get inhibited.

## Resynchronizing the IMS bridge

The IMS bridge is automatically restarted whenever MQSeries, IMS, or OTMA are restarted.

The first task undertaken by the IMS bridge is to resynchronize with IMS. This involves MQSeries and IMS checking sequence numbers on every synchronized Tpipe. A synchronized Tpipe is used when persistent messages are sent to IMS from an MQSeries-IMS bridge queue using commit mode 0 (commit-then-send).

If the bridge is unable to resynchronize with IMS at this time, the IMS sense code is returned in message CSQ2023E and the connection to OTMA is stopped. If the bridge is unable to resynchronize with an individual IMS Tpipe at this time, the IMS sense code is returned in message CSQ2025E and the Tpipe is stopped. If a Tpipe has been cold started, the recoverable sequence numbers are automatically reset to 1.

If the bridge discovers mismatched sequence numbers when resynchronizing with a Tpipe, message CSQ2020E is issued. Use the MQSeries command RESET TPIPE to initiate resynchronization with the IMS Tpipe. You need to provide the XCF group and member name, and the name of the Tpipe; this information is provided by the message.

You can also specify:

- A new recoverable sequence number to be set in the Tpipe for messages sent by MQSeries, and to be set as the partners receive sequence number. If you do not specify this, the partners receive sequence number is set to the current MQSeries send sequence number
- A new recoverable sequence number to be set in the Tpipe for messages received by MQSeries, and to be set as the partners send sequence number. If you do not specify this, the partners send sequence number is set to the current MQSeries receive sequence number

If there is an unresolved unit of recovery associated with the Tpipe, this is also notified in the message. Use the RESET TPIPE MQSeries command to specify whether to commit it or back it out. If you commit the unit of recovery, the batch of messages has already been sent to IMS, and is deleted from the bridge queue. If you back the unit of recovery out, the messages are returned to the bridge queue, to be subsequently sent to IMS.

Commit mode 1 (Send-then-commit) Tpipes are not synchronized.

## Considerations for Commit mode 1 transactions

In IMS, commit mode 1 (CM1) transactions send their output replies before syncpoint.

It is possible that a CM1 transaction is unable to send its reply, for example because:

- The Tpipe on which the reply is to be sent is stopped
- OTMA is stopped
- The OTMA client (that is, MQSeries) has gone away
- The reply-to queue and dead-letter queue are unavailable

## Controlling the IMS bridge

For all of the above reasons, the IMS application sending the message will pseudo-abend with code U0119. The IMS transaction and program are not stopped in this case.

These reasons often prevent messages being sent into IMS, as well as replies being delivered from IMS. A U0119 abend can occur if:

- The Tpipe, or OTMA, or MQSeries are stopped while the message is in IMS
- IMS replies on a different Tpipe to the incoming message, and that Tpipe is stopped
- IMS replies to a different OTMA client, and that client is unavailable.

Whenever a U0119 abend occurs, both the incoming message to IMS and the reply messages to MQSeries are lost. If the output of a CM0 transaction cannot be delivered for any of the above reasons, it is queued on the Tpipe within IMS.

---

## Deleting messages from IMS

A message that is destined for MQSeries via the IMS bridge can be deleted if the Tmember/Tpipe is stopped. To delete one message for the MQSeries system with XCF member name *tmember*, issue the following IMS command:

```
/DEQUEUE TMEMBER tmember TPIPE tpipe PURGE1
```

To delete all the message on the Tpipe, issue the following IMS command:

```
/DEQUEUE TMEMBER tmember TPIPE tpipe PURGE
```

---

## Deleting Tpipes

There are no commands to delete IMS Tpipes created by MQSeries. They are deleted by IMS at the following times:

- Synchronized Tpipes are deleted when IMS is cold started.
- Non-synchronized Tpipes are deleted when IMS is restarted.



## Part 4. Managing MQSeries resources

<b>Chapter 8. Managing the logs</b> . . . . .	91		Adding a queue manager to a queue-sharing	
Archiving logs with the ARCHIVE LOG command	91		group . . . . .	117
Restarting the log archive process after a failure	93		Removing a queue manager from a	
Optimizing archive log reading from tape devices	93		queue-sharing group . . . . .	117
Printing log records . . . . .	93		Removing a queue-sharing group from the DB2	
Recovering logs . . . . .	93		tables . . . . .	118
Discarding archive log data sets . . . . .	94		Managing shared queues . . . . .	118
Automatic archive log data set deletion . . . . .	94		Recovering shared queues . . . . .	118
Manually deleting archive log data sets . . . . .	95		Moving shared queues . . . . .	118
Locate and discard archive log data sets . . . . .	95		Moving a queue from one Coupling Facility	
			structure to another . . . . .	118
<b>Chapter 9. Managing the BSDS</b> . . . . .	97		Moving a non-shared queue to a shared	
Finding out what the BSDS contains . . . . .	97		queue . . . . .	120
Time stamps in the BSDS . . . . .	97		Moving a shared queue to a non-shared	
Active log data set status . . . . .	98		queue . . . . .	120
Changing the BSDS . . . . .	99		Migrating non-shared queues to shared queues	121
Changes for active logs . . . . .	99		The first (or only) queue manager in the	
Adding record entries to the BSDS . . . . .	99		queue-sharing group . . . . .	121
Deleting information about the active log			Any other queue managers in the	
data set from the BSDS . . . . .	100		queue-sharing group . . . . .	121
Recording information about the log data set			Managing group objects . . . . .	122
in the BSDS . . . . .	100		Managing the Coupling Facility . . . . .	122
Enlarging the active log . . . . .	100		Adding a Coupling Facility structure . . . . .	122
Changes for archive logs . . . . .	100		Removing a Coupling Facility structure . . . . .	122
Adding an archive log . . . . .	100			
Deleting an archive log . . . . .	101			
Changing the password of an archive log . . . . .	101			
Recovering the BSDS . . . . .	102			
<b>Chapter 10. Managing page sets</b> . . . . .	105			
How to add a page set to a queue manager . . . . .	105			
What to do when one of your page sets becomes				
full . . . . .	106			
How to balance loads on page sets . . . . .	107			
Moving a non-shared queue . . . . .	107			
How to expand a page set . . . . .	109			
How to reduce a page set . . . . .	110			
How to back up and recover page sets . . . . .	111			
Creating a point of recovery . . . . .	111			
Method 1: Full backup . . . . .	111			
Method 2: Fuzzy backup . . . . .	112			
Backing up page sets . . . . .	112			
Backing up your object definitions . . . . .	113			
Recovering page sets . . . . .	113			
Simple recovery . . . . .	113			
Advanced recovery . . . . .	114			
What happens when MQSeries is restarted	114			
How to back up and restore queues using				
CSQUTIL . . . . .	115			
<b>Chapter 11. Managing queue-sharing groups</b>				
<b>and shared queues</b> . . . . .	117			
Managing queue-sharing groups . . . . .	117			
Adding a queue-sharing group to the DB2 tables	117			



---

## Chapter 8. Managing the logs

This chapter describes the tasks involved in managing the MQSeries logs. It contains these sections:

- “Archiving logs with the ARCHIVE LOG command”
- “Optimizing archive log reading from tape devices” on page 93
- “Printing log records” on page 93
- “Recovering logs” on page 93
- “Discarding archive log data sets” on page 94

---

### Archiving logs with the ARCHIVE LOG command

An authorized operator can archive the current MQSeries active log data sets whenever required using the ARCHIVE LOG command.

When you issue the ARCHIVE LOG command, MQSeries truncates the current active log data sets, then runs an asynchronous off-load, and updates the BSDS with a record of the off-load.

The ARCHIVE LOG command has a MODE(QUIESCE) option. With this option, MQSeries users are quiesced after a commit point, and the resulting point of consistency is captured in the current active log before it is off-loaded.

Consider using the MODE(QUIESCE) option when planning a backup strategy for off site recovery. It creates a system-wide point of consistency, which minimizes the number of data inconsistencies when the archive log is used with the most current backup page set copy during recovery. For example:

```
ARCHIVE LOG MODE(QUIESCE)
```

If the ARCHIVE LOG command is issued without specifying a TIME parameter, the quiesce time period defaults to the value of the QUIESCE parameter of the CSQ6ARVP macro. If the time required for the ARCHIVE LOG MODE(QUIESCE) to complete is less than the time specified, the command completes successfully; otherwise, the command fails when the time period expires. You can specify the time period explicitly by using the TIME option, for example:

```
ARCHIVE LOG MODE(QUIESCE) TIME(60)
```

This command specifies a quiesce period of up to 60 seconds before ARCHIVE LOG processing occurs.

**Attention:** Using this option when time is critical can cause a significant disruption in MQSeries availability for all jobs and users that use MQSeries resources.

By default, the command is processed asynchronously from the time you submit the command. (To process the command synchronously with other MQSeries commands use the WAIT(YES) option QUIESCE, but be aware that the OS/390 console is locked from MQSeries command input for the entire QUIESCE period.)

## Archiving logs

During the quiesce period:

- Jobs and users on MQSeries are allowed to go through commit processing, but are suspended if they try to update any MQSeries resource after the commit.
- Jobs and users that only read data can be affected, since they can be waiting for locks held by jobs or users that were suspended.
- New tasks can start, but they are not allowed to update data.

The DISPLAY THREAD output uses the message CSQV400I to indicate that a quiesce is in effect. For example:

```
CSQV401I +CSQ1 DISPLAY THREAD REPORT FOLLOWS -
CSQV400I +CSQ1 ARCHIVE LOG QUIESCE CURRENTLY ACTIVE
CSQV402I +CSQ1 ACTIVE THREADS -
NAME      ST A  REQ THREAD-XREF  USERID      ASID  URID
BATCH    T   14              CON0327     0016  000000000000
DISPLAY ACTIVE REPORT COMPLETE
CSQ9022I +CSQ1 CSQVDT ' DISPLAY THREAD' NORMAL COMPLETION
```

When all updates are quiesced, the quiesce history record in the BSDS is updated with the date and time that the active log data sets were truncated, and with the last-written RBA in the current active log data sets. MQSeries truncates the current active log data sets, switches to the next available active log data sets, and issues message CSQJ311E stating that off-load started.

If updates cannot be quiesced before the quiesce period expires, MQSeries issues message CSQJ317I, and ARCHIVE LOG processing terminates. The current active log data sets are not truncated and not switched to the next available log data sets, and off-load is not started.

Whether the quiesce was successful or not, all suspended users and jobs are then resumed, and MQSeries issues message CSQJ312I, stating that the quiesce is ended and update activity is resumed.

If ARCHIVE LOG is issued when the current active log is the last available active log data set, the command is not processed, and MQSeries issues this message:

```
CSQJ319I - csect-name CURRENT ACTIVE LOG DATA SET IS THE LAST
          AVAILABLE ACTIVE LOG DATA SET.  ARCHIVE LOG PROCESSING
          WILL BE TERMINATED.
```

If ARCHIVE LOG is issued when another ARCHIVE LOG command is already in progress, the new command is not processed, and MQSeries issues this message:

```
CSQJ318I - ARCHIVE LOG COMMAND ALREADY IN PROGRESS.
```

For information about the syntax of the ARCHIVE LOG command, see the *MQSeries MQSC Command Reference* manual. For information about the messages issued during archiving, see the *MQSeries for OS/390 Messages and Codes* manual.

## Restarting the log archive process after a failure

If there is a problem during the log archive process (for example, a problem with allocation or tape mounts), the archiving of the active log might be suspended. You can cancel the archive process and restart it by using the ARCHIVE LOG CANCEL OFFLOAD command. This command cancels any off-load processing currently in progress, and restarts the archive process. It starts with the oldest log data set that has not been archived, and proceeds through all active log data sets that need off-loading. Any log archive operations that have been suspended are restarted.

You should use this command only if you are sure that the current log archive task is no longer functioning, or if you want to restart a previous attempt that failed. This is because the command might cause an abnormal termination of the off-load task, which might result in a dump.

---

## Optimizing archive log reading from tape devices

Logging parameters are set using the CSQ6LOGP macro of the system parameter module when the queue manager is customized (described in the *MQSeries for OS/390 System Setup Guide*). To optimize archive log reading from tape devices, you can reset some of these parameters using the SET LOG command. These parameters are:

### MAXRTU

To specify the maximum number of dedicated tape units that can be allocated to read archive log tape volumes.

### DEALLCT

To specify the length of time that an allocated archive tape read unit is allowed to remain unused before it is deallocated.

You can display the settings of these parameters using the DISPLAY LOG command. These commands are described in the *MQSeries MQSC Command Reference*.

---

## Printing log records

You can extract and print log records using the CSQ1LOGP utility. For instructions, see “Chapter 20. The log print utility (CSQ1LOGP)” on page 221.

---

## Recovering logs

Normally, you do not need to back up and restore the MQSeries logs, especially if you are using dual logging. However, in rare circumstances, such as an I/O error on a log, you might need to recover the logs. Use Access Method Services to delete and redefine the data set, and then copy the corresponding dual log into it.

### Discarding archive log data sets

You must keep enough log records to recover units of recovery or perform media recovery if a page set is lost. Do not discard archive log data sets that might be required for recovery; if you discard these archive log data sets you might not be able to recover using your page set backups.

However, if you have confirmed that your archive log data sets can be discarded, you can do this in either of the following ways:

- Automatic archive deletion
- Manual archive deletion

### Automatic archive log data set deletion

You can use a DASD or tape management system to delete archive log data sets automatically. The retention period for MQSeries archive log data sets is specified by the retention period field ARCRETN in the CSQ6ARVP installation macro (see the *MQSeries for OS/390 System Setup Guide* for more information). This value is passed to the management system in the JCL parameter RETPD.

The default for the retention period specifies that archive logs are to be kept for 9999 days (the maximum possible). **You can change the retention period but you must ensure that you can accommodate the number of backup cycles that you have planned for.**

MQSeries uses the value as the value for the JCL parameter RETPD when archive log data sets are created.

The retention period set by MVS/DFP™'s storage management subsystem (SMS) can be overridden by this MQSeries parameter. Typically, the retention period is set to the smaller value specified by either MQSeries or SMS. The storage administrator and MQSeries administrator must agree on a retention period value that is appropriate for MQSeries.

**Note:** Because some tape management systems provide external manual overrides of retention periods, MQSeries does not have an automated method to delete information about archive log data sets from the BSDS. Therefore, information about an archive log data set can still be in the BSDS long after the data-set retention period has expired and the data set has been scratched by the tape management system. Conversely, the maximum number of archive log data sets might have been exceeded and the data from the BSDS might have been dropped before the data set has reached its expiration date.

If archive log data sets are deleted automatically, remember that the operation does not update the list of archive logs in the BSDS. You can update the BSDS with the change log inventory utility, as described in “Changing the BSDS” on page 99. The update is not essential. Recording old archive logs wastes space in the BSDS, but does no other harm.

## Manually deleting archive log data sets

You must keep all the log records as far back as the lowest RBA identified in messages CSQI024I and CSQI025I. This RBA is obtained using the DISPLAY USAGE command as issued when creating a point of recovery using “Method 1: Full backup” on page 111. **You should read “Creating a point of recovery” on page 111 before discarding any logs.**

### Locate and discard archive log data sets

Having established the minimum log RBA required for recovery from your page set backup cycles, you can find archive log data sets that contain only earlier log records by performing the following procedure:

1. Use the print log map utility to print the contents of the BSDS. For an example of the output, see “Chapter 19. The print log map utility (CSQJU004)” on page 219.
2. Find the sections of the output titled “ARCHIVE LOG COPY n DATA SETS”. If you use dual logging, there are two sections. The columns labeled STARTRBA and ENDRBA show the range of RBAs contained in each volume. Find the volumes whose ranges include the minimum RBA you found with messages CSQI024I and CSQI025I. These are the earliest volumes you need to keep. If you are using dual-logging, there are two such volumes.

If no volumes have an appropriate range, one of these cases applies:

- The minimum RBA has not yet been archived, and you can discard all archive log volumes.
- The list of archive log volumes in the BSDS wrapped around when the number of volumes exceeded the number allowed by the MAXARCH parameter of the CSQ6LOGP macro. If the BSDS does not register an archive log volume, that volume cannot be used for recovery. Therefore, you should consider adding information about existing volumes to the BSDS. For instructions, see “Changes for archive logs” on page 100.

You should also consider increasing the value of MAXARCH. For information, see the *MQSeries for OS/390 System Setup Guide*.

3. Delete any archive log data set or volume whose ENDRBA value is less than the STARTRBA value of the earliest volume you want to keep. If you are using dual logging, delete both such copies.

Because BSDS entries wrap around, the first few entries in the BSDS archive log section might be more recent than the entries at the bottom. Look at the combination of date and time and compare their ages. Do not assume that you can discard all entries *above* the entry for the archive log containing the minimum LOGRBA.

Delete the data sets. If the archives are on tape, erase the tapes. If they are on DASD, run an OS/390 utility to delete each data set. Then, if you want the BSDS to list only existing archive volumes, use the change log inventory utility (CSQJU003) to delete entries for the discarded volumes. See “Changes for archive logs” on page 100 for an example.

## Discarding archive logs



---

## Chapter 9. Managing the BSDS

This chapter describes the tasks involved in managing the bootstrap data set. It contains these sections:

- “Finding out what the BSDS contains”
- “Changing the BSDS” on page 99
- “Recovering the BSDS” on page 102

---

### Finding out what the BSDS contains

The print log map utility (CSQJU004) is a batch utility that lists the information stored in the BSDS. For instructions on running it, see “Chapter 19. The print log map utility (CSQJU004)” on page 219.

### Time stamps in the BSDS

The output of the print log map utility shows the time stamps, which are used to record the date and time of various system events, that are stored in the BSDS.

The following time stamps are included in the header section of the report:

#### SYSTEM TIMESTAMP

Reflects the date and time the BSDS was last updated. The BSDS time stamp can be updated when:

- MQSeries starts.
- The write threshold is reached during log write activities. Depending on the number of output buffers you have specified and the system activity rate, the BSDS can be updated several times a second, or could not be updated for several seconds, minutes, or even hours. For details of the write threshold, see the WRTHRSR parameter of the CSQ6LOGP macro in the *MQSeries for OS/390 System Setup Guide*.
- MQSeries drops into a single BSDS mode from its normal dual BSDS mode due to an error. This can occur when a request to get, insert, point to, update, or delete a BSDS record is unsuccessful. When this error occurs, MQSeries updates the time stamp in the remaining BSDS to force a time stamp mismatch with the disabled BSDS.

#### UTILITY TIMESTAMP

The date and time the contents of the BSDS were altered by the change log inventory utility (CSQJU003).

The following time stamps are included in the active and archive log data sets portion of the report:

#### Active log date

The date the active log entry was created in the BSDS, that is, when the CSQJU003 NEWLOG was done.

#### Active log time

The time the active log entry was created in the BSDS, that is, when the CSQJU003 NEWLOG was done.

#### Archive log date

The date the archive log entry was created in the BSDS, that is, when the CSQJU003 NEWLOG was done or the archive itself was done.

## Finding what the BSDS contains

### Archive log time

The time the archive log entry was created in the BSDS, that is, when the CSQJU003 NEWLOG was done or the archive itself was done.

## Active log data set status

The BSDS records the status of an active log data set as one of the following:

**NEW** The data set has been defined but never used by MQSeries, or the log was truncated to a point before the data set was first used. In either case, the data set starting and ending RBA values are reset to zero.

### REUSABLE

Either the data set has been defined but never used by MQSeries, or the data set has been off-loaded. In the print log map output, the start RBA value for the last REUSABLE data set is equal to the start RBA value of the last archive log data set.

### NOT REUSABLE

The data set contains records that have not been off-loaded.

### STOPPED

The off-load processor encountered an error while reading a record, and that record could not be obtained from the other copy of the active log.

### TRUNCATED

Either:

- An I/O error occurred, and MQSeries has stopped writing to this data set. The active log data set is off-loaded, beginning with the starting RBA and continuing up to the last valid record segment in the truncated active log data set. The RBA of the last valid record segment is lower than the ending RBA of the active log data set. Logging is switched to the next available active log data set, and continues uninterrupted.

or

- An ARCHIVE LOG function has been called, which has truncated the active log.

The status appears in the output from the print log map utility.

## Changing the BSDS

You do not have to take special steps to keep the BSDS updated with records of logging events because MQSeries does that automatically. However, you might want to change the BSDS if you do any of the following:

- Add more active log data sets.
- Copy active log data sets to newly allocated data sets, for example, when providing larger active log allocations.
- Move log data sets to other devices.
- Recover a damaged BSDS.
- Discard outdated archive log data sets.

You can change the BSDS by running the change log inventory utility (CSQJU003). This utility can be run whether MQSeries is active or inactive. However, you are recommended **not** to run it when MQSeries is active, or you might get inconsistent results. The action of the utility is controlled by statements in the SYSIN data set. This section shows several examples. For complete instructions, see “Chapter 18. The change log inventory utility (CSQJU003)” on page 209.

You can copy an active log data set only when MQSeries is inactive because MQSeries allocates the active log data sets as exclusive (DISP=OLD) at MQSeries startup.

### Changes for active logs

You can add to, delete from, and record entries in the BSDS for active logs using the change log utility. Examples only are shown here; replace the data set names shown with the ones you want to use. For more details of the utility, see “Chapter 18. The change log inventory utility (CSQJU003)” on page 209.

#### Adding record entries to the BSDS

If an active log has been flagged as “stopped”, it is not reused for logging; however, it continues to be used for reading. Use the access method services to define new active log data sets, then use the change log inventory utility to register the new data sets in the BSDS. For example, use:

```
NEWLOG DSNAME=MQM111.LOGCOPY1.DS10,COPY1
NEWLOG DSNAME=MQM111.LOGCOPY2.DS10,COPY2
```

If you are copying the contents of an old active log data set to the new one, you can also give the RBA range and the starting and ending time stamps on the NEWLOG function.

## Changing the BSDS

### Deleting information about the active log data set from the BSDS

To delete information about an active log data set from the BSDS, you could use:

```
DELETE DSNAME=MQM111.LOGCOPY1.DS99
DELETE DSNAME=MQM111.LOGCOPY2.DS99
```

### Recording information about the log data set in the BSDS

To record information about an existing active log data set in the BSDS, use:

```
NEWLOG DSNAME=MQM111.LOGCOPY1.DS10,COPY2,STARTIME=19930212205198,
        ENDTIME=19930412205200,STARTRBA=6400,ENDRBA=94FF
```

Inserting a record containing this type of information in the BSDS might be necessary because:

- The entry for the data set has been deleted, but is needed again.
- You are copying the contents of one active log data set to another data set.
- You are recovering the BSDS from a backup copy.

### Enlarging the active log

This procedure must only be used when MQSeries is inactive:

1. Stop MQSeries. This step is required because MQSeries allocates all active log data sets for its exclusive use when it is active.
2. Use Access Method Services ALTER with the NEWNAME option to rename your active log data sets.
3. Use Access Method Services DEFINE to define larger active log data sets.  
By reusing the old data set names, you do not have to run the change log inventory utility to establish new names in the BSDSs. The old data set names and the correct RBA ranges are already in the BSDSs.
4. Use Access Method Services REPRO to copy the old (renamed) data sets into their respective new data sets.
5. Start MQSeries.

Although it is not necessary for all log data sets to be the same size, it is operationally more consistent and efficient. If the log data sets are not the same size, it is more difficult to track your system's logs, and so space can be wasted.

## Changes for archive logs

You can add to, delete from, and change the password of entries in the BSDS for archive logs. Examples only are shown here; you must replace the data set names shown with the ones you want to use. For more details of the utility, see "Chapter 18. The change log inventory utility (CSQJU003)" on page 209.

### Adding an archive log

When the recovery of an object depends on reading an existing archive log data set, the BSDS must contain information about that data set, so that MQSeries can find it. To register information about an existing archive log data set in the BSDS, use:

```
NEWLOG DSNAME=CSQARCL.ARCHLOG1.E00021.T2205197.A0000015,COPY1VOL=CSQV04,
        UNIT=TAPE,STARTRBA=3A190000,ENDRBA=3A1F0FFF,CATALOG=NO
```

### Deleting an archive log

To delete an entire archive log data set on one or more volumes, use:

```
DELETE DSNNAME=CSQARC1.ARCHLOG1.E00021.T2205197.A0000015,COPY1VOL=CSQV04
```

### Changing the password of an archive log

If you change the password of an existing archive log data set, you must also change the information in the BSDS.

1. List the BSDS, using the print log map utility.
2. Delete the entry for the archive log data set with the changed password, using the DELETE function of the CSQJU003 utility (see page 209).
3. Name the same data set as a new archive log data set. Use the NEWLOG function of the CSQJU003 utility (see page 209), and give the new password, the starting and ending RBAs, and the volume serial numbers (which can be found in the print log map utility output, see page 219).

To change the password for new archive log data sets, use:

```
ARCHIVE PASSWORD=password
```

To stop placing passwords on new archive log data sets, use:

```
ARCHIVE NOPASSWD
```

**Note:** You should only use the ARCHIVE utility function if you do not have an external security manager.

---

### Recovering the BSDS

If MQSeries is operating in dual BSDS mode and one BSDS becomes damaged, forcing MQSeries into single BSDS mode, MQSeries continues to operate without a problem (until the next restart). To return the environment to dual BSDS mode:

1. Use Access Method Services to rename or delete the damaged BSDS and to define a new BSDS with the same name as the damaged BSDS. Control statements can be found in job CSQ4BSDS in thlqual.SCSQPROC (described in the *MQSeries for OS/390 System Setup Guide*).
2. Issue the MQSeries command +CSQ1 RECOVER BSDS to make a copy of the valid BSDS in the newly allocated data set and to reinstate dual BSDS mode.

If MQSeries is operating in single BSDS mode and the BSDS is damaged, or if MQSeries is operating in dual BSDS mode and both BSDSs are damaged, MQSeries stops and does not restart until the BSDS data sets are repaired. In this case:

1. Locate the BSDS associated with the most recent archive log data set. The data set name of the most recent archive log appears on the job log in the last occurrence of message CSQJ003I, which indicates that off-loading has been completed successfully. In preparation for the rest of this procedure, it is a good practice to keep a log of all successful archives noted by that message:
  - If archive logs are on DASD, the BSDS is allocated on any available DASD. The BSDS name is like the corresponding archive log data set name; change only the first letter of the last qualifier, from A to B, as in the example below:  
**Archive log name**  
CSQ.ARCHLOG1.A0000001  
**BSDS copy name**  
CSQ.ARCHLOG1.B0000001
  - If archive logs are on tape, the BSDS is the first data set of the first archive log volume. The BSDS is not repeated on later volumes.
2. If the most recent archive log data set has no copy of the BSDS (presumably because an error occurred when off-loading it), then locate an earlier copy of the BSDS from an earlier off-load.
3. Rename *damaged* BSDSs using the Access Method Services ALTER command with the NEWNAME option. If you decide to delete a damaged BSDS, use the Access Method Services DELETE command. For each damaged BSDS, use Access Method Services to define a new BSDS as a replacement data set. Job CSQ4BSDS in thlqual.SCSQPROC contains Access Method Services control statements to define a new BSDS.
4. Use Access Method Services to delete or rename the damaged data set, to define a replacement data set, and to copy the remaining BSDS to the replacement with the Access Method Services REPRO command.
5. Use the Access Method Services REPRO command to copy the BSDS from the archive log to one of the replacement BSDSs you defined in step 3. Do not copy any data to the second replacement BSDS—you do that in step 6 on page 104.
  - a. Print the contents of the replacement BSDS.

Use the print log map utility (CSQJU004) to print the contents of the replacement BSDS. This enables you to review the contents of the replacement BSDS before continuing your recovery work.
  - b. Update the archive log data set inventory in the replacement BSDS.

Examine the output from the print log map utility and check that the replacement BSDS does not contain a record of the archive log from which the BSDS was copied. If the replacement BSDS is an old copy, its inventory might not contain all archive log data sets that were created more recently.

Therefore, the BSDS inventory of the archive log data sets must be updated to reflect the current subsystem inventory.

Use the change log inventory utility (CSQJU003) NEWLOG statement to update the replacement BSDS, adding a record of the archive log from which the BSDS was copied. If the archive log data set is password-protected, be certain to use the PASSWORD option of the NEWLOG function. Also, make certain the CATALOG option of the NEWLOG function is properly set to CATALOG=YES if the archive log data set is cataloged. Use the NEWLOG statement to add any additional archive log data sets that were created later than the BSDS copy.

- c. Update passwords in the replacement BSDS.

The BSDS contains passwords for the archive log data sets and for the active log data sets. To ensure that the passwords in the replacement BSDS reflect the current passwords used by your installation, use the change log inventory ARCHIVE utility function with the PASSWORD option.

- d. Update the active log data set inventory in the replacement BSDS.

In unusual circumstances, your installation could have added, deleted, or renamed active log data sets since the BSDS was copied. In this case, the replacement BSDS does not reflect the actual number or names of the active log data sets your installation has currently in use.

If you need to delete an active log data set from the replacement BSDS log inventory, use the change log inventory utility DELETE function.

If you need to add an active log data set to the replacement BSDS log inventory, use the change log inventory utility NEWLOG function. Ensure that the RBA range is specified correctly on the NEWLOG function. If the active log data set is password-protected, be sure to use the PASSWORD option.

If you need to rename an active log data set in the replacement BSDS log inventory, use the change log inventory utility DELETE function, followed by the NEWLOG function. Be sure that the RBA range is specified correctly on the NEWLOG function. If the active log data set is password-protected, be certain to use the PASSWORD option.

- e. Update the active log RBA ranges in the replacement BSDS.

Later, when MQSeries restarts, it compares the RBAs of the active log data sets listed in the BSDS with the RBAs found in the actual active log data sets. If the RBAs do not agree, MQSeries does not restart. The problem is magnified when a particularly old copy of the BSDS is used. To solve this problem, you can use the change log inventory utility (CSQJU003) to adjust the RBAs found in the BSDS using the RBAs in the actual active log data sets. This can be done by:

- Using the print log records utility (CSQ1LOGP) to print a summary report of the active log data set. This shows the starting and ending RBAs.
- Comparing the actual RBA ranges with the RBA ranges you have just printed, when the RBAs of all active log data sets are known.

If the RBA ranges are equal for all active log data sets, you can proceed to the next recovery step without any additional work.

If the RBA ranges are not equal, then the values in the BSDS must be adjusted to reflect the actual values. For each active log data set that needs to have the RBA range adjusted, use the change log inventory utility DELETE function to delete the active log data set from the inventory in the replacement BSDS. Then use the NEWLOG function to

## Recovering the BSDS

redefine the active log data set to the BSDS. If the active log data sets are password-protected, be certain to use the PASSWORD option of the NEWLOG function.

- f. If only two active log data sets are specified for each copy of the active log, MQSeries can have difficulty during restart. The problem can arise when one of the active log data sets is full and has not been off-loaded, while the second active log data set is close to filling. In this case, add a new active log data set for each copy of the active log and define each new active log data set in the replacement BSDS log inventory.

Use the Access Method Services DEFINE command to define a new active log data set for each copy of the active log. The control statements to accomplish this task can be found in job CSQ4BSDS in thlqual.SCSQPROC. Once the active log data sets are physically defined and allocated, use the change log inventory utility NEWLOG function to define the new active log data sets in the replacement BSDS. The RBA ranges need not be specified on the NEWLOG statement; however, if the active log data sets are password-protected, be certain to use the PASSWORD option of the NEWLOG function.

6. Copy the updated BSDS to the second new BSDS data set. The BSDSs are now identical.

Consider using the print log map utility (CSQJU004) to print the contents of the second replacement BSDS at this point.

7. See "Active log problems" on page 151 for information about what to do if you have lost your current active log data set.
8. Restart MQSeries, using the newly constructed BSDS. MQSeries determines the current RBA and what active logs need to be archived.



---

## Chapter 10. Managing page sets

This chapter describes how to create, copy, and generally manage the page sets associated with a queue manager. It contains these sections:

- “How to add a page set to a queue manager”
- “What to do when one of your page sets becomes full” on page 106
- “How to balance loads on page sets” on page 107
- “How to expand a page set” on page 109
- “How to reduce a page set” on page 110
- “How to back up and recover page sets” on page 111
- “How to back up and restore queues using CSQUTIL” on page 115

See the *MQSeries for OS/390 Concepts and Planning Guide* for a description of page sets, storage classes, buffers, and buffer pools, and some of the performance considerations that apply.

---

### How to add a page set to a queue manager

This description assumes that you have an MQSeries subsystem that is already running. You might need to add a page set if, for example, your MQSeries subsystem has to cope with new applications using new queues.

To add a new page set, use the following procedure:

1. Stop the queue manager by issuing a STOP QMGR command.
2. Define and format the new page set. You can use the sample JCL in thlqual.SCSQPROC(CSQ4PAGE) as a basis. For more information, see “Formatting page sets (FORMAT)” on page 183.  
Take care not to format any page sets that are in use, unless this is what you intend. If so, use the FORCE option of the FORMAT utility function.
3. Add the new page set to the startup procedure for your MQSeries subsystem.
4. Add a definition for the new page set to your CSQINP1 initialization data set. Use the DEFINE PSID command to associate the page set with a buffer pool.
5. Add the appropriate storage class definitions for your page set to your CSQINP2 initialization data set concatenation. This step is optional but recommended, see the *MQSeries for OS/390 System Setup Guide*.
6. Restart the queue manager.

---

### What to do when one of your page sets becomes full

You can find out about the utilization of page sets by using the MQSeries command `DISPLAY USAGE`. For example, the command:

```
DISPLAY USAGE PSID(03)
```

displays the current state of the page set 03. This tells you how many free pages this page set has.

If you have defined secondary extents for your page sets, they will be dynamically expanded each time they fill up. Eventually, all secondary extents will be used, or no further disk space will be available. If this happens, an application will receive return code `MQRC_STORAGE_MEDIUM_FULL`.

If an application receives a return code of `MQRC_STORAGE_MEDIUM_FULL` from an MQI call, this is a clear indication that there is not enough space left on the page set. If the problem persists or is likely to reoccur, you must do something to solve it.

You can approach this problem in two ways:

1. Balance the load between page sets by moving queues from one page set to another.
2. Expand the page set.

## How to balance loads on page sets

Load balancing on page sets means moving the messages associated with one or more queues from one page set to another, less utilized page set. You should use this technique if it is not practical to expand the page set.

To identify which queues are using a page set, use the appropriate MQSeries commands. For example, to find out which queues are mapped to page set 02, first, find out which storage classes map to page set 02, by using this command:

```
DISPLAY STGCLASS(*) PSID(02)
```

Then use this command:

```
DISPLAY QUEUE(*) TYPE(QLOCAL) STGCLASS
```

to find out which queues use which storage class.

## Moving a non-shared queue

To move queues and their messages from one page set to another, use the MOVE QLOCAL command (described in the *MQSeries MQSC Command Reference*). When you have identified the queue or queues that you want to move to a new page set, follow this procedure for each of these queues:

1. Ensure that the queue to be moved is not in use by any applications (that is, the queue attributes IPPROCS and OPPROCS are zero) and that it has no uncommitted messages (the UNCOM value from the DISPLAY QSTATUS command is NO).

**Note:** The only way to ensure that this state continues is to change your security settings temporarily. If you cannot do this, later stages in this procedure might fail if applications start to use the queue despite the precautionary steps such as setting PUT(DISABLED). However, messages can never be lost by this procedure.

2. Prevent applications from putting messages on the queue being moved by altering the queue definition to disable MQPUTs. Change the queue definition to PUT(DISABLED).
3. Define a temporary queue with the same attributes as the queue that is being moved:

```
DEFINE QL(TEMP_QUEUE) LIKE(Queue_TO_MOVE) PUT(ENABLED) GET(ENABLED)
```

**Note:** If this temporary queue already exists from a previous run, delete it before doing the define.

4. Move the messages to the temporary queue by issuing the following command:  
MOVE QLOCAL(Queue\_TO\_MOVE) TOQLOCAL(TEMP\_QUEUE)
5. Delete the queue you are moving, using the command:  
DELETE QLOCAL(Queue\_TO\_MOVE)

## Balancing loads

6. Define a new storage class which maps to the required page set, for example:  
`DEFINE STGCLASS(NEW) PSID(nn)`  
  
Add the new storage class definition to CSQINP2 ready for the next MQSeries subsystem restart.
7. Redefine the queue that is being moved, changing the storage class attribute. When the queue is redefined, it is based on the temporary queue created in step 3:  
`DEFINE QL(Queue_To_Move) LIKE(TEMP_QUEUE) STGCLASS(NEW)`
8. Move the messages back to the new queue using the command:  
`MOVE QLOCAL(TEMP) TOQLOCAL(Queue_To_Move)`
9. The queue created in step 3 is no longer required. Use the following command to delete it:  
`DELETE QL(TEMP_QUEUE)`
10. If the queue being moved was defined in the CSQINP2 concatenation, change the STGCLASS attribute of the appropriate `DEFINE QLOCAL` command in the CSQINP2 concatenation. Add the `REPLACE` keyword so that the existing queue definition is replaced.

Figure 47 shows an extract from a load balancing job.

```
//UTILITY EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD DSN=th1qua1.SCSQANLE,DISP=SHR
// DD DSN=th1qua1.SCSQAUTH,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
COMMAND DDNAME(MOVEQ)
/*
//MOVEQ DD *
ALTER QL(Queue_To_Move) PUT(DISABLED)
DELETE QL(TEMP_QUEUE) PURGE
DEFINE QL(TEMP_QUEUE) LIKE(Queue_To_Move) PUT(ENABLED) GET(ENABLED)
MOVE QLOCAL(Queue_To_Move) TOQLOCAL(TEMP_QUEUE)
DELETE QL(Queue_To_Move)
DEFINE STGCLASS(NEW) PSID(2)
DEFINE QL(Queue_To_Move) LIKE(TEMP_QUEUE) STGCLASS(NEW)
MOVE QLOCAL(TEMP_QUEUE) TOQLOCAL(Queue_To_Move)
DELETE QL(TEMP_QUEUE)
/*
```

Figure 47. Extract from a load balancing job for a page set

## How to expand a page set

You expand a page set by creating a new, larger page set and copying the messages from the old page set to the new one. You then have to ensure that the new page set is used when you restart the queue manager.

### Note

This technique involves stopping and restarting the queue manager. This will result in any nonpersistent messages that are not on shared queues being deleted at restart time. If you have nonpersistent messages that you do not want to be deleted, consider load balancing (see “How to balance loads on page sets” on page 107).

In this description, the page set that you want to expand is referred to as the *source* page set; the new, larger page set is referred to as the *destination* page set.

Follow these steps:

1. Stop the queue manager.
2. Define the destination page set, ensuring that it is larger than the source page set, with a larger secondary extent value.
3. Use the FORMAT function of CSQUTIL to format the destination page set. See “Formatting page sets (FORMAT)” on page 183 for more details.
4. Use the COPYPAGE function of CSQUTIL to copy all the messages from the source page set to the destination page set. See “Expanding a page set (COPYPAGE)” on page 185 for more details.
5. Restart the queue manager using the destination page set by doing one of the following:
  - Change the MQSeries startup procedure to reference the destination page set. See the *MQSeries for OS/390 System Setup Guide* for more details.
  - Use Access Method Services to delete the source page set and then rename the destination page set, giving it the same name as that of the source page set.

**Attention:** Before you delete any MQSeries page set, be sure that you have made the required backup copies.

### How to reduce a page set

If you have a large page set that is mostly empty (as shown by the DISPLAY USAGE command), you might want to reduce its size. The procedure to do this involves using the COPY, EMPTY, RESETPAGE, FORMAT, LOAD, and COMMAND functions of CSQUTIL (see “Chapter 17. MQSeries utility program (CSQUTIL)” on page 179). This procedure will not work for page set zero; it is not practical to reduce the size of this page set.

1. Prevent all users, other than the MQSeries administrator, from using the queue manager. This could be done by changing the access security settings for example.
2. Wait until all queue manager use has ended; you might have to stop and restart the queue manager to achieve this. (Remember that if you do this, you will lose all your nonpersistent messages.)
3. Run the COPY function of CSQUTIL with the PSID option to copy all message data from the large page set and save them in a sequential data set.
4. Run the EMPTY function of CSQUTIL with the PSID option to remove all messages from the page set. Use the DISPLAY USAGE PSID(n) command to verify that the page set is totally empty.
5. Use the DISPLAY STGCLASS(\*) PSID(n) command to identify all storage classes that relate to the page set that is to be reduced in size.
6. Use the DISPLAY QUEUE(\*) TYPE(QLOCAL) QSGDISP(PRIVATE) STGCLASS command to identify all queues that use any of the storage classes identified in step 5.
7. Alter each queue that you have identified to use a different storage class that maps to a different page set. This does not have to be a permanent change to the queue, but is essential for the queue manager to be able to restart. If you do not do this, you could get 00C91B01 abends when the queue manager attempts to start.  
Use the ALTER QLOCAL(q-name) STGCLASS(stgcl-name) command to alter the storage class attribute of each queue.
8. Use the STOP QMGR command with the QUIESCE or FORCE attribute to stop the queue manager.
9. Run the RESETPAGE function of CSQUTIL against all page sets other than the page set that is to be reduced in size. (You can choose to reset the page set in place, or you can copy and reset the page set.)
10. Define a new smaller page set data set to replace the large page set. Run the FORMAT function of CSQUTIL against it.
11. Define new log data sets (BSDS and active logs) with new data set names.
12. Restart the queue manager using the page sets created in steps 9 and 10 and the new BSDS and log data sets created in step 11.
13. Use the ALTER QLOCAL(q-name) STGCLASS(stgcl-name) command to reset the storage class attribute to the previous value for each queue altered in step 7.
14. Run the LOAD function of CSQUTIL to load back all the messages saved during step 3.
15. Allow all users access to the queue manager.
16. You can now delete the old large page set and the old BSDS and log data sets.

## How to back up and recover page sets

This section describes:

- “Creating a point of recovery”
- “Recovering page sets” on page 113

### Creating a point of recovery

MQSeries can recover objects and persistent messages to their current state only if there is:

1. A copy of all page sets from an earlier point.
2. All the MQSeries logs since that point.

These represent a point of recovery.

Both objects and messages are held on page sets. Multiple objects and messages from different queues can exist on the same page set. Therefore, for recovery purposes, objects and messages cannot be backed up in isolation so that a page set must be backed up as a whole to ensure the proper recovery of the data.

The MQSeries recovery log contains a record of all persistent messages and changes made to objects. If MQSeries fails (for example, due to an I/O error on a page set), you can recover the page set by restoring the backup copy and restarting MQSeries. MQSeries applies the log changes to the page set from the point of the backup copy.

There are two ways of creating a point of recovery. The first involves stopping the queue manager thereby forcing all updates on to the page sets. The second involves taking ‘fuzzy’ backup copies of the page sets without stopping the queue manager.

The first will allow you to restart from the point of recovery, using only the backed up page set data sets and the logs from that point on.

If you use the second method, and your associated logs subsequently become damaged or lost you will not be able to use the fuzzy page set backup copies to recover. This is because the fuzzy page set backup copies contain an inconsistent view of the state of MQSeries and are dependent on the logs being available. If the logs are not available, you will have to return to the last set of backup page set copies taken while the subsystem was inactive (Method 1 below) and accept the loss of data from that time.

#### Method 1: Full backup

This method involves shutting MQSeries down. This forces all updates on to the page sets so that the page sets are in a consistent state.

1. Stop all MQSeries applications using the queue manager (allowing them to complete first). This could be done by changing the access security or queue settings, for example.
2. When all activity has completed, display and resolve any in-doubt units of recovery in the subsystem. (Use the MQSeries DISPLAY THREAD and RESOLVE INDOUBT commands as described in the *MQSeries MQSC Command Reference* manual.)

This will bring the page sets to a consistent state; if you do not do this, or can’t, your page sets might not be consistent, and you are effectively doing a “fuzzy” backup.

## Page set backup

3. Issue the MQSeries command ARCHIVE LOG to ensure that the latest log data is written out to the log data sets.
4. Issue the MQSeries command STOP QMGR MODE(QUIESCE). Record the lowest RBA value in the CSQI024I or CSQI025I messages (see the *MQSeries for OS/390 Messages and Codes* manual for information about these messages). You should keep the log data sets starting from the one indicated by the RBA value up to the current one.
5. Take backup copies of the page sets (see “Backing up page sets”).

### Method 2: Fuzzy backup

This method does not involve shutting MQSeries down. Therefore, updates might be in virtual storage buffers during the backup process. This means that the page sets are not in a consistent state, and can only be used for recovery in conjunction with the logs.

1. Issue the MQSeries command DISPLAY USAGE and record the RBA value in the CSQI024I or CSQI025I message (see the *MQSeries for OS/390 Messages and Codes* manual for information about these messages).
2. Take backup copies of the page sets (see “Backing up page sets”).
3. Issue the MQSeries command ARCHIVE LOG to ensure that the latest log data is written out to the log data sets. In order to restart from the point of recovery, you must keep copies of all the log data sets from that containing the recorded RBA value to that created by the ARCHIVE LOG command.

### Backing up page sets

You can take a backup of your page sets in two ways:

- “Using Access Method Services”
- “Using volume dump and restore” on page 113.

To recover a page set, MQSeries needs to know how far back in the log to go. MQSeries maintains a log RBA number in page 0 of each page set, called the recovery log sequence number (LSN). This number is the starting RBA in the log from which MQSeries can recover the page set. When you back up a page set, this number is also copied.

If the copy is later used to recover the page set, MQSeries must have access to all of the log records from this RBA value to the current RBA. That means you must keep enough of the log records to enable MQSeries to recover from the oldest backup copy of a page set you intend to keep.

**Using Access Method Services:** You can use Access Method Services REPRO function (or any equivalent) to make copies of your page sets. You can do this whether or not MQSeries is running. If you want to do it while MQSeries is running, you must DEFINE the page sets with SHAREOPTIONS(2,3).

If you copy the page set while MQSeries is running you *must* use a copy utility that copies page 0 of the page set first – if you do not do this you could corrupt the data in your page set. Access Method Services REPRO meets this requirement.

If the process of dynamically expanding a page set is interrupted, for example by power to the system being lost, you can still use Access Method Services REPRO to take a backup of a page set.



If you perform an Access Method Services IDCAMS LISTCAT ENT('page set data set name') ALLOC, you will see that the HI-ALLOC-RBA will be higher than the HI-USED-RBA. Access Method Services REPRO will just copy the page set up to the high used RBA, and not give an error.

The next time this page set fills up it will be extended again, if possible, and the pages between the high used RBA and the highest allocated RBA will be used, along with another new extent.

For more information on the REPRO statement, see the *DFSMS/MVS Access Method Services for VSAM* or the *DFSMS/MVS Access Method Services for the Integrated Catalog Facility* manual.

**Using volume dump and restore:** Volume dump services dumps all the data sets on the volume. Likewise, restore volume services can (depending on the type of service) restore all the data sets.

### Backing up your object definitions

You should also back up copies of your object definitions. To do this, use the MAKEDEF feature of the CSQUTIL COMMAND function (described in “Issuing commands to MQSeries (COMMAND)” on page 190).

You should do this whenever you take a backup copy of your queue manager, and keep the most current version.

## Recovering page sets

If MQSeries has suffered a failure that has caused it to terminate, MQSeries can normally be restarted with all recovery being performed during restart. However, such recovery is not possible if any of your page sets or log data sets are not available. The extent to which you can now recover depends on the availability of backup copies of page sets and log data sets.

To restart from a point of recovery you must have:

- A backup copy of the page set that is to be recovered.
- If you used “fuzzy” backup, the log data set that included the recorded RBA value, the log data set that was made by the ARCHIVE LOG command, and all the log data sets between these.
- If you used full backup, but you do not have the log data set following that made by the ARCHIVE LOG command, you will need to use the RESETPAGE function of the CSQUTIL utility. The RBA identified using either method in “Creating a point of recovery” on page 111 is the restart point for the backed-up page sets.

To recover a page set to its current state, you must also have all the log data sets and records since the ARCHIVE LOG command.

There are two methods for recovering a page set. To use either method, the queue manager must not be running.

### Simple recovery

This is the simpler method, and is appropriate for most recovery situations.

1. Delete and redefine the page set you want to restore from backup.
2. Use Access Method Services REPRO to copy the backup copy of the page set into the new page set. You should define your new page set with a secondary extent value so that it can be expanded dynamically.

## Page set recovery

Alternatively, you can rename your backup copy to the original name, or change the CSQP00xx DD statement in your queue manager procedure to point to your backup page set. However, if you then lose or corrupt the page set, you will no longer have a backup copy to restore from.

3. Restart the queue manager.
4. When the queue manager has restarted successfully, you can restart your applications
5. Reinststate your normal backup procedures for the restored page.

### Advanced recovery

This method provides performance advantages if you have a large page set to recover, or if there has been a lot of activity on the page set since the last backup copy was taken. However, it requires more manual intervention than the simple method, which might increase the risk of error and the time taken to perform the recovery.

1. Delete and redefine the page set you want to restore from backup.
2. Use Access Method Services REPRO to copy the backup copy of the page set into the new page set. You should define your new page set with a secondary extent value so that it can be expanded dynamically.  
  
Alternatively, you can rename your backup copy to the original name, or change the CSQP00xx DD statement in your queue manager procedure to point to your backup page set. However, if you then lose or corrupt the page set, you will no longer have a backup copy to restore from.
3. Change the CSQINP1 definitions for your queue manager to make the buffer pool associated with the page set being recovered as large as possible. By making the buffer pool this large, you might be able to keep all of the changed pages resident in the buffer pool and reduce the amount of I/O to the page set.
4. Restart the queue manager.
5. When the queue manager has restarted successfully, stop it (using quiesce) and then restart it using the normal buffer pool definition for that page set. After this second restart completes successfully, you can restart your applications
6. Reinststate your normal backup procedures for the restored page.

### What happens when MQSeries is restarted

When MQSeries is restarted, it applies all changes made to the page set that are registered in the log, beginning at the restart point for the page set. MQSeries can recover multiple page sets in this way. The page set will be dynamically expanded, if required, during media recovery.

During restart MQSeries determines the log RBA to start from by taking the lowest value from the following:

- Recovery LSN from the checkpoint log record for each page set.
- Recovery LSN from page 0 in each page set.
- The RBA of the oldest incomplete unit of recovery in the system at the time the backup was taken.

All object definitions are stored on page set zero. Messages can be stored on any available page set.

**Note:** MQSeries cannot restart if page set zero is not available.

## How to back up and restore queues using CSQUTIL

You can use the CSQUTIL utility functions for backing up and restoring queues. To back up a queue, use the COPY or SCOPY function to copy the messages from a queue onto a data set. To restore the queue, use the complementary function LOAD. For more information, see “Chapter 17. MQSeries utility program (CSQUTIL)” on page 179.

|



---

## Chapter 11. Managing queue-sharing groups and shared queues

This chapter discusses the following topics:

- “Managing queue-sharing groups”
- “Managing shared queues” on page 118
- “Managing group objects” on page 122
- “Managing the Coupling Facility” on page 122

---

### Managing queue-sharing groups

This section describes the following tasks:

- “Adding a queue-sharing group to the DB2 tables”
- “Adding a queue manager to a queue-sharing group”
- “Removing a queue manager from a queue-sharing group”
- “Removing a queue-sharing group from the DB2 tables” on page 118

#### Adding a queue-sharing group to the DB2 tables

To add a queue-sharing group to the DB2 tables, use the ADD QSG function of the queue-sharing group utility (CSQ5PQSG). This program is described in “Chapter 21. The queue-sharing group utility (CSQ5PQSG)” on page 225. A sample is provided in thlqual.SCSQPROC(CSQ45AQS).

#### Adding a queue manager to a queue-sharing group

To add a queue manager to a queue-sharing group, use the ADD QMGR function of the queue-sharing group utility (CSQ5PQSG). This program is described in “Chapter 21. The queue-sharing group utility (CSQ5PQSG)” on page 225. A sample is provided in thlqual.SCSQPROC(CSQ45AQM).

The queue-sharing group must exist before you can add queue managers to it.

**Note:** A queue manager can only be a member of one queue-sharing group.

#### Removing a queue manager from a queue-sharing group

To remove a queue manager from a queue-sharing group, use the REMOVE QMGR function of the queue-sharing group utility (CSQ5PQSG). This program is described in “Chapter 21. The queue-sharing group utility (CSQ5PQSG)” on page 225. A sample is provided in thlqual.SCSQPROC(CSQ45RQM).

You can only remove a queue manager from a queue-sharing group if:

- The queue manager has never started as a member of the queue-sharing group, or it terminated normally the last time it was stopped.
- The queue manager is not active.

## Managing queue-sharing groups and shared queues

### Removing a queue-sharing group from the DB2 tables

To remove a queue-sharing group from the DB2 tables, use the REMOVE QSG function of the queue-sharing group utility (CSQ5PQSG). This program is described in “Chapter 21. The queue-sharing group utility (CSQ5PQSG)” on page 225. A sample is provided in thlqual.SCSQPROC(CSQ45RQS).

You can only remove a queue-sharing group from the common DB2 data-sharing group tables after you have removed all of the queue managers from the queue-sharing group (as described above).

When the queue-sharing group record is deleted from the queue-sharing group administration table, all objects and administrative information relating to that queue-sharing group are deleted from other MQSeries DB2 tables. This includes shared queue and group object information.

---

## Managing shared queues

This section describes the following tasks:

- “Recovering shared queues”
- “Moving shared queues”
- “Migrating non-shared queues to shared queues” on page 121

### Recovering shared queues

The definition of a shared queue is kept in a DB2 database and so can be recovered if necessary using standard DB2 database procedures.

Messages on a shared queue are stored in the Coupling Facility. They have no built-in recovery procedures; this means that they could be lost in the unlikely event of Coupling Facility or similar problems. This risk can be alleviated by periodically making back-up copies of the messages. Use the COPY function of the CSQUTIL utility program to do this.

### Moving shared queues

This section describes how to perform load balancing by moving a shared queue from one Coupling Facility structure to another. It also describes how to move a non-shared queue to a shared queue (and how to move a shared queue to a non-shared queue).

When you move a queue, you need to define a temporary queue as part of the procedure. This is because every queue must have a unique name so you can't have two queues of the same name, even if the queues have different queue dispositions. MQSeries tolerates having two queues with the same name (as in step 2 on page 121), but you cannot use the queues.

#### Moving a queue from one Coupling Facility structure to another

To move queues and their messages from one Coupling Facility structure to another, use the MOVE QLOCAL command (described in the *MQSeries MQSC Command Reference*). When you have identified the queue or queues that you want to move to a new Coupling Facility structure, use the following procedure to move each queue:

1. Ensure that the queue to be moved is not in use by any applications, that is, the queue attributes IPPROCS and OPPROCS are zero on all queue managers in the queue-sharing group.

## Managing queue-sharing groups and shared queues

2. Prevent applications from putting messages on the queue being moved by altering the queue definition to disable MQPUTs. Change the queue definition to PUT(DISABLED).
3. Define a temporary queue with the same attributes as the queue that is being moved:  

```
DEFINE QL(TEMP_QUEUE) LIKE(Queue_To_Move) PUT(ENABLED) GET(ENABLED) QSGDISP(QMGR)
```
- Note:** If this temporary queue already exists from a previous run, delete it before doing the define.
4. Move the messages to the temporary queue by issuing the following command:  

```
MOVE QLOCAL(Queue_To_Move) TOQLOCAL(TEMP_QUEUE)
```
5. Delete the queue you are moving, using the command:  

```
DELETE QLOCAL(Queue_To_Move)
```
6. Redefine the queue that is being moved, changing the CFSTRUCT attribute. When the queue is redefined, it is based on the temporary queue created in step 3:  

```
DEFINE QL(Queue_To_Move) LIKE(TEMP_QUEUE) CFSTRUCT(NEW) QSGDISP(SHARED)
```
7. Move the messages back to the new queue using the command:  

```
MOVE QLOCAL(TEMP) TOQLOCAL(Queue_To_Move)
```
8. The queue created in step 3 is no longer required. Use the following command to delete it:  

```
DELETE QL(TEMP_QUEUE)
```
9. If the queue being moved was defined in the CSQINP2 concatenation, change the CFSTRUCT attribute of the appropriate DEFINE QLOCAL command in the CSQINP2 concatenation. Add the REPLACE keyword so that the existing queue definition is replaced.

Figure 48 shows an extract from a load balancing job.

```
//UTILITY EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD DSN=thlqua1.SCSQANLE,DISP=SHR
// DD DSN=thlqua1.SCSQAUTH,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
COMMAND DDNAME(MOVEQ)
/*
//MOVEQ DD *
ALTER QL(Queue_To_Move) PUT(DISABLED)
DELETE QL(TEMP_QUEUE) PURGE
DEFINE QL(TEMP_QUEUE) LIKE(Queue_To_Move) PUT(ENABLED) GET(ENABLED) QSGDISP(QMGR)
MOVE QLOCAL(Queue_To_Move) TOQLOCAL(TEMP_QUEUE)
DELETE QL(Queue_To_Move)
DEFINE QL(Queue_To_Move) LIKE(TEMP_QUEUE) CFSTRUCT(NEW) QSGDISP(SHARED)
MOVE QLOCAL(TEMP_QUEUE) TOQLOCAL(Queue_To_Move)
DELETE QL(TEMP_QUEUE)
/*
```

Figure 48. Extract from a load balancing job for a Coupling Facility structure

## Managing queue-sharing groups and shared queues

### Moving a non-shared queue to a shared queue

To move a non-shared queue to a shared queue, combine the instructions for moving a shared queue with the instructions for moving a non-shared queue (described in “Moving a non-shared queue” on page 107) as appropriate. Figure 49 gives a sample job to do this.

**Note:** Remember that messages on shared queues are subject to certain restrictions on the maximum message size, message persistence, and queue index type, so you might not be able to move some non-shared queues to a shared queue.

```
//UTILITY EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD DSN=thlqua1.SCSQANLE,DISP=SHR
// DD DSN=thlqua1.SCSQAUTH,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
COMMAND DDNAME(MOVEQ)
/*
//MOVEQ DD *
ALTER QL(Queue_To_Move) PUT(DISABLED)
DELETE QL(TEMP_QUEUE) PURGE
DEFINE QL(TEMP_QUEUE) LIKE(Queue_To_Move) PUT(ENABLED) GET(ENABLED)
MOVE QLOCAL(Queue_To_Move) TOQLOCAL(TEMP_QUEUE)
DELETE QL(Queue_To_Move)
DEFINE QL(Queue_To_Move) LIKE(TEMP_QUEUE) CFSTRUCT(NEW) QSGDISP(SHARED)
MOVE QLOCAL(TEMP_QUEUE) TOQLOCAL(Queue_To_Move)
DELETE QL(TEMP_QUEUE)
/*
```

Figure 49. Sample job for moving a non-shared queue to a shared queue

### Moving a shared queue to a non-shared queue

To move a shared queue to a non-shared queue, combine the instructions for moving a shared queue with the instructions for moving a non-shared queue (described in “Moving a non-shared queue” on page 107) as appropriate. Figure 50 gives a sample job to do this.

```
//UTILITY EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD DSN=thlqua1.SCSQANLE,DISP=SHR
// DD DSN=thlqua1.SCSQAUTH,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
COMMAND DDNAME(MOVEQ)
/*
//MOVEQ DD *
ALTER QL(Queue_To_Move) PUT(DISABLED)
DELETE QL(TEMP_QUEUE) PURGE
DEFINE QL(TEMP_QUEUE) LIKE(Queue_To_Move) PUT(ENABLED) GET(ENABLED) QSGDISP(QMGR)
MOVE QLOCAL(Queue_To_Move) TOQLOCAL(TEMP_QUEUE)
DELETE QL(Queue_To_Move)
DEFINE QL(Queue_To_Move) LIKE(TEMP_QUEUE) STGCLASS(NEW) QSGDISP(QMGR)
MOVE QLOCAL(TEMP_QUEUE) TOQLOCAL(Queue_To_Move)
DELETE QL(TEMP_QUEUE)
/*
```

Figure 50. Sample job for moving a shared queue to a non-shared queue



## Migrating non-shared queues to shared queues

When you migrate non-shared queues to shared queues, perform the whole procedure on Version 5.2 of MQSeries for OS/390. The queue manager you use must be a member of a queue-sharing group.

### The first (or only) queue manager in the queue-sharing group

Figure 49 on page 120 shows an example job for moving a non-shared queue to a shared queue. Do this for each queue that needs migrating.

#### Notes:

1. Messages on shared queues are subject to certain restrictions on the maximum message size, message persistence, and queue index type, so you might not be able to move some non-shared queues to a shared queue.
2. You must use the correct index type for shared queues. If you migrate a transmission queue to be a shared queue, the index type must be MSGID.

If the queue is empty, or you do not need to keep the messages that are on it, migrating the queue is simpler. Figure 51 shows an example job to use in these circumstances.

```
//UTILITY EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD DSN=thlqua1.SCSQANLE,DISP=SHR
// DD DSN=thlqua1.SCSQAUTH,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
COMMAND DDNAME(MOVEQ)
/*
//MOVEQ DD *
DELETE QL(TEMP_QUEUE) PURGE
DEFINE QL(TEMP_QUEUE) LIKE(Queue_TO_MOVE) PUT(ENABLED) GET(ENABLED)
DELETE QL(Queue_TO_MOVE)
DEFINE QL(Queue_TO_MOVE) LIKE(TEMP_QUEUE) CFSTRUCT(NEW) QSGDISP(SHARED)
DELETE QL(TEMP_QUEUE)
/*
```

Figure 51. Sample job for moving a non-shared queue without messages to a shared queue

### Any other queue managers in the queue-sharing group

1. For each queue that does not have the same name as an existing shared queue, move the queue as described in Figure 49 on page 120 or Figure 51.
2. For queues that have the same name as an existing shared queue, move the messages to the shared queue using the commands shown in Figure 52.

```
MOVE QLOCAL(Queue_TO_MOVE) QSGDISP(QMGR) TOQLOCAL(Queue_TO_MOVE)
DELETE QLOCAL(Queue_TO_MOVE) QSGDISP(QMGR)
```

Figure 52. Moving messages from a non-shared queue to an existing shared queue

## Managing queue-sharing groups and shared queues

---

### Managing group objects

MQSeries automatically copies the definition of a group object to page set zero of each queue manager that uses it. You can alter the copy of the definition temporarily if you want, and MQSeries allows you to refresh the page set copies from the repository copy if required. MQSeries always tries to refresh the page set copies from the repository copy on start up (for channel objects, this is done when the channel initiator restarts). This ensures that the page set copies reflect the version on the repository, including any changes that were made when the queue manager was inactive.

There are circumstances under which the refresh is not performed, for example:

- If a copy of the queue is open, a refresh that changes the usage of the queue will fail.
- If a copy of a queue has messages on it, a refresh that deletes that queue will fail.

In these circumstances, the refresh is not performed on that copy, but is performed on the copies on all other queue managers. Because of this, you should check for and correct any problems with copy objects after adding, changing, or deleting a group object, and at queue manager or channel initiator restart.

---

### Managing the Coupling Facility

This section describes the following tasks:

- “Adding a Coupling Facility structure”
- “Removing a Coupling Facility structure”

#### Adding a Coupling Facility structure

There are no MQSeries actions required when you add a Coupling Facility structure. The information about setting up the Coupling Facility in the *MQSeries for OS/390 System Setup Guide* describes the rules for naming Coupling Facility structures, and how to define structures in the CFRM policy data set.

#### Removing a Coupling Facility structure

To remove a Coupling Facility structure, follow this procedure:

- Use the following command to get a list of all the queues using the Coupling Facility structure that you want to delete:  
`DISPLAY QUEUE(*) QSGDISP(SHARED) CFSTRUCT(structure-name)`
- Delete all the queues that use the structure.
- Stop and restart each queue manager in the queue-sharing group in turn to cause MQSeries and DB2 to disconnect from the structure and delete information about it. (You don't need to stop all the queue managers at once; just one at a time.)
- Remove the structure definition from your CFRM policy data set and run the IXCMIAPU utility. (This is the reverse of customization task 10 (set up the coupling facility) described in the *MQSeries for OS/390 System Setup Guide*.)

## Part 5. Recovery and restart

<b>Chapter 12. Restarting MQSeries</b> . . . . .	125
Restarting after a normal shutdown . . . . .	125
Restarting after an abnormal termination . . . . .	125
Restarting if you have lost your page sets . . . . .	125
Restarting if you have lost your log data sets. . . . .	126
Alternative site recovery. . . . .	127
Reinitializing MQSeries . . . . .	130
Reinitializing a queue manager that is not in a queue-sharing group . . . . .	130
Reinitializing queue managers in a   queue-sharing group . . . . .	131
<b>Chapter 13. Using the OS/390 Automatic Restart Manager (ARM)</b> . . . . .	133
What is the ARM?. . . . .	133
ARM couple data sets . . . . .	134
ARM policies . . . . .	134
Defining an ARM policy. . . . .	134
Activating an ARM policy . . . . .	135
Registering with ARM . . . . .	135
Using ARM in an MQSeries network . . . . .	136
Restarting on a different OS/390 image with LU 6.2 . . . . .	136
Restarting on a different OS/390 image with   TCP/IP . . . . .	137
When using clustering . . . . .	137
When connecting to a queue-sharing group	138
<b>Chapter 14. Recovering units of work manually</b>	139
Displaying connections and threads . . . . .	139
Active threads . . . . .	140
In-doubt threads . . . . .	140
Recovering CICS units of recovery manually . . . . .	141
What happens when the CICS adapter restarts	141
How to resolve CICS units of recovery manually	143
Recovering IMS units of recovery manually . . . . .	145
What happens when the IMS adapter restarts	145
How to resolve IMS units of recovery manually	145
Recovery procedure . . . . .	145
Recovering RRS units of recovery manually . . . . .	147
Recovering units of recovery on another queue   manager in the queue-sharing group . . . . .	148
<b>Chapter 15. Example recovery scenarios</b> . . . . .	149
Shared queue problems . . . . .	150
Queue is both private and shared . . . . .	150
Active log problems . . . . .	151
Dual logging is lost . . . . .	151
Active log stopped . . . . .	151
One or both copies of the active log data set are damaged . . . . .	152
Write I/O errors on an active log data set . . . . .	153
I/O errors occur while reading the active log	153
Replacing the data set . . . . .	154
Active log is becoming full or is full. . . . .	155
Archive log problems. . . . .	157
Allocation problems . . . . .	157
Off-load task terminated abnormally . . . . .	157
Insufficient DASD space to complete off-load processing . . . . .	158
Read I/O errors on the archive data set while MQSeries is restarting . . . . .	159
BSDS problems. . . . .	160
Error occurs while opening the BSDS . . . . .	160
Log content does not agree with the BSDS information . . . . .	161
Both copies of the BSDS are damaged . . . . .	161
Unequal time stamps. . . . .	162
Out of synchronization . . . . .	162
I/O error. . . . .	163
Page set problems. . . . .	164
Page set I/O errors . . . . .	164
Page set full. . . . .	165
Coupling Facility and DB2 problems . . . . .	166
Storage medium full . . . . .	166
A DB2 system fails . . . . .	166
A DB2 data-sharing group fails . . . . .	167
DB2 and the Coupling Facility fail . . . . .	168
Problems with long-running units of work . . . . .	169
Old unit of work found during restart . . . . .	169
IMS-related problems. . . . .	170
IMS is unable to connect to MQSeries . . . . .	170
IMS application problem . . . . .	170
IMS is not operational . . . . .	171
Hardware problems . . . . .	172



---

## Chapter 12. Restarting MQSeries

This chapter discusses how to restart your MQSeries subsystem in the following circumstances:

- “Restarting after a normal shutdown”
- “Restarting after an abnormal termination”
- “Restarting if you have lost your page sets”
- “Restarting if you have lost your log data sets” on page 126
- “Alternative site recovery” on page 127
- “Reinitializing MQSeries” on page 130

---

### Restarting after a normal shutdown

If MQSeries was stopped with the +CSQ1 STOP QMGR command, the system finishes its work in an orderly way and takes a termination checkpoint before stopping. When you restart MQSeries, it uses information from the system checkpoint and recovery log to determine the system status at shutdown.

To restart MQSeries, issue the +CSQ1 START QMGR command as described in “Chapter 2. Starting and stopping MQSeries” on page 21.

---

### Restarting after an abnormal termination

MQSeries automatically detects whether restart follows a normal shutdown or an abnormal termination.

Starting MQSeries after it has terminated abnormally is different from starting it after the +CSQ1 STOP QMGR command has been issued. If MQSeries terminates abnormally, it terminates without being able to finish its work or take a termination checkpoint.

To restart MQSeries, issue the +CSQ1 START QMGR command as described in “Chapter 2. Starting and stopping MQSeries” on page 21. When you restart MQSeries after an abnormal termination, it refreshes its knowledge of its status at termination using information in the log, and notifies you of the status of various tasks.

Normally, the restart process resolves all inconsistent states. But, in some cases, you must take specific steps to resolve inconsistencies. This is described in “Chapter 14. Recovering units of work manually” on page 139.

---

### Restarting if you have lost your page sets

If you have lost your page sets, you will need to restore them from your backup copies before you can restart MQSeries. This is described in “How to back up and recover page sets” on page 111.

MQSeries might take a long time to restart under these circumstances because of the length of time needed for media recovery.

### Restarting if you have lost your log data sets

If, after stopping MQSeries (using the STOP QMGR command), both copies of the log are lost or found to be damaged, it is possible to restart MQSeries providing you have a consistent set of page sets (produced using “Method 1: Full backup” on page 111).

Follow this procedure:

1. Define new page sets to correspond to each existing page set in your MQSeries subsystem. See the *MQSeries for OS/390 System Setup Guide* for information about page set definition.  
Ensure that each new page set is larger than the corresponding source page set.
2. Use the FORMAT function of CSQUTIL to format the destination page set. See “Formatting page sets (FORMAT)” on page 183 for more details.
3. Use the RESETPAGE function of CSQUTIL to copy the existing page sets or reset them in place, and reset the log RBA in each page. See “Copying a page set and resetting the log (RESETPAGE)” on page 187 for more information about this function.
4. Redefine your MQSeries log data sets and BSDS using CSQJU003 (see “Chapter 18. The change log inventory utility (CSQJU003)” on page 209).
5. Restart MQSeries, using the new page sets. To do this, you do one of the following:
  - Change the MQSeries startup procedure to reference the new page sets. See the *MQSeries for OS/390 System Setup Guide* for more information.
  - Use Access Method Services to delete the old page sets and then rename the new page sets, giving them the same names as the old page sets.

**Attention:** Before you delete any MQSeries page set, be sure that you have made the required backup copies.

| If the queue manager is a member of a queue-sharing group, GROUP and  
| SHARED object definitions and messages will not be affected by this in general.  
| However, if any shared-queue messages are involved in a unit of work that was  
| covered by the lost or damaged logs, the effect on such uncommitted messages is  
| unpredictable.

## Alternative site recovery

In the case of a total loss of an MQSeries computing center, you can recover on another MQSeries system or queue-sharing group of systems at a recovery site. To be able to do this, you must regularly back up the page sets and the logs. As with all data recovery operations, the objectives of disaster recovery are to lose as little data, workload processing (updates), and time as possible.

At the recovery site:

- The recovery MQSeries queue managers **must** have the same names as the lost queue managers.
- The system parameter module (for example, CSQZPARM) used on each recovery queue manager should contain the same parameters as the corresponding lost queue manager.

If you are using a queue-sharing group, you should first set up the Coupling Facility to match that at the lost computing center, and reestablish your DB2 systems and data-sharing groups with the same names as the lost systems. This is described in “Coupling Facility and DB2 problems” on page 166 and the *DB2 for OS/390 Administration Guide*. There is no recovery of shared messages; they will all be lost unless you have made back-up copies independently using the COPY function of the CSQUTIL utility program. Shared objects will be recovered to the point in time achieved by DB2 recovery (as described in “A DB2 system fails” on page 166).

When you have done this, reestablish all your queue managers as described in the following procedure. This can be used to perform disaster recovery at the recovery site for a single queue manager. It assumes that all that is available are:

- Copies of the archive logs and BSDSs created by normal running at the primary site (the active logs will have been lost along with the queue manager at the primary site).
- Copies of the page sets from the queue manager at the primary site that are the same age or older than the most recent archive log copies available.

If required, dual active and archive logs should be considered, and the BSDS updates applied to both copies:

1. Define new page set data sets and load them with the data in the copies of the page sets from the primary site.
2. Define new active log data sets.
3. Define a new BSDS data set and use Access Method Services REPRO to copy the **most recent** archived BSDS into it.
4. Use the print log map utility CSQJU004 to print information from this most recent BSDS. At the time this BSDS was archived, the most recent archived log you have would have just been truncated as an active log, and will not appear as an archived log. Record the STARTRBA and ENDRBA of this log.
5. Use Access Method Services REPRO to copy the most recent archived log into one of the active logs.
6. Use the change log inventory utility CSQJU003 to remove all active log information from the BSDS.
7. Use CSQJU003 to add active logs to the BSDS, including the RBA range of the logs used in Step 5 as found in Step 4.

## Alternative site recovery

8. Use CSQJU003 to add a restart control record to the BSDS. Specify CRESTART CREATE, ENDRBA=highrba, where highrba is the high RBA of the most recent archive log available (found in Step 4 on page 127), plus 1.

The BSDS now describes one active log with an RBA range, all other active logs as being empty, all the archived logs you have available, and no checkpoints beyond the end of your logs.

9. Restart MQSeries with the usual START QMGR command. During initialization, an operator reply message such as the following will be issued:

```
CSQJ245D +CSQ1 RESTART CONTROL INDICATES TRUNCATION AT RBA highrba.  
REPLY Y TO CONTINUE, N TO CANCEL
```

Reply Y to start MQSeries. MQSeries will start, and will recover data up to ENDRBA specified in the CRESTART statement.

See “Part 6. Using the MQSeries Utilities” on page 173 for information about using CSQJU003 and CSQJU004.

Figure 53 shows sample input statements for CSQJU003 for steps 6, 7, and 8:

```
* Step 6  
DELETE DSNAME=MQM2.LOGCOPY1.DS01  
DELETE DSNAME=MQM2.LOGCOPY1.DS02  
DELETE DSNAME=MQM2.LOGCOPY1.DS03  
DELETE DSNAME=MQM2.LOGCOPY1.DS04  
DELETE DSNAME=MQM2.LOGCOPY2.DS01  
DELETE DSNAME=MQM2.LOGCOPY2.DS02  
DELETE DSNAME=MQM2.LOGCOPY2.DS03  
DELETE DSNAME=MQM2.LOGCOPY2.DS04  
  
* Step 7  
NEWLOG DSNAME=MQM2.LOGCOPY1.DS01,COPY1  
STARTRBA=05C000, ENDRBA=000000062FFF  
NEWLOG DSNAME=MQM2.LOGCOPY1.DS02,COPY1  
NEWLOG DSNAME=MQM2.LOGCOPY1.DS03,COPY1  
NEWLOG DSNAME=MQM2.LOGCOPY1.DS04,COPY1  
NEWLOG DSNAME=MQM2.LOGCOPY2.DS01,COPY2  
STARTRBA=05C000, ENDRBA=000000062FFF  
NEWLOG DSNAME=MQM2.LOGCOPY2.DS02,COPY2  
NEWLOG DSNAME=MQM2.LOGCOPY2.DS03,COPY2  
NEWLOG DSNAME=MQM2.LOGCOPY2.DS04,COPY2  
  
* Step 8  
CRESTART CREATE, ENDRBA=063000
```

Figure 53. Sample input statements for CSQJU003

The things you need to consider for restarting the channel initiator at the recovery site are similar to those faced when using ARM to restart the channel initiator on a different OS/390 image. See “Using ARM in an MQSeries network” on page 136 for more information. Your recovery strategy should also cover recovery of the MQSeries product libraries and the application programming environments that use MQSeries (CICS, for example).

Other functions of the change log inventory utility (CSQJU003) can also be used in disaster recovery scenarios. The HIGHRBA function allows the update of the highest RBA written and highest RBA off-loaded values within the bootstrap data set. The CHECKPT function allows the addition of new checkpoint queue records



## **Alternative site recovery**

or the deletion of existing checkpoint queue records in the BSDS. These functions might affect the integrity of the MQSeries system and should only be used in disaster recovery scenarios under the guidance of IBM service personnel.

### Reinitializing MQSeries

If MQSeries has terminated abnormally you might not be able to restart it. This could be because your page sets or logs have been lost, truncated, or corrupted. If this has happened, you might have to reinitialize MQSeries (perform a cold start). Before you attempt this, check that none of the other restart scenarios described in this chapter will work for you.

#### Attention

**You should only perform a cold start if you are unable to restart MQSeries any other way.** Performing a cold start will enable you to recover your MQSeries system and your object definitions; you will **not** be able to recover your message data. Check that none of the other restart scenarios described in this chapter will work for you before you do this.

When you have restarted, all your MQSeries objects will be defined and available for use, but there will be no message data.

### Reinitializing a queue manager that is not in a queue-sharing group

To cold start MQSeries, follow this procedure:

1. Prepare the object definition statements that will be used when you restart MQSeries. To do this, either:
  - If page set zero is available, use the CSQUTIL SDEFS function (see “Producing a list of MQSeries define commands (SDEFS)” on page 195). You must get definitions for all object types (channels, namelists, processes, queues, and storage classes).
  - If page set zero is not available, use the definitions from the last time you backed up your object definitions.
2. Redefine your queue manager data sets (do not do this until you have completed step 1). See the *MQSeries for OS/390 System Setup Guide* for information about redefining your log data sets, BSDS, and page sets.
3. Restart MQSeries using the newly defined and initialized log data sets, BSDS, and page sets. Use the object definition input statements that you created in step 1 as input in the CSQINP2 initialization input data set.

You do not need to include objects with object dispositions of GROUP or SHARED because their definitions are stored in the DB2 shared repository. However, it does not matter if you do.

## Reinitializing queue managers in a queue-sharing group

In a queue-sharing group, the situation is more complex. It might be necessary to reinitialize one or more queue managers because of page set or log problems, but there might also be problems with DB2 or the Coupling Facility to deal with. Because of this, there are a number of alternatives:

### Cold start

Reinitializing the entire queue-sharing group involves forcing all the Coupling Facilities structures, clearing all object definitions for the queue-sharing group from DB2, deleting or redefining the logs and BSDS, and formatting page sets for all the queue managers in the queue-sharing group.

### Shared definitions retained

Delete or redefine the logs and BSDS, format page sets for all queue managers in the queue-sharing group, and force all the Coupling Facilities structures. On restart, all messages will have been deleted. The queue managers will recreate COPY objects that correspond to GROUP objects that still exist in the DB2 database. Any shared queues will exist (although the messages on them will not) and can be used.

### Single queue manager reinitialized

Delete or redefine the logs and BSDS, and format page sets for the single queue manager (this deletes all its private objects and messages). On restart, the queue manager will recreate COPY objects that correspond to GROUP objects that still exist in the DB2 database. Any shared queues will exist as will the messages on them and can be used.

### Point in time recovery of a queue-sharing group

This is the alternative site disaster recovery scenario. There is no recovery of shared messages; they will all be lost unless you have made back-up copies independently using the COPY function of the CSQUTIL utility program. Shared objects will be recovered to the point in time achieved by DB2 recovery (described in "A DB2 system fails" on page 166). Each queue manager can be recovered to a point in time achievable from the back-up copies available at the alternative site.

It is not necessary to try to restore each queue manager to the same point in time because there are no interdependencies between the local objects on different queue managers (which are what is actually being recovered), and the queue manager resynchronization with DB2 on restart will create or delete COPY objects as necessary on a queue manager by queue manager basis.

## Reinitializing MQSeries

---

## Chapter 13. Using the OS/390 Automatic Restart Manager (ARM)

This chapter discusses the following topics:

- “What is the ARM?”
- “ARM couple data sets” on page 134
- “ARM policies” on page 134
- “Using ARM in an MQSeries network” on page 136

---

### What is the ARM?

The OS/390 Automatic Restart Manager (ARM) is an OS/390 recovery function that can improve the availability of your MQSeries subsystems. When a job or task fails, or the system on which it is running fails, ARM can restart the job or task without operator intervention.

If a queue manager or a channel initiator has failed, ARM restarts it on the same OS/390 image. If OS/390, and hence a whole group of related subsystems and applications have failed, ARM can restart all the failed systems automatically, in a predefined order, on another OS/390 image within the sysplex. This is called a cross-system restart.

The channel initiator should be restarted by ARM only in exceptional circumstances. If the queue manager is restarted by ARM, the channel initiator should be restarted from the CSQINP2 initialization data set (see “Using ARM in an MQSeries network” on page 136).

You can use ARM to restart an MQSeries subsystem that uses LU 6.2 communication protocols on a different OS/390 image within the sysplex in the event of OS/390 failure. (You cannot do this if you use TCP/IP communication protocols.) The network implications of MQSeries ARM restart on a different OS/390 image are discussed in “Using ARM in an MQSeries network” on page 136.

To enable automatic restart:

- You must set up an ARM couple data set.
- You must define the automatic restart actions that you want OS/390 to perform in an *ARM policy*.
- You must start the ARM policy.

Also, MQSeries must register with ARM at startup (this happens automatically).

**Note:** If you want to restart queue managers in different OS/390 images automatically, every queue manager must be defined as a subsystem in each OS/390 image on which that queue manager might be restarted, with a sysplex wide unique 4-character subsystem name.

---

### ARM couple data sets

You must ensure that you define the couple data sets required for ARM, and that they are online and active before you start any MQSeries subsystem for which you want ARM support. MQSeries automatic ARM registration fails if the couple data sets are not available at MQSeries startup. In this situation, MQSeries assumes that the absence of the couple data set means that you do not want ARM support, and initialization continues.

See the *OS/390 MVS Setting up a Sysplex* manual for information about ARM couple data sets. x

---

### ARM policies

ARM functions are controlled by a user-defined *ARM policy*. Each OS/390 image running a queue manager instance that is to be restarted by ARM must be connected to an ARM couple data set with an active ARM policy.

IBM® provides a default ARM policy. You can define new policies, or override the policy defaults by using the *administrative data utility* (IXCMIAPU) provided with OS/390. The *OS/390 MVS Setting up a Sysplex* manual describes this utility, and includes full details of how to define an ARM policy.

Figure 54 shows an example of an ARM policy. This sample policy will restart any MQSeries queue manager within a sysplex, in the event that either the queue manager failed, or a whole system failed.

```
//IXCMIAPU EXEC PGM=IXCMIAPU,REGION=2M
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  DATA TYPE(ARM)
DEFINE POLICY NAME(ARMPOL1) REPLACE(YES)
  RESTART_GROUP(DEFAULT)
  ELEMENT(*)
  RESTART_ATTEMPTS(0) /* Jobs not to be restarted by ARM */
  RESTART_GROUP(GROUP1)
  ELEMENT(SYSMQMGRMQ*) /* These jobs to be restarted by ARM */
/*
```

Figure 54. Sample ARM policy

### Defining an ARM policy

We recommend that you set up your ARM policy as follows:

- Define RESTART\_GROUPS for each queue manager instance which also contain any CICS or IMS subsystems that connect to that queue manager instance. If you use a subsystem naming convention, you might be able to use the '?' and '\*' wild-card characters in your element names to achieve the above with minimum definition effort.
- Specify TERMTYPE(ELEMTerm) for your channel initiators to indicate that they will be restarted only if the channel initiator has failed and the OS/390 image has not failed.
- Specify TERMTYPE(ALLTERM) for your queue managers to indicate that they will be restarted if either the queue manager has failed or the OS/390 image has failed.
- Specify RESTART\_METHOD(BOTH, PERSIST) for both queue managers and channel initiators. This tells ARM to restart using the JCL it saved (after

resolution of system symbols) during the last startup. It tells ARM to do this irrespective of whether the individual element failed, or the OS/390 image failed.

- Accept the default values for all the other ARM policy options.

## Activating an ARM policy

To start your automatic restart management policy, issue the following OS/390 command:

```
SETXCF START,POLICY,TYPE=ARM,POLNAME=mypol
```

When the policy is started, all systems connected to the ARM couple data set use the same active policy.

Use the SETXCF STOP command to disable automatic restarts.

### Registering with ARM

MQSeries registers automatically as an *ARM element* during its startup phase (subject to ARM availability). It deregisters during its shutdown phase, unless requested not to.

At startup, the queue manager determines whether ARM is available. If it is, MQSeries registers using the name SYSMQMGR*ssid*, where *ssid* is the 4-character queue-manager name, and SYSMQMGR is the element type.

The STOP QMGR MODE(QUIESCE) and STOP QMGR MODE(FORCE) MQSeries commands deregister MQSeries from ARM (if it was registered with ARM at startup). This prevents ARM restarting this queue manager. The STOP QMGR MODE(RESTART) command does not deregister the queue manager from ARM. It is thus eligible for immediate automatic restart.

Each MQSeries channel initiator address space determines whether ARM is available, and if so will register with the element name SYSMQCH*ssid*, where *ssid* is the queue manager name, and SYSMQCH is the element type.

The channel initiator is always deregistered from ARM when it stops normally, and remains registered only if it ends abnormally. The channel initiator is always deregistered if the queue manager fails.

### Using ARM in an MQSeries network

You should set up your MQSeries system so that the channel initiators and associated listeners are started automatically when MQSeries is restarted. To ensure fully automatic MQSeries restart on the same OS/390 image for both LU 6.2 and TCP/IP communication protocols:

- Start your channel initiator automatically by adding the appropriate START CHINIT command to the CSQINP2 data set.
- Start your listeners automatically by adding the appropriate START LISTENER command to the CSQINPX data set.

See the *MQSeries for OS/390 System Setup Guide* for information about the CSQINP2 and CSQINPX data sets.

### Restarting on a different OS/390 image with LU 6.2

If you use only LU 6.2 communication protocols you should also do the following to enable network reconnect after automatic restart of MQSeries on a different OS/390 image within the sysplex:

- Define each MQSeries queue manager within the sysplex with a unique subsystem name.
- Define each channel initiator within the sysplex with a unique LUNAME. This is specified in both the channel initiator parameter module and in the START LISTENER command.

**Note:** The LUNAME names an entry in the APPC side table, which in turn maps this to the actual LUNAME.

- Set up a shared APPC side table, which is referenced by each OS/390 image within the sysplex. This should contain an entry for each channel initiator's LUNAME. See the *MVS Planning: APPC/MVS Management* manual for information about this.
- Set up an APPCPMxx member of SYS1.PARMLIB for each channel initiator within the sysplex to contain an LUADD to activate the APPC side table entry for that channel initiator. These members should be shared by each OS/390 image. The appropriate SYS1.PARMLIB member is activated by a SET APPC=xx OS/390 command which is issued automatically during ARM restart of MQSeries (and its channel initiator) on a different OS/390 image, as described below.
- Use the LU62ARM keyword of the CSQ6CHIP macro to specify the xx suffix of this SYS1.PARMLIB member for each channel initiator in the channel initiator parameter module. This will cause the channel initiator to issue the required SET APPC=xx OS/390 command to activate its LUNAME.

You should define your ARM policy to restart the channel initiator only if it fails while its OS/390 image stays up. You should not restart the channel initiator automatically if its OS/390 image also fails, but use the CSQINP2 and CSQINPX data sets to start the channel initiator and listeners.



## Restarting on a different OS/390 image with TCP/IP

If you are using TCP/IP as your communication protocol, you can reallocate a TCP/IP address after moving MQSeries to a different OS/390 image only if you are using clusters or if you are connecting to a queue-sharing group using a WLM dynamic Domain Name System (DNS) logical group name. If you are using virtual IP addresses, it is possible to configure these to recover on other OS/390 images, allowing channels connecting to that queue manager to reconnect without any changes.

### When using clustering

OS/390 ARM responds to a system failure by restarting MQSeries on a different OS/390 image in the same sysplex; this system has a different TCP/IP address to the original OS/390 image. The following explains how you can use MQSeries clusters to re-assign a queue manager's TCP/IP address after it has been moved by ARM restart to a different OS/390 image.

When a client queue manager detects the MQSeries for OS/390 failure (as a channel failure), it responds to this by reallocating suitable messages on its cluster transmission queue to a different server queue manager that hosts a different instance of the target cluster queue. However, it cannot reallocate messages that are bound to the original server by affinity constraints, or messages that are in doubt because the server queue manager failed during end-of-batch processing. In order to process these messages, you need to do the following:

1. Allocate a different cluster-receiver channel name and a different TCP/IP port to each OS/390 queue manager. Each queue manager needs a different port so that two systems can share a single TCP/IP stack on an OS/390 image. One of these is the queue manager originally running on that OS/390 image, and the other is the queue manager that ARM will restart on that OS/390 image following a system failure. You should configure each port on each OS/390 image, so that ARM can restart any queue manager on any OS/390 image.
2. Create a different channel initiator command input file (CSQINPX) for each queue manager and OS/390 image combination, to be referenced during channel initiator startup.

Each CSQINPX file must include a START LISTENER PORT(port) command specific to that queue manager, and an ALTER CHANNEL command for a cluster-receiver channel specific to that queue manager and OS/390 image combination. The ALTER CHANNEL command needs to set the connection name to the TCP/IP name of the OS/390 image on which it is restarted. It must include the port number specific to the restarted queue manager as part of the connection name.

The start-up JCL of each queue manager can have a fixed data set name for this CSQINPX file, and each OS/390 image must have a different version of each of these CSQINPX files on a non-shared DASD volume.

In the event of ARM restart, MQSeries for OS/390 advertises the changed channel definition to the cluster repository, which in turn publishes it to all the client queue managers that have expressed an interest in the server queue manager.

The client queue manager sees the server queue manager failure as a channel failure, and tries to restart the failed channel. When the client queue manager learns the new server connection-name, the channel restart reconnects the client queue manager to the restarted server queue manager. The client queue manager can then resynchronize its messages, resolve any in-doubt messages on the client queue manager's transmission queue, and normal processing can continue.

## Using ARM in an MQSeries network

### When connecting to a queue-sharing group

When connecting to a queue-sharing group through a TCP/IP dynamic Domain Name System (DNS) logical group name, the connection name in your channel definition specifies the logical group name of your queue-sharing group, not the hostname or IP address of a physical machine. When this channel starts, it connects to the dynamic DNS and is then connected to one of the queue managers in the queue-sharing group. This process is explained in the *MQSeries Intercommunication* manual.

In the unlikely event of an image failure, one of the following will occur:

- The queue managers on the failing image will de-register from the dynamic DNS running on your sysplex. The channel responds to the connection failure by entering RETRYING state and then connects to the dynamic DNS running on the sysplex. The dynamic DNS allocates the inbound request to one of the remaining members of the queue-sharing group that is still running on the remaining images.
- If no other queue manager in the queue-sharing group is active and ARM restarts the queue manager and channel initiator on a different image, the group listener registers with dynamic DNS from this new image. This means that the logical group name (from the connection name field of the channel) connects to the dynamic DNS and is then connected to the same queue manager, now running on a different image. No change was required to the channel definition.

For this type of transparent recovery to occur, the following points must be noted:

- On OS/390, the dynamic DNS runs on one of the OS/390 images in the sysplex. If this image were to fail, the dynamic DNS needs to be configured so that there is a secondary name server active in the sysplex, acting as an alternative to the primary name server. Information about primary and secondary dynamic DNS servers can be found in the *OS/390 SecureWay CS IP Configuration* manual.
- The TCP/IP group listener might have been started on a particular IP address that might not be available on this OS/390 image. If this is the case, the listener might need to be started on a different IP address on the new image. If you are using virtual IP addresses, you can configure these to recover on other OS/390 images so that no change to the START LISTENER command is required.

---

## Chapter 14. Recovering units of work manually

This chapter discusses the following topics:

- “Displaying connections and threads”
- “Recovering CICS units of recovery manually” on page 141
- “Recovering IMS units of recovery manually” on page 145
- “Recovering RRS units of recovery manually” on page 147
- “Recovering units of recovery on another queue manager in the queue-sharing group” on page 148

---

### Displaying connections and threads

You can use the DISPLAY THREAD command (described in the *MQSeries MQSC Command Reference* manual) to get information about connections to MQSeries and their associated threads. You can display active threads to see what is currently happening, or to see what needs to be terminated in order to allow MQSeries to shut down and you can display in-doubt threads to help with recovery.

The following information is returned for active threads:

- The connection name
- The connection status
- The number of MQSeries requests
- The thread cross-reference identifier
- The user ID of the connection
- The address space ID
- The unit of recovery ID

You can find more information about these fields in the description of message CSQV402I in the *MQSeries for OS/390 Messages and Codes* manual.

The following information is returned for in-doubt threads:

- The connection name
- The thread cross-reference identifier
- The recovery network ID
- The unit of recovery ID

You can find more information about these fields in the description of message CSQV406I in the *MQSeries for OS/390 Messages and Codes* manual.

To reduce the amount of information produced, you can choose to display a summary of active threads for each active connection (this does not include threads used internally by MQSeries). The following information is produced for each job:

- The connection name
- The connection type
- The user ID
- The address space ID
- The number of threads associated with the connection

## Recovering units of work manually

You can find more information about these fields in the description of message CSQV432I in the *MQSeries for OS/390 Messages and Codes* manual.

### Active threads

Each current connection to MQSeries is represented by one active thread, but certain connections (such as those by the CICS adapter or the mover) might have additional threads associated with them. Note that the CTHREAD system parameter (described in the *MQSeries for OS/390 System Setup Guide*) controls the number of *connections*, not the number of threads.

In addition to the connection name, the display includes the associated user ID (if known), the number of MQSeries requests made by a thread, and the thread cross-reference identifier. The number of MQSeries requests is generally 0 for associated threads. The thread cross-reference identifier is shown in character form if possible, but otherwise in hexadecimal; its format depends on the type of connection:

**CICS** Contains the CICS thread number, transaction name, and task number.

**IMS** Contains the IMS PST region identifier and PSB name.

#### **Batch, TSO, and RRS**

Contains nulls or blanks.

**Mover** Blank for connections. Associated threads contain 'T' (X'E3') or 'XX\*\*' (X'E7E75C5C') at character position 5.

### In-doubt threads

An in-doubt thread is one that is in the second pass of the two-phase commit operation. Resources are held in MQSeries on its behalf. External intervention is needed to resolve the status of in-doubt threads. This might only involve starting the recovery coordinator (CICS, IMS, or RRS) or might involve more, as described in the following sections. They might have been in doubt at the last restart, or they might have become in doubt since the last restart.

The display includes the thread cross-reference identifier, which might be needed if manual recovery is necessary.

## Recovering CICS units of recovery manually

This section describes what happens when the CICS adapter restarts, and then explains how to deal with any unresolved units of recovery that arise.

### What happens when the CICS adapter restarts

For background information, see the *MQSeries for OS/390 Concepts and Planning Guide*.

Whenever a connection is broken, the adapter has to go through a *restart phase* during the *reconnect process*. The restart phase resynchronizes resources. Resynchronization between CICS and MQSeries enables *in-doubt units of work* to be identified and resolved.

Resynchronization can be caused by:

- An explicit request from the distributed queuing component
- An implicit request when a connection is made to MQSeries

If the resynchronization is caused by connecting to MQSeries, the sequence of events is:

1. The connection process gets a list of unit of work (UOW) IDs that MQSeries thinks are in doubt.
2. The UOW IDs are displayed on the console in CSQC313I messages.
3. The UOW IDs are passed to CICS.
4. CICS initiates a resynchronization task (CRSY) for each in-doubt UOW ID.
5. The result of the task for each in-doubt UOW is displayed on the console.

You need to check the messages that are displayed during the connect process:

#### CSQC313I

Shows that a UOW is in doubt.

#### CSQC400I

Identifies the UOW and is followed by one of these messages:

- CSQC402I and CSQC403I show that the UOW was resolved successfully (committed or backed out).
- CSQC404E, CSQC405E, CSQC406E, and CSQC407E show that the UOW was not resolved.

#### CSQC409I

Shows that all UOWs were resolved successfully.

#### CSQC408I

Shows that not all UOWs were resolved successfully.

#### CSQC314I

Warns that UOW IDs highlighted with a \* will not be resolved automatically. These UOWs must be resolved explicitly by the distributed queuing component when it is restarted.

Figure 55 on page 142 shows an example set of restart messages displayed on the OS/390 console.

## Recovering units of work manually

```
CSQ9022I +CSQ1 CSQYASCP ' START QMGR' NORMAL COMPLETION
+CSQC323I VICIC1 CSQCQCON CONNECT received from TERMID=PB62 TRANID=CKCN
+CSQC303I VICIC1 CSQCQCON CSQCSERV loaded. Entry point is 850E8918.
+CSQC313I VICIC1 CSQCQCON UOWID=VICIC1.A6E5A6F0E2178D25 is in doubt
+CSQC313I VICIC1 CSQCQCON UOWID=VICIC1.A6E5A6F055B2AC25 is in doubt
+CSQC313I VICIC1 CSQCQCON UOWID=VICIC1.A6E5A6EFFF60D425 is in doubt
+CSQC313I VICIC1 CSQCQCON UOWID=VICIC1.A6E5A6F07AB56D22 is in doubt
+CSQC307I VICIC1 CSQCQCON Successful connection to subsystem VC2
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008BAD18) connect
successful.
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008BAA10) connect
successful.
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008BA708) connect
successful.
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008CAE88) connect
successful.
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008CAB80) connect
successful.
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008CA878) connect
successful.
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008CA570) connect
successful.
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008CA268) connect
successful.
+CSQC403I VICIC1 CSQCTRUE Resolved BACKOUT for
+CSQC400I VICIC1 CSQCTRUE UOWID=VICIC1.A6E5A6F0E2178D25
+CSQC403I VICIC1 CSQCTRUE Resolved BACKOUT for
+CSQC400I VICIC1 CSQCTRUE UOWID=VICIC1.A6E5A6F055B2AC25
+CSQC403I VICIC1 CSQCTRUE Resolved BACKOUT for
+CSQC400I VICIC1 CSQCTRUE UOWID=VICIC1.A6E5A6F07AB56D22
+CSQC403I VICIC1 CSQCTRUE Resolved BACKOUT for
+CSQC400I VICIC1 CSQCTRUE UOWID=VICIC1.A6E5A6EFFF60D425
+CSQC409I VICIC1 CSQCTRUE Resynchronization completed successfully
```

Figure 55. Example restart messages

The total number of CSQC313I messages should equal the total number of CSQC402I plus CSQC403I messages. If the totals are not equal, there are UOWs that the connection process cannot resolve. Those UOWs that cannot be resolved are caused by problems with CICS (for example, a cold start) or with MQSeries, or by distributing queuing. When these problems have been fixed, you can initiate another resynchronization by disconnecting and then reconnecting.

Alternatively, you can resolve each outstanding UOW yourself using the MQSeries RESOLVE INDOUBT command and the UOW ID shown in message CSQC400I. You must then initiate a disconnect and a connect to clean up the *unit of recovery descriptors* in CICS. You need to know the correct outcome of the UOW to resolve UOWs manually.

All messages that are associated with unresolved UOWs are locked by MQSeries and no Batch, TSO, or CICS task can access them.

If CICS fails and an emergency restart is necessary, *do not* vary the GENERIC APPLID of the CICS system. If you do and then reconnect to MQSeries, data integrity with MQSeries cannot be guaranteed. This is because MQSeries treats the new instance of CICS as a different CICS (because the APPLID is different). In-doubt resolution is then based on the wrong CICS log. Similarly, if MQSeries fails, do not change the subsystem ID of the MQSeries system.

## How to resolve CICS units of recovery manually

If the adapter abends, CICS and MQSeries build in-doubt lists either dynamically or during restart, depending on which subsystem caused the abend.

**Note:** If you use the DFH£INDB sample program to show units of work, you might find that it does not always show MQSeries ones correctly.

When CICS connects to MQSeries, there might be one or more units of recovery, that have not been resolved.

One of the following messages is sent to the console:

- CSQC404E
- CSQC405E
- CSQC406E
- CSQC407E
- CSQC408I

For details of what these messages mean, see the *MQSeries for OS/390 Messages and Codes* manual.

CICS retains details of units of recovery that were not resolved during connection startup. An entry is purged when it no longer appears on the list presented by MQSeries.

Any units of recovery that CICS cannot resolve must be resolved manually using MQSeries commands. This manual procedure is rarely used within an installation, because it is required only where operational errors or software problems have prevented automatic resolution. *Any inconsistencies found during in-doubt resolution must be investigated.*

To recover the units of recovery:

1. Obtain a list of the units of recovery from MQSeries by issuing the following command:

```
+CSQ1 DISPLAY THREAD(*) TYPE(INDOUBT)
```

You receive the following messages:

```
CSQV401I +CSQ1 DISPLAY THREAD REPORT FOLLOWS -
CSQV406I +CSQ1 INDOUBT THREADS
NAME      THREAD-XREF  URID  NID
VICIC3    xref              VICIC3.A75E483235A90900
DISPLAY THREAD REPORT COMPLETE
CSQ9022I +CSQ1 CSQVDT ' DISPLAY THREAD' NORMAL COMPLETION
```

For CICS connections, the NID consists of the CICS applid and a unique number provided by CICS at the time the syncpoint log entries are written. This unique number is stored in records written to both the CICS system log and the MQSeries log at syncpoint processing time. This value is referred to in CICS as the *recovery token*.

## Recovering units of work manually

2. Scan the CICS log for entries related to a particular unit of recovery.  
Look for a PREPARE record, for the task-related installation where the recovery token field (JCSRMTKN) equals the value obtained from the network ID. The network ID is supplied by MQSeries in the DISPLAY THREAD command output.  
The PREPARE record in the CICS log for the units of recovery provides the CICS task number. All other entries on the log for this CICS task can be located using this number.  
You can use the CICS journal print utility DFHJUP when scanning the log. For details of using this program, see the *CICS Operations and Utilities Guide*.
3. Scan the MQSeries log for entries related to a particular unit of recovery.  
To do this, scan the MQSeries log to locate the record with the NID required. Then use the URID from this record to obtain the rest of the log records for this unit of recovery.  
When scanning the MQSeries log, note that the MQSeries startup message CSQJ001I provides the start RBA for this session.  
The print log records program (CSQ1LOGP) can be used for that purpose.
4. If you need to, do in-doubt resolution in MQSeries.  
MQSeries can be directed to take the recovery action for a unit of recovery using an MQSeries RESOLVE INDOUBT command.  
For information about RESOLVE INDOUBT, see the *MQSeries MQSC Command Reference* manual.  
To recover all threads associated with *connection-name*, use the NID(\*) option.  
The command produces one of the following messages showing whether the thread is committed or backed out:

```
CSQV414I +CSQ1 THREAD network-id COMMIT SCHEDULED  
CSQV415I +CSQ1 THREAD network-id ABORT SCHEDULED
```

When performing in-doubt resolution, CICS and the adapter are not aware of the commands to MQSeries to commit or back out units of recovery, because only MQSeries resources are affected. However, CICS keeps details about the in-doubt threads that could not be resolved by MQSeries. This information is purged either when the list presented is empty, or when the list does not include a unit of recovery of which CICS has details.



## Recovering IMS units of recovery manually

This section describes what happens when the IMS adapter restarts, and then explains how to deal with any unresolved units of recovery that arise.

### What happens when the IMS adapter restarts

For background information, see the *MQSeries for OS/390 Concepts and Planning Guide*.

Whenever the connection to MQSeries is restarted, either following an MQSeries restart, or an IMS /START SUBSYS command, IMS initiates the following resynchronization process:

1. IMS presents the list of unit of work (UOW) IDs that it believes are in doubt to the MQSeries IMS adapter one at a time with a resolution parameter of Commit or Backout.
2. The IMS adapter passes the resolution request to MQSeries and reports the result back to IMS.
3. Having processed all the IMS resolution requests, the IMS adapter gets from MQSeries a list of all UOWs that MQSeries still holds in doubt that were initiated by the IMS system. These are reported to the IMS master terminal in message CSQQ008I.

**Note:** While a UOW is in doubt, any associated MQSeries message is locked by MQSeries and is not available to any application.

### How to resolve IMS units of recovery manually

When IMS connects to MQSeries, MQSeries might have one or more in-doubt units of recovery that have not been resolved.

If MQSeries has in-doubt units of recovery that IMS did not resolve, the following message is issued at the IMS master terminal:

```
CSQQ008I nn units of recovery are still in doubt in queue manager qmgr-name
```

When this message is issued, IMS was either cold-started or it was started with an incomplete log tape. This message can also be issued if MQSeries or IMS terminates abnormally because of a software error or other subsystem failure.

After receiving the CSQQ008I message:

- The connection remains active.
- IMS applications can still access MQSeries resources.
- Some MQSeries resources remain locked out.

If the in-doubt thread is not resolved, IMS message queues can start to build up. If the IMS queues fill to capacity, IMS terminates. Therefore, users must be aware of this potential difficulty and must monitor IMS until the in-doubt units of recovery are fully resolved.

#### Recovery procedure

Use the following procedure to recover the IMS units of work:

1. Force the IMS log closed, using /SWI OLDS, and then archive the IMS log. Use the utility, DFSERA10, to print the records from the previous IMS log tape.

## Recovering units of work manually

Type X'3730' log records indicate a phase-2 commit request and type X'38' log records indicate an abort request. Record the requested action for the last transaction in each dependent region.

2. Run the DL/I batch job to back out each PSB involved that has not reached a commit point. The process might take some time because transactions are still being processed. It might also lock up a number of records, which could impact the rest of the processing and the rest of the message queues.
3. Produce a list of the in-doubt units of recovery from MQSeries by issuing the following command:

```
+CSQ1 DISPLAY THREAD(*) TYPE(INDOUBT)
```

You receive the following messages:

```
CSQV401I +CSQ1 DISPLAY THREAD REPORT FOLLOWS -
CSQV406I +CSQ1 INDOUBT THREADS -
NAME      THREAD-XREF    URID  NID
name      xref              network-id
name      xref              network-id
DISPLAY THREAD REPORT COMPLETE
CSQ9022I +CSQ1 CSQVDT ' DISPLAY THREAD' NORMAL COMPLETION
```

For IMS, the NID consists of the IMS connection name and a unique number provided by IMS. The value is referred to in IMS as the *recovery token*. For more information, see the *IMS Customization Guide*.

4. Compare the NIDs (IMSID plus OASN in hexadecimal) displayed in the DISPLAY THREAD messages with the OASNs (4 bytes decimal) shown in the DFSERA10 output. Decide whether to commit or back out.
5. Perform in-doubt resolution in MQSeries with the RESOLVE INDOUBT command, as follows:

```
RESOLVE INDOUBT(connection-name)
          ACTION(COMMIT|BACKOUT)
          NID(network-id)
```

For information about RESOLVE INDOUBT, see the *MQSeries MQSC Command Reference* manual.

To recover all threads associated with *connection-name*, use the NID(\*) option. The command results in one of the following messages to indicate whether the thread is committed or backed out:

```
CSQV414I  THREAD network-id COMMIT SCHEDULED
CSQV415I  THREAD network-id BACKOUT SCHEDULED
```

When performing in-doubt resolution, IMS and the adapter are not aware of the commands to MQSeries to commit or back out in-doubt units of recovery because only MQSeries resources are affected.

---

## Recovering RRS units of recovery manually

When RRS connects to MQSeries, MQSeries may have one or more in-doubt units of recovery that have not been resolved. If MQSeries has in-doubt units of recovery that RRS did not resolve, one of the following messages is issued at the OS/390 console:

- CSQ3011I
- CSQ3013I
- CSQ3014I
- CSQ3016I

Both MQSeries and RRS provide tools to display information about in-doubt units of recovery, and techniques for manually resolving them.

In MQSeries, use the `DISPLAY THREAD` command to display information about in-doubt MQSeries threads. The output from the command includes RRS unit of recovery IDs for those MQSeries threads that have RRS as a coordinator. This can be used to determine the outcome of the unit of recovery.

Use the MQSeries `RESOLVE INDOUBT` command to resolve the MQSeries in-doubt thread manually. This command can be used to either commit or back out the unit of recovery after you have determined what the correct decision is.

### Recovering units of recovery on another queue manager in the queue-sharing group

If a queue manager that is a member of a queue-sharing group fails and cannot be restarted, other queue managers in the group can perform peer recovery, and take over from it. However, the queue manager might have in-doubt units of recovery that cannot be resolved by peer recovery because the final disposition of that unit of recovery is known only to the failed queue manager. These units of recovery will be resolved when the queue manager is eventually restarted, but until then, they remain in doubt.

This means that certain resources (for example, messages) might be locked, making them unavailable to other queue managers in the group. In this situation, you can use the DISPLAY THREAD command to display these units of work on the inactive queue manager. If you want to resolve these units of recovery manually to make the messages available to other queue managers in the group, you can use the RESOLVE INDOUBT command.

When you issue the DISPLAY THREAD command to display units of recovery that are in doubt, you can use the QMNAME keyword to specify the name of the inactive queue manager. For example, if you issue the following command:

```
+CSQ1 DISPLAY THREAD(*) TYPE(INDOUBT) QMNAME(QM01)
```

You receive the following messages:

```
CSQV436I +CSQ1 INDOUBT THREADS FOR QM01 -
NAME      THREAD-XREF      URID  NID
USER1     00000000000000000000 CSQ:0001.0
USER2     00000000000000000000 CSQ:0002.0
DISPLAY  THREAD REPORT COMPLETE
```

If the queue manager specified is not inactive, MQSeries does not return information about in-doubt threads, but issues the following message:

```
CSQV435I CANNOT USE QMNAME KEYWORD, QM01 IS ACTIVE
```

Use the MQSeries RESOLVE INDOUBT command to resolve the in-doubt threads manually. Use the QMNAME keyword to specify the name of the inactive queue manager in the command.

This command can be used to commit or back out the unit of recovery after you have determined what the correct decision is. The command resolves the shared portion of the unit of recovery only, any local messages are unaffected and remain locked until the queue manager restarts, or reconnects to CICS, IMS, or RRS batch.

## Chapter 15. Example recovery scenarios

This chapter describes procedures for recovering MQSeries after various error conditions. These error conditions are grouped in the following categories:

Table 3. Example recovery scenarios

Problem category	Problem	Where to look next
Shared queue problems	Queue both private and shared.	"Shared queue problems" on page 150
Active log problems	Dual logging is lost.	"Active log problems" on page 151
	Active log has stopped.	
	One or both copies of the active log data set are damaged.	
	Write errors on active log data set.	
	Active log is becoming full or is full.	
	Read errors on active log data set.	
Archive log problems	Insufficient DASD space to complete off-loading active log data sets.	"Archive log problems" on page 157
	Off-load task has terminated abnormally.	
	Archive data set allocation problem.	
	Read I/O errors on the archive data set during restart.	
BSDS problems	Error opening BSDS.	"BSDS problems" on page 160
	Log content does not correspond with BSDS information.	
	Both copies of the BSDS are damaged.	
	Unequal time stamps.	
	Dual BSDS data sets are out of synchronization.	
	I/O error on BSDS.	
Page set problems	Page set full.	"Page set problems" on page 164
	A page set has an I/O error.	
Coupling Facility and DB2 problems	Storage medium full.	"Coupling Facility and DB2 problems" on page 166
	DB2 system fails.	
	DB2 data-sharing group fails.	
	DB2 and the Coupling Facility fail.	
Unit of work problems	A long-running unit of work is encountered.	"Problems with long-running units of work" on page 169
IMS problems	An IMS application terminates abnormally.	"IMS-related problems" on page 170
	The IMS adapter cannot connect to MQSeries.	
	IMS not operational.	

### Shared queue problems

This section covers the following shared queue problems:

- “Queue is both private and shared”

#### Queue is both private and shared

##### Symptoms

MQSeries issues the following message:

```
CSQI063E +CSQ1 QUEUE queue-name IS BOTH PRIVATE AND SHARED
```

During restart, MQSeries discovered that a page set based queue and a shared queue of the same name coexist.

##### System action

Once restart processing has completed, any **MQOPEN** request to that queue name fails, indicating the coexistence problem.

##### System programmer action

None.

##### Operator action

Delete one version of the queue in order to allow processing of that queue name. If there are messages on the queue that must be kept, you can use the **MOVE QLOCAL** command to move them to the other queue.

## Active log problems

This section covers the following active log problems:

- “Dual logging is lost”
- “Active log stopped”
- “One or both copies of the active log data set are damaged” on page 152
- “Write I/O errors on an active log data set” on page 153
- “I/O errors occur while reading the active log” on page 153
- “Active log is becoming full or is full” on page 155

### Dual logging is lost

#### Symptoms

MQSeries issues the following message:

```
CSQJ004I +CSQ1 ACTIVE LOG COPY n INACTIVE, LOG IN SINGLE MODE,
      ENDRBA=...
```

Having completed one active log data set, MQSeries found that the subsequent (COPY *n*) data sets were not off-loaded or were marked stopped.

#### System action

MQSeries continues in single mode until off-loading has been completed, then returns to dual mode.

#### System programmer action

None.

#### Operator action

Check that the off-load is proceeding and is not waiting for a tape mount. It might be necessary to run the print log map utility to determine the state of all data sets. It might also be necessary to define additional data sets.

### Active log stopped

#### Symptoms

MQSeries issues the following message:

```
CSQJ030E +CSQ1 RBA RANGE startdba TO enddba NOT AVAILABLE IN ACTIVE
      LOG DATA SETS
```

#### System action

The active log data sets that contain the RBA range reported in message CSQJ030E are unavailable to MQSeries. The status of these logs is STOPPED in the BSDS. MQSeries will terminate with a dump.

#### System programmer action

This problem must be resolved before restarting MQSeries. The log RBA range must be available for MQSeries to be recoverable. An active log that is marked as STOPPED in the BSDS will never be reused or archived and this will create a hole in the log.

Look for messages that indicate why the log data set has stopped, and follow the instructions for those messages.

The BSDS active log inventory needs to be modified to reset the STOPPED status. To do this, follow this procedure after MQSeries has terminated:

1. Use the print log utility (CSQJU004) to obtain a copy of the BSDS log inventory. This shows the status of the log data sets.

## Active log problems

2. Use the DELETE function of the change log inventory utility (CSQJU003) to delete the active log data sets that are marked as STOPPED.
3. Use the NEWLOG function of CSQJU003 to add the active logs back into the BSDS inventory. The starting and ending RBA for each active log data set must be specified on the NEWLOG statement. (The correct values to use can be found from the print log utility report obtained in Step 1 on page 151.)
4. Rerun CSQJU004. The active log data sets that were marked as STOPPED will now be shown as NEW and NOT REUSABLE. These active logs will be archived in due course.
5. Restart MQSeries.

**Note:** If your MQSeries subsystem is running in dual BSDS mode, you must update both BSDS inventories.

## One or both copies of the active log data set are damaged

### Symptoms

MQSeries issues the following messages:

```
CSQJ102E +CSQ1 LOG RBA CONTENT OF LOG DATA SET DSNAME=...,  
          STARTRBA=..., ENDRBA=...,  
          DOES NOT AGREE WITH BSDS INFORMATION  
CSQJ232E +CSQ1 OUTPUT DATA SET CONTROL INITIALIZATION PROCESS FAILED
```

### System action

MQSeries startup processing is terminated.

### System programmer action

If one copy of the data set is damaged, carry out these steps:

1. Rename the damaged active log data set and define a replacement data set.
2. Copy the undamaged data set to the replacement data set.
3. Use the change log inventory utility to:
  - Remove information relating to the damaged data set from the BSDS.
  - Add information relating to the replacement data set to the BSDS.
4. Restart MQSeries.

If both copies of the active log data sets are damaged, the current page sets are available, **and MQSeries shut down cleanly**, carry out these steps:

1. Rename the damaged active log data sets and define replacement data sets.
2. Use the change log records utility to:
  - Remove information relating to the damaged data set from the BSDS.
  - Add information relating to the replacement data set to the BSDS.
3. Rename the current page sets and define replacement page sets.
4. Use CSQUTIL (FORMAT and RESETPAGE) to format the replacement page sets and copy the renamed page sets to them. The RESETPAGE function also resets the log information in the replacement page sets.

If MQSeries did not shut down cleanly, you must either restore your system from a previous known point of consistency, or perform a cold start (described in “Reinitializing MQSeries” on page 130).



**Operator action**

None.

**Write I/O errors on an active log data set****Symptoms**

MQSeries issues the following message:

```
CSQJ105E +CSQ1 csect-name LOG WRITE ERROR DSNAME=...,
        LOGRBA=..., ERROR STATUS=ccccffss
```

**System action**

MQSeries carries out these steps:

1. Marks the log data set that has the error as TRUNCATED in the BSDS.
2. Goes on to the next available data set.
3. If dual active logging is used, truncates the other copy at the same point.

The data in the truncated data set is off-loaded later, as usual.

The data set is not *stopped* and is reused on the next cycle.

**System programmer action**

None.

**Operator action**

If errors on this data set still exist, take MQSeries down after the next off-load. Then use Access Method Services (AMS) and the change log inventory utility to add a replacement. (For instructions, see "Changing the BSDS" on page 99.)

**I/O errors occur while reading the active log****Symptoms**

MQSeries issues the following message:

```
CSQJ106E +CSQ1 LOG READ ERROR DSNAME=..., LOGRBA=...,
        ERROR STATUS=ccccffss
```

**System action**

This depends on when the error occurred:

- If the error occurs during the off-load process, the process tries to read the RBA range from a second copy.
  - If no second copy exists, the active log data set is stopped.
  - If the second copy also has an error, only the original data set that triggered the off-load is stopped. The archive log data set is then terminated, leaving a gap in the archived log RBA range.
  - This message is issued:
 

```
CSQJ124E +CSQ1 OFFLOAD OF ACTIVE LOG SUSPENDED FROM
          RBA xxxxxx TO RBA xxxxxx DUE TO I/O ERROR
```
  - If the second copy is satisfactory, the first copy is not stopped.
- If the error occurs during recovery, MQSeries provides data from specific log RBAs requested from another copy or archive. If this is unsuccessful, recovery does not succeed, and MQSeries terminates abnormally.
- If the error occurs during restart, MQSeries terminates. All the copies of the active log data sets must be available to MQSeries.

## Active log problems

### System programmer action

Look for system messages, such as IEC prefixed messages, and try to resolve the problem using the recommended actions for these messages.

If the active log data set has been stopped, it is not used for logging. The data set is not deallocated; it is still used for reading. Even if the data set is not stopped, an active log data set that gives persistent errors should nevertheless be replaced.

### Operator action

None. You are not told explicitly whether the data set has been stopped.

## Replacing the data set

How you replace the data set depends on whether you are using single or dual active logging.

### *If you are using dual active logging:*

1. Ensure that the data has been saved.  
The data is saved on the other active log and this can be copied to a replacement active log.
2. Stop MQSeries and delete the data set in error using Access Method Services.
3. Redefine a new log data set using Access Method Services DEFINE so that you can write to it. Use DFDSS or Access Method Services REPRO to copy the good log into the redefined data set so that you have two consistent, correct logs again.
4. Use the change log inventory utility, CSQJU003, to update the information in the BSDS about the corrupt data set as follows:
  - a. Use the DELETE function to remove information about the corrupt data set.
  - b. Use the NEWLOG function to name the new data set as the new active log data set and give it the RBA range that was successfully copied.  
The DELETE and NEWLOG functions can be run in the same job step. Put the DELETE statement before NEWLOG statement in the SYSIN input data set.
5. Restart MQSeries.

### *If you are using single active logging:*

1. Ensure that the data has been saved.
2. Stop MQSeries.
3. Determine whether the data set with the error has been off-loaded:
  - a. Use the CSQJU003 utility to list information about the archive log data sets from the BSDS.
  - b. Search the list for a data set whose RBA range includes the RBA of the corrupt data set.
4. If the corrupt data set has been off-loaded, copy its backup in the archive log to a new data set. Then, skip to step 6.
5. If an active log data set is stopped, an RBA is not off-loaded. Use DFDSS or Access Method Services REPRO to copy the data from the corrupt data set to a new data set.

If further I/O errors prevent you from copying the entire data set, a gap occurs in the log.

**Note:** MQSeries restart will not be successful if a gap in the log is detected.

6. Use the change log inventory utility, CSQJU003, to update the information in the BSDS about the corrupt data set as follows:
  - a. Use the DELETE function to remove information about the corrupt data set.
  - b. Use the NEWLOG function to name the new data set as the new active log data set and to give it the RBA range that was successfully copied.

The DELETE and NEWLOG functions can be run in the same job step. Put the DELETE statement before NEWLOG statement in the SYSIN input data set.
7. Restart MQSeries.

### Active log is becoming full or is full

The active log can fill up for several reasons, for example, delays in off-loading and excessive logging.

#### Symptoms

An out-of-space condition on the active log has serious consequences. When the active log becomes full, the MQSeries subsystem halts processing until an off-load has been completed. If the off-load processing stops when the active log is full, the MQSeries subsystem can abend. Corrective action is required before MQSeries can be restarted.

Because of the serious implications of this event, the MQSeries subsystem issues the following warning message when the last available active log data set is 75% full:

```
CSQJ110E +CSQ1 LAST COPY $n$  ACTIVE LOG DATA SET IS  $nnn$  PERCENT FULL
```

and reissues the message after each additional 5% of the data set space is filled. Each time the message is issued, the off-load process is started.

If the active log fills to capacity, MQSeries issues the following message:

```
CSQJ111A +CSQ1 OUT OF SPACE IN ACTIVE LOG DATA SETS
```

and an off-load is started. The MQSeries subsystem then halts processing until an off-load has been completed.

#### System action

MQSeries waits for an available active log data set before resuming normal MQSeries processing. Normal shutdown, with either QUIESCE or FORCE, is not possible because the shutdown sequence requires log space to record system events related to shutdown (for example, checkpoint records). If the off-load processing stops when the active log is full, MQSeries forces itself to stop using an X'6C6' abend; restart in this case requires special attention. For more details, see the *MQSeries for OS/390 Problem Determination Guide*.

#### System programmer action

Additional active log data sets can be provided as required before restarting MQSeries. This permits MQSeries to continue its normal operation while the error causing the off-load problems is corrected. To add new active log data sets, use the change log inventory utility (CSQJU003) when MQSeries is not active. For more details about adding new active log data sets, see "Changing the BSDS" on page 99.

You should also consider increasing the number of logs by:

1. Making sure MQSeries is stopped, then using the Access Method Services DEFINE command to define a new active log data set.

## Active log problems

2. Defining the new active log data set in the BSDS using the change log inventory utility (CSQJU003).

Restarting MQSeries: off-load starts automatically during startup, and work in progress when MQSeries was forced to stop is recovered.

### **Operator action**

Check whether the off-load process is waiting for a tape drive. If it is, mount the tape. If you cannot mount the tape, force MQSeries to stop by using OS/390 CANCEL.

## Archive log problems

This section covers the following archive log problems:

- “Allocation problems”
- “Off-load task terminated abnormally”
- “Insufficient DASD space to complete off-load processing” on page 158
- “Read I/O errors on the archive data set while MQSeries is restarting” on page 159

### Allocation problems

#### Symptoms

MQSeries issues the following message:

```
CSQJ103E +CSQ1 LOG ALLOCATION ERROR DSNAME=dsname,
        ERROR STATUS=eeeeiii, SMS REASON CODE=sss
```

OS/390 dynamic allocation provides the ERROR STATUS. If the allocation was for off-load processing, the following message is also displayed:

```
CSQJ115E +CSQ1 OFFLOAD FAILED, COULD NOT ALLOCATE AN ARCHIVE
        DATA SET
```

#### System action

The following actions take place:

- If the input is needed for recovery, recovery is not successful, and MQSeries will abend.
- If the active log had become full and an off-load was scheduled but not completed, off-load tries again the next time it is triggered. The active log does not reuse a data set that has not yet been archived.

#### System programmer action

None.

#### Operator action

Check the allocation error code for the cause of the problem, and correct it. Ensure that drives are available, and either restart or wait for the off-load to be retried. Be careful if a DFP/DFSMS ACS user-exit filter has been written for an archive log data set, because this can cause a device allocation error when the MQSeries subsystem tries to read the archive log data set.

### Off-load task terminated abnormally

#### Symptoms

No specific MQSeries message is issued for write I/O errors.

Only an OS/390 error recovery program message appears. If you get MQSeries message CSQJ128E, the off-load task has terminated abnormally and you should consult the *MQSeries for OS/390 Messages and Codes* manual.

#### System action

The following actions take place:

- Off-load abandons the output data set; no entry is made in the BSDS.
- Off-load dynamically allocates a new archive and restarts off-loading from the point at which it was previously triggered.

## Archive log problems

- If an error occurs on the new data set:
  - In dual archive mode, this message is generated and the off-load processing changes to single mode:  
CSQJ114I +CSQ1 ERROR ON ARCHIVE DATA SET, OFFLOAD  
CONTINUING WITH ONLY ONE ARCHIVE DATA SET BEING  
GENERATED
  - In single archive mode, the output data set is abandoned. Another attempt to off-load this RBA range is made the next time off-load is triggered.
  - The active log does not wrap around; if there are no more active logs, data is not lost.

### System programmer action

None.

### Operator action

Ensure that off-load is allocated on a reliable drive and control unit.

## Insufficient DASD space to complete off-load processing

### Symptoms

While off-loading the active log data sets to DASD, the process terminates unexpectedly. MQSeries issues the following message:

```
CSQJ128E +CSQ1 LOG OFF-LOAD TASK FAILED FOR ACTIVE LOG nnnn
```

The error is preceded by OS/390 messages IEC030I, IEC031I, or IEC032I.

### System action

MQSeries de-allocates the data set on which the error occurred. If MQSeries is running in dual archive mode, MQSeries changes to single archive mode and continues the off-load. If the off-load cannot be completed in single archive mode, the active log data sets cannot be off-loaded, and the state of the active log data sets remains NOT REUSABLE. Another attempt to off-load the RBA range of the abandoned active log data sets is made the next time the off-load process is triggered.

### System programmer action

Quiesce the MQSeries subsystem (using +CSQ1 STOP QMGR MODE(QUIESCE)) to restrict logging activity until the OS/390 abend is resolved.

The most likely causes of these symptoms are:

- The size of the archive log data set is too small to contain the data from the active log data sets during off-load processing. All the secondary space allocations have been used. This condition is normally accompanied by OS/390 message IEC030I.

To solve the problem, either increase the primary or secondary allocations (or both) for the archive log data set (in the CSQ6ARVP system parameters), or reduce the size of the active log data set. If the data to be off-loaded is particularly large, you can mount another online storage volume or make one available to MQSeries.

- All available space on the DASD volumes to which the archive data set is being written has been exhausted. This condition is normally accompanied by OS/390 message IEC032I.

To solve the problem, make more space available on the DASD volumes, or make another online storage volume available for MQSeries.

## Archive log problems

- The primary space allocation for the archive log data set (as specified in the CSQ6ARVP system parameters) is too large to allocate to any available online DASD device. This condition is normally accompanied by OS/390 message IEC032I.

To solve the problem, make more space available on the DASD volumes, or make another online storage volume available for MQSeries. If this is not possible, you must adjust the value of PRIQTY in the CSQ6ARVP system parameters to reduce the primary allocation. (For details, see the *MQSeries for OS/390 System Setup Guide*.)

**Note:** If the primary allocation is reduced, the size of the secondary space allocation might have to be increased to avoid future abends.

### Operator action

None.

## Read I/O errors on the archive data set while MQSeries is restarting

### Symptoms

No specific MQSeries message is issued; only the OS/390 error recovery program message appears.

### System action

This depends on whether a second copy exists:

- If a second copy exists, it is allocated and used.
- If a second copy does not exist, restart is not successful.

### System programmer action

None.

### Operator action

Try to restart, using a different drive.

### BSDS problems

For background information about the bootstrap data set (BSDS), see the *MQSeries for OS/390 Concepts and Planning Guide*.

This section describes the following BSDS problems:

- “Error occurs while opening the BSDS”
- “Log content does not agree with the BSDS information” on page 161
- “Both copies of the BSDS are damaged” on page 161
- “Unequal time stamps” on page 162
- “Out of synchronization” on page 162
- “I/O error” on page 163

Normally, there are two copies of the BSDS, but if one is damaged, MQSeries immediately changes to single BSDS mode. However, the damaged copy of the BSDS must be recovered before restart. If you are in single mode and damage the only copy of the BSDS, or if you are in dual mode and damage both copies, see “Recovering the BSDS” on page 102.

This section covers some of the BSDS problems that can occur at startup. Problems *not* covered here include:

- +CSQ1 RECOVER BSDS command errors (messages CSQJ301E - CSQJ307I)
- Change log inventory utility errors (message CSQJ123E)
- Errors in the BSDS backup being dumped by off-load (message CSQJ125E)

For information about those problems, see the *MQSeries for OS/390 Messages and Codes* manual.

## Error occurs while opening the BSDS

### Symptoms

MQSeries issues the following message:

```
CSQJ100E +CSQ1 ERROR OPENING BSDSn DSNAME=..., ERROR STATUS=eei
```

where *eei* is the VSAM return code. For information about VSAM codes, see the *DFSMS/MVS Macro Instructions for Data Sets* manual. For an explanation of this message, see the *MQSeries for OS/390 Messages and Codes* manual.

### System action

During system initialization, the startup is terminated.

During a +CSQ1 RECOVER BSDS command, the system continues in single BSDS mode.

### System programmer action

None.

### Operator action

Carry out these steps:

1. Run the print log map utility on both copies of the BSDS, and compare the lists to determine which copy is accurate or current.
2. Rename the data set that had the problem, and define a replacement for it.
3. Copy the accurate data set to the replacement data set, using Access Method Services.



- Restart MQSeries.

## Log content does not agree with the BSDS information

### Symptoms

MQSeries issues the following message:

```
CSQJ102E +CSQ1 LOG RBA CONTENT OF LOG DATA SET DSNAME=...,
          STARTRBA=..., ENDRBA=...,
          DOES NOT AGREE WITH BSDS INFORMATION
```

This message indicates that the change log inventory utility was used incorrectly or that a down-level data set is being used.

### System action

MQSeries startup processing is terminated.

### System programmer action

None.

### Operator action

Run the print log map utility and the change log inventory utility to print and correct the contents of the BSDS.

## Both copies of the BSDS are damaged

### Symptoms

MQSeries issues the following messages:

```
CSQJ107E +CSQ1 READ ERROR ON BSDS
          DSNAME=... ERROR STATUS=0874
CSQJ117E +CSQ1 REG8 INITIALIZATION ERROR READING BSDS
          DSNAME=... ERROR STATUS=0874
CSQJ119E +CSQ1 BOOTSTRAP ACCESS INITIALIZATION PROCESSING FAILED
```

### System action

MQSeries startup processing is terminated.

### System programmer action

Carry out these steps:

- Rename the data set, and define a replacement for it.
- Locate the BSDS associated with the most recent archive log data set, and copy it to the replacement data set.
- Use the print log map utility to print the contents of the replacement BSDS.
- Use the print log records utility to print a summary report of the active log data sets missing from the replacement BSDS, and to establish the RBA range.
- Use the change log inventory utility to update the missing active log data set inventory in the replacement BSDS.
- If dual BSDS data sets had been in use, copy the updated BSDS to the second copy of the BSDS.
- Restart MQSeries.

### Operator action

None.

## BSDS problems

### Unequal time stamps

#### Symptoms

MQSeries issues the following message:

```
CSQJ120E +CSQ1 DUAL BSDS DATA SETS HAVE UNEQUAL TIME STAMPS,  
        SYSTEM BSDS1=...,BSDS2=...,  
        UTILITY BSDS1=...,BSDS2=...
```

The possible causes are:

- One copy of the BSDS has been restored. All information on the restored BSDS is down-level. The down-level BSDS has the lower time stamp.
- One of the volumes containing the BSDS has been restored. All information on the restored volume is down-level. If the volume contains any active log data sets or MQSeries data, they are also down-level. The down-level volume has the lower time stamp.
- Dual logging has degraded to single logging, and you are trying to start without recovering the damaged log.
- The MQSeries subsystem terminated abnormally after updating one copy of the BSDS but before updating the second copy.

#### System action

MQSeries startup is terminated.

#### System programmer action

None.

#### Operator action

Carry out these steps:

1. Run the print log map utility on both copies of the BSDS, compare the lists to determine which copy is accurate or current.
2. Rename the down-level data set and define a replacement for it.
3. Copy the good data set to the replacement data set, using Access Method Services.
4. If applicable, determine whether the volume containing the down-level BSDS has been restored. If it has been restored, all data on that volume, such as the active log data, is also down-level.

If the restored volume contains active log data and you were using dual active logs on separate volumes, you need to copy the current version of the active log to the down-level log data set. "Recovering logs" on page 93 tells you how to do this.

### Out of synchronization

#### Symptoms

MQSeries issues the following message:

```
CSQJ122E +CSQ1 DUAL BSDS DATA SETS ARE OUT OF SYNCHRONIZATION
```

The system time stamps of the two data sets are identical. Differences can exist if operator errors occurred while the change log inventory utility was being used. (For example, the change log inventory utility was only run on one copy.) The change log inventory utility sets a private time stamp in the BSDS control record when it starts, and a close flag when it ends.

MQSeries checks the change log inventory utility time stamps and, if they are different, or they are the same but one close flag is not set, MQSeries compares the copies of the BSDSs. If the copies are different, CSQJ122E is issued.

**System action**

MQSeries startup is terminated.

**System programmer action**

None.

**Operator action**

Carry out these steps:

1. Run the print log map utility on both copies of the BSDS, and compare the lists to determine which copy is accurate or current.
2. Rename the data set that had the problem, and define a replacement for it.
3. Copy the accurate data set to the replacement data set, using access method services.
4. Restart MQSeries.

**I/O error**

**Symptoms**

MQSeries changes to single BSDS mode and issues the user message:

```
CSQJ126E +CSQ1 BSDS ERROR FORCED SINGLE BSDS MODE
```

This is followed by one of these messages:

```
CSQJ107E +CSQ1 READ ERROR ON BSDS
        DSNAME=... ERROR STATUS=...
```

```
CSQJ108E +CSQ1 WRITE ERROR ON BSDS
        DSNAME=... ERROR STATUS=...
```

**System action**

The BSDS mode changes from dual to single.

**System programmer action**

None.

**Operator action**

Carry out these steps:

1. Use Access Method Services to rename or delete the damaged BSDS and to define a new BSDS with the same name as the BSDS that had the error. Control statements can be found in job CSQ4BSDS in thlqual.SCSQPROC.
2. Issue the MQSeries command +CSQ1 RECOVER BSDS to make a copy of the good BSDS in the newly allocated data set and reinstate dual BSDS mode. See also "Recovering the BSDS" on page 102.

### Page set problems

This section covers the problems that you might encounter with page sets:

- “Page set I/O errors” describes what happens if a page set is damaged.
- “Page set full” on page 165 describes what happens if there is not enough space on the page set for any more MQI operations.

### Page set I/O errors

#### Problem

A page set has an I/O error.

#### Symptoms

This message is issued:

```
CSQP004E +CSQ1 csect-name I/O ERROR STATUS ret-code  
PSID psid RBA rba
```

#### System action

MQSeries terminates abnormally.

#### System programmer action

None.

#### Operator action

Repair the I/O error cause.

If none of the page sets are damaged, restart MQSeries. MQSeries automatically restores the page set to a consistent state from the logs.

If one or more page sets are damaged:

1. Rename the damaged page sets and define replacement page sets.
2. Copy the most recent backup page sets to the replacement page sets.
3. Restart MQSeries. MQSeries automatically applies any updates that are necessary from the logs.

You cannot restart MQSeries if page set zero is not available. However, if one of the other page sets is not available, you can comment out the page set DD statement in the MQSeries start-up JCL procedure. This lets you defer recovery of the defective page set, enabling other users to continue accessing MQSeries.

**When you add the page set back to the JCL procedure, system restart will read the log from the point where the page set was removed from the JCL to the end of the log. This could take a long time if a lot of data has been logged.**

A reason code of MQRC\_PAGESET\_ERROR is returned to any application that tries to access a queue defined on a page set that is not available. When you have restored the defective page set, restore its associated DD statement and restart MQSeries.

The operator actions described here are only possible if all log data sets are available. If your log data sets are lost or damaged, see “Restarting if you have lost your log data sets” on page 126.

## Page set full

### Problem

There is not enough space on a page set for one of the following:

- **MQPUT** or **MQPUT1** calls to be completed
- Object manipulation commands to be completed (for example, **DEFINE QLOCAL**)
- **MQOPEN** calls for dynamic queues to be completed

### Symptoms

The request fails with reason code **MQRC\_STORAGE\_MEDIUM\_FULL**.

The queue manager is unable to complete the request because there is not enough space remaining on the page set.

The cause of this problem could be due to messages accumulating on a transmission queue because they cannot be sent to another system.

### System action

Further requests that use this page set are blocked until enough messages are removed or objects deleted to make room for the new incoming requests.

### Operator action

Use the MQSeries command **DISPLAY USAGE PSID(\*)** to identify which page set is full.

### System programmer action

You can either enlarge the page set involved or reduce the loading on that page set by moving queues to another page set. See “Chapter 10. Managing page sets” on page 105 for more information about these tasks. If the cause of the problem is messages accumulating on the transmission queue, consider starting distributed queuing to transmit the messages.

### Coupling Facility and DB2 problems

This section covers the problems that you might encounter with the Coupling Facility and DB2:

- “Storage medium full”
- “A DB2 system fails”
- “A DB2 data-sharing group fails” on page 167
- “DB2 and the Coupling Facility fail” on page 168

#### Storage medium full

##### Problem

A Coupling Facility structure is full.

##### Symptoms

If a queue structure becomes full, return code MQRC\_STORAGE\_MEDIUM\_FULL is returned to the application.

If the administration structure becomes full, the exact symptoms depend on which processes experience the error, they might range from no responses to CMDSCOPE(GROUP) commands to queue manager failure as a result of problems during commit processing.

##### System programmer action

You might use MQSeries to inhibit **MQPUT** operations to some of the queues in the structure to prevent applications from writing more messages, start more applications to get messages from the queues, or quiesce some of the applications that are putting messages to the queue.

Alternatively XES facilities can be used to alter the structure size in place. The OS/390 command `SETXCF START,ALTER,CFNAME=cfname,SIZE=newsiz` alters the size of the structure to *newsiz*, where *newsiz* is a value that is less than the value of MAXSIZE specified on the CFRM policy for the structure, but greater than the current Coupling Facility size.

You can monitor the utilization of a Coupling Facility structure with the MQSeries DISPLAY GROUP command.

#### A DB2 system fails

If a DB2 subsystem that MQSeries is connected to fails, MQSeries attempts to reconnect to the subsystem and continue working. If you specified a DB2 group attach name in the QSGDATA parameter of the CSQ6SYSP system parameter module, MQSeries reconnects to another active DB2 that is a member of the same data-sharing group as the failed DB2, if one is available on the same OS/390 image.

There are some queue manager operations that will not work while MQSeries is not connected to DB2. These are:

- Delete of a shared queue or group object definition.
- Alter or **MQSET** of a shared queue or group object definition. The restriction of **MQSET** on shared queues means that operations such as triggering or generation of performance events do not work correctly.
- Define of new shared queues or group objects.
- Display of shared queues or group objects.
- Starting, stopping, or other actions for shared channels.

## Coupling Facility and DB2 problems

However, other MQSeries API operations continue to function as normal for shared queues, and all MQSeries operations can be performed against the queue manager private versions (COPY objects) built from GROUP objects. Similarly, any shared channels that are running will continue normally until they end or have an error, when they will go into retry state.

When MQSeries reconnects to DB2, resynchronization is performed between the queue manager and DB2. This involves notifying the queue manager of new objects that have been defined in DB2 while it was disconnected (other queue managers might have been able to continue working as normal on other OS/390 images through other DB2 subsystems), and updating object attributes of shared queues that have changed in DB2. Any shared channels in retry state will be recovered.

If a DB2 fails, it might have owned locks on DB2 resources at the time of failure. In some cases, this might make certain MQSeries objects unavailable to other queue managers that are not otherwise affected. To resolve this, restart the failed DB2 so that it can perform recovery processing and release the locks.

### A DB2 data-sharing group fails

If an entire DB2 data-sharing group fails, recovery might be to the time of failure, or to a previous point in time.

In the case of recovery to the point of failure, MQSeries reconnects when DB2 has been recovered, the resynchronization process takes place and normal queue manager function is resumed.

However, if DB2 is recovered to a previous point in time, there might be inconsistencies between the actual queues in the Coupling Facility structures and the DB2 view of those queues. For example, at the point in time DB2 is recovered to, a queue existed that has since been deleted and its location in the Coupling Facility structure reused by the definition of a new queue that now contains messages.

If you find yourself in this sort of situation, you must stop all the queue managers in the queue-sharing group, clear out the Coupling Facility structures, and restart the queue managers. You must then use MQSC commands to define any missing objects. To do this, use the following procedure:

1. Prevent MQSeries from reconnecting to DB2 by starting DB2 in utility mode, or by altering security profiles.
2. If you have any important messages on shared queues, you might be able to off-load them using the COPY function of the CSQUTIL utility program, but this might not work.
3. Terminate all queue managers.
4. Use the OS/390 command SETXCF FORCE,STRUCTURE,STRNAME= to clear all structures.
5. Restore DB2 to a historical point in time.
6. Reestablish queue manager access to DB2.
7. Restart the queue managers.
8. Recover the MQSeries definitions from back-up copies.
9. Reload any off-loaded messages to the shared queues.

## Coupling Facility and DB2 problems

When the queue managers restart, they attempt to resynchronize local COPY objects with the DB2 GROUP objects. This might cause MQSeries to attempt to do the following:

- Create COPY objects for old GROUP objects that existed at the point in time DB2 has recovered to
- Delete COPY objects for GROUP objects that were created since the point in time DB2 has recovered to and so do not exist in the database

The DELETE of COPY objects is attempted with the NOPURGE option, so it will fail for queue managers that still have messages on these COPY queues.

## DB2 and the Coupling Facility fail

If the Coupling Facility fails, the queue managers and DB2 will also fail.

Recover DB2 using DB2 recovery procedures. When DB2 has been restarted, you can restart the queue managers.

All messages on shared queues that were in Coupling Facility structures affected by the Coupling Facility failure will have been lost. The queue object definitions will have been restored by DB2 and can continue to be used.



---

## Problems with long-running units of work

This section explains what to do if you encounter a long-running unit of work during restart. In this context, this means a unit of work that has been active for a long period of time (possibly days or even weeks) so that the origin RBA of the unit of work is outside the scope of the current active logs. This means that restart could take a long time, because all of the log records relating to the unit of work have to be read, which might involve reading archive logs.

### Old unit of work found during restart

**Problem**

A unit of work with an origin RBA that predates the oldest active log has been detected during restart.

**Symptoms**

MQSeries issues the following message:

```
CSQR020I +CSQ1 OLD UOW FOUND
```

**System action**

Information about the unit of work is displayed, and message CSQR021D is issued, requesting a response from the operator.

**System programmer action**

None.

**Operator action**

Decide whether to commit the unit of work or not. If you choose not to commit the unit of work, it will be handled by normal restart recovery processing. Because the unit of work is old, this is likely to involve using the archive log, and so will take longer to complete.

### IMS-related problems

This section includes plans for problems that you might encounter in the IMS environment:

- “IMS is unable to connect to MQSeries”
- “IMS application problem”
- “IMS is not operational” on page 171

#### IMS is unable to connect to MQSeries

##### Problem

The IMS adapter cannot connect to MQSeries.

##### Symptoms

IMS remains operative. The IMS adapter issues these messages for control region connect:

```
CSQQ001I
CSQQ002E
CSQQ003E
CSQQ004E
CSQQ005E
CSQQ007E
```

For details, see the *MQSeries for OS/390 Messages and Codes* manual.

If an IMS application program tries to access MQSeries while the IMS adapter cannot connect, it can either receive a completion code and reason code or terminate abnormally. This depends on the value of the REO option in the SSM member of IMS PROCLIB.

##### System action

All connection errors are also reported in the IMS message DFS3611.

##### System programmer action

None.

##### Operator action

Analyze and correct the problem, then restart the connection with the IMS command:

```
/START SUBSYS subsysname
```

IMS requests the adapter to resolve in-doubt units of recovery.

#### IMS application problem

##### Problem

An IMS application terminates abnormally.

##### Symptoms

The following message is sent to the user's terminal:

```
DFS555I TRANSACTION tran-id ABEND abcode
MSG IN PROCESS: message data:
```

where *tran-id* represents any IMS transaction that is terminating abnormally and *abcode* is the abend code.

##### System action

IMS requests the adapter to resolve the unit of recovery. IMS remains connected to MQSeries.

**System programmer action**

None.

**Operator action**

As indicated in message DFS554A on the IMS master terminal.

**IMS is not operational****Problem**

IMS is not operational.

**Symptoms**

More than one symptom is possible:

- IMS waits or loops  
Because MQSeries cannot detect a wait or loop in IMS, you must find the origin of the wait or loop. This can be IMS, IMS applications, or the IMS adapter.
- IMS terminates abnormally.
  - See the manuals *IMS/ESA Messages and Codes* and *IMS/ESA Failure Analysis Structure Tables* for more information.
  - If threads are connected to MQSeries when IMS terminates, MQSeries issues message CSQ3201E. This message indicates that MQSeries end-of-task (EOT) routines have been run to clean up and disconnect any connected threads.

**System action**

MQSeries detects the IMS error and:

- Backs out in-flight work.
- Saves in-doubt units of recovery to be resolved when IMS is reconnected.

**System programmer action**

None.

**Operator action**

Resolve and correct the problem that caused IMS to terminate abnormally, then carry out an emergency restart of IMS. The emergency restart:

- Backs out in-flight transactions that changed IMS resources.
- Remembers the transactions with access to MQSeries that might be in doubt.

It might be necessary to restart the connection to MQSeries with the IMS command:

`/START SUBSYS subsysname`

During startup, IMS requests the adapter to resolve in-doubt units of recovery.

### Hardware problems

If a hardware error causes data to be unreadable on your subsystem, MQSeries can still be recovered by using the *media recovery* technique:

1. To recover the data, you need a backup copy of the data. Use DFDSS or Access Method Services REPRO regularly to make a copy of your data.
2. Reinstall the most recent backup copy.
3. Restart MQSeries.

The more recent your backup copy, the more quickly your subsystem can be made available again.

When MQSeries restarts, it uses the archive logs to reinstate changes made since the backup copy was taken. You must keep sufficient archive logs to enable MQSeries to reinstate the subsystem fully. Do not delete archive logs until there is a backup copy that includes all the changes in the log.

---

## Part 6. Using the MQSeries Utilities

<b>Chapter 16. Using the MQSeries utilities</b> . . . . .	175		<b>Chapter 22. The dead-letter queue handler utility (CSQUDLQH).</b> . . . . .	227
How to read syntax diagrams . . . . .	176		Invoking the DLQ handler . . . . .	227
<b>Chapter 17. MQSeries utility program (CSQUTIL)</b> . . . . .	179		Data definition statements . . . . .	228
Invoking the MQSeries utility program . . . . .	180		Sample JCL . . . . .	228
PARM parameters . . . . .	180		The DLQ handler rules table . . . . .	228
Monitoring the progress of the MQSeries utility program . . . . .	182		Control data . . . . .	229
Formatting page sets (FORMAT) . . . . .	183		Rules (patterns and actions) . . . . .	230
Expanding a page set (COPYPAGE) . . . . .	185		The pattern-matching keywords . . . . .	230
Copying a page set and resetting the log (RESETPAGE) . . . . .	187		The action keywords . . . . .	231
Issuing commands to MQSeries (COMMAND) . . . . .	190		Rules table conventions . . . . .	233
Producing a list of MQSeries define commands (SDEFS) . . . . .	195		Processing the rules table . . . . .	235
Copying queues into a data set while the queue manager is running (COPY) . . . . .	198		Ensuring that all DLQ messages are processed . . . . .	236
Copying queues into a data set while the queue manager is not running (SCOPY) . . . . .	201		An example DLQ handler rules table . . . . .	237
Emptying a queue of all messages (EMPTY) . . . . .	204			
Restoring messages from a data set to a queue (LOAD) . . . . .	206			
<b>Chapter 18. The change log inventory utility (CSQJU003)</b> . . . . .	209			
Invoking the CSQJU003 utility . . . . .	209			
Data definition (DD) statements . . . . .	209			
Multiple statement operation . . . . .	210			
Adding information about a data set to the BSDS (NEWLOG) . . . . .	211			
Deleting information about a data set from the BSDS (DELETE) . . . . .	214			
Supplying a password for archive log data sets (ARCHIVE) . . . . .	215			
Controlling the next restart (CRESTART) . . . . .	216			
Setting checkpoint records (CHECKPT) . . . . .	217			
Updating the highest written log RBA (HIGHRBA) . . . . .	218			
<b>Chapter 19. The print log map utility (CSQJU004)</b> . . . . .	219			
Invoking the CSQJU004 utility . . . . .	219			
Data definition statements . . . . .	219			
<b>Chapter 20. The log print utility (CSQ1LOGP)</b> . . . . .	221			
Invoking the CSQ1LOGP utility . . . . .	221			
Input control parameters . . . . .	222			
Output . . . . .	223			
<b>Chapter 21. The queue-sharing group utility (CSQ5PQSG)</b> . . . . .	225			
Invoking the queue-sharing group utility . . . . .	225			
Data definition statements . . . . .	225			
Keywords and parameters . . . . .	225			
Example . . . . .	226			



## Chapter 16. Using the MQSeries utilities

This chapter introduces the MQSeries utility programs that are provided to help you perform various administrative tasks. The utility programs are described in the following chapters. Table 4 summarizes what you can do with these utilities.

Table 4. A summary of MQSeries utilities

Name	Purpose		See page
CSQUTIL (MQSeries utility program)	<i>Managing page sets</i>		
	Format VSAM data sets as MQSeries page sets.	FORMAT	183
	Copy MQSeries page sets.	COPYPAGE	185
	Copy MQSeries page sets and reset the log information.	RESETPAGE	187
CSQUTIL (MQSeries utility program)	<i>Issuing commands</i>		
	Issue MQSeries commands from a sequential data set.	COMMAND	190
	Produce a set of DEFINE commands for objects.	COMMAND	192
	Produce a client channel definition file.	COMMAND	193
	Produce a set of DEFINE commands for objects (offline).	SDEFS	195
CSQUTIL (MQSeries utility program)	<i>Managing queues</i>		
	Copy contents of a queue to a data set.	COPY	198
	Copy contents of a queue to a data set (offline).	SCOPY	201
	Delete contents of a queue.	EMPTY	204
	Restore contents of a queue.	LOAD	206
CSQUCVX (Data conversion exit utility)	Generate data conversion exit routines. For information about the CSQUCVX utility, see the <i>MQSeries Application Programming Guide</i> .		
CSQJU003 (Change log inventory utility)	Add active or archive log data sets.	NEWLOG	211
	Delete active or archive log data sets.	DELETE	214
	Supply passwords for archive logs.	ARCHIVE	215
	Control the next restart of MQSeries.	CRESTART	216
	Set checkpoint records.	CHECKPT	217
	Update the highest written log RBA.	HIGHRBA	218
CSQJU004 (Print log map utility)	List information about the log.		219
CSQ1LOGP (Log print utility)	Print the log.		221
CSQ5PQSG (MQSeries table update utility)	Add and remove queue-sharing group and queue manager entries in the MQSeries tables held in the shared DB2 data-sharing group.		225
CSQUDLQH (Dead-letter queue handler utility)	Process messages on the dead-letter queue.		227

## Utilities

These utilities are located in the thlqual.SCSQAUTH or thlqual.SCSQLOAD MQSeries load libraries. Include the appropriate MQSeries language load library thlqual.SCSQANLx (where x is the language letter) in the STEPLIB concatenation before thlqual.SCSQAUTH or thlqual.SCSQLOAD. The utility control statements are available only in U.S. English. In some cases, the DB2 library db2qual.SDSNLOAD is also needed.

## How to read syntax diagrams

This book contains syntax diagrams (sometimes referred to as “railroad” diagrams).

Each syntax diagram begins with a double right arrow and ends with a right and left arrow pair. Lines beginning with a single right arrow are continuation lines. You read a syntax diagram from left to right and from top to bottom, following the direction of the arrows.

Other conventions used in syntax diagrams are:

*Table 5. How to read syntax diagrams*

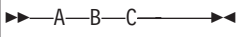
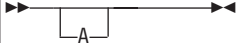

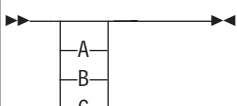
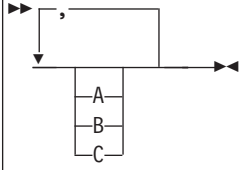
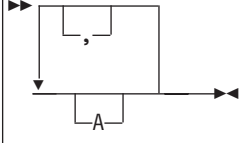
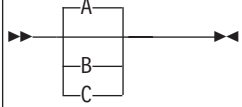
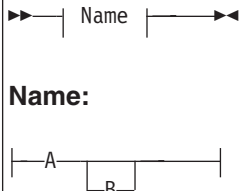
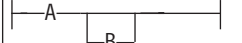
Convention	Meaning
	You must specify values A, B, and C. Required values are shown on the main line of a syntax diagram.
	You may specify value A. Optional values are shown below the main line of a syntax diagram.
	Values A, B, and C are alternatives, one of which you must specify.
	Values A, B, and C are alternatives, one of which you may specify.



Table 5. How to read syntax diagrams (continued)

Convention	Meaning
	<p>You may specify one or more of the values A, B, and C. Any required separator for multiple or repeated values (in this example, the comma (,)) is shown on the arrow.</p>
	<p>You may specify value A multiple times. The separator in this example is optional.</p>
	<p>Values A, B, and C are alternatives, one of which you may specify. If you specify none of the values shown, the default A (the value shown above the main line) is used.</p>
 <p><b>Name:</b></p> 	<p>The syntax fragment Name is shown separately from the main syntax diagram.</p>
<p>Punctuation and uppercase values</p>	<p>Specify exactly as shown.</p>
<p>Lowercase values (for example, <i>name</i>)</p>	<p>Supply your own text in place of the <i>name</i> variable.</p>

## Syntax diagrams

---

## Chapter 17. MQSeries utility program (CSQUTIL)

The CSQUTIL utility program is provided with MQSeries to help you to perform backup, restoration, and reorganization tasks, and to issue MQSeries commands. Through this utility program, you can invoke functions in these groups:

### Page set management

These functions enable you to manage MQSeries page sets. You can format data sets as page sets, you can increase the size of page sets and, if required, reset the log information contained in a page set. The page set must not belong to a queue manager that is currently running.

### Command management

These functions enable you to:

- Issue commands to MQSeries
- Produce a list of define commands describing the objects in your MQSeries subsystem

### Queue management

These functions enable you to back up and restore queues and page sets or to copy queues and page sets to another MQSeries system. You can use these functions to reset your MQSeries subsystem or for migrating from one MQSeries subsystem to another.

Specifically, you can:

- Copy messages from a queue to a data set
- Delete messages from a queue
- Restore previously copied messages to their respective queues

The scope of these functions can be either:

- A *queue*, in which case the function operates on all messages in the specified queue.
- A *page set*, in which case the function operates on all the messages, in all the queues, on the specified page set.

You should use these functions only for your own queues; do not use them for system queues (those with names beginning SYSTEM).

All of the page set management functions and some of the other functions operate while the queue manager is not running; for these therefore, you do not need any special authorization other than the appropriate access to the page set data sets. For the functions that operate while the queue manager is running, CSQUTIL runs as an ordinary OS/390 batch MQSeries program, issuing commands through the command server and using the MQSeries API to access queues.

You need the necessary authority to use the command server queues (SYSTEM.COMMAND.INPUT, SYSTEM.COMMAND.REPLY.MODEL, and SYSTEM.CSQUTIL.\*), to use the MQSC DISPLAY commands, and to use the MQSeries API to access any queues that you wish to manage. See the usage notes for each function for more information.

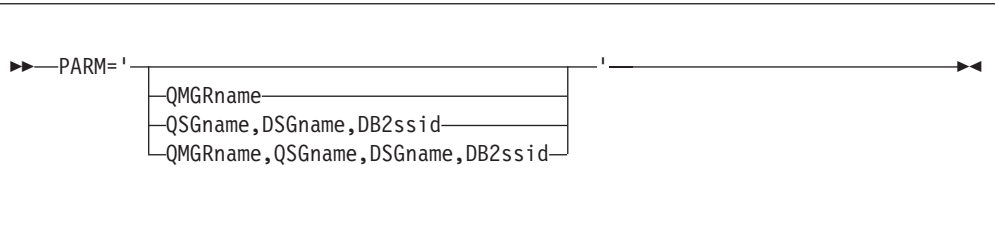
## Invoking the MQSeries utility program

The CSQUTIL utility program runs as an OS/390 batch program, below the 16 MB storage line. Specify the resources that the utility is to work with in the PARM parameter of the EXEC statement of the JCL.

```
// EXEC PGM=CSQUTIL,PARM=
```

Figure 56. How to invoke the CSQUTIL utility program

where PARM= expands to :



## PARM parameters

### QMGRname

Specifies the 1- to 4- character name of the queue manager or queue-sharing group to which CSQUTIL is to connect.

If you specify the name of a queue-sharing group, CSQUTIL connects to any queue manager in that group

### QSGname

Specifies the 1- to 4- character name of the queue-sharing group from which CSQUTIL is to extract definitions.

### DSGname

Specifies the 8-character name of the DB2 data-sharing group from which CSQUTIL is to extract definitions.

### DB2ssid

Specifies the 4-character name, or group attach name, of the DB2 database subsystem to which CSQUTIL is to attach for stand-alone functions.

### Which PARM parameters do you need?

Figure 56 shows that you can specify one of four options on the PARM statement. The option you specify depends on the function you need to implement, as follows:

- Use PARM= (or omit it all together) if you are using only offline functions, and not QSGDISP(GROUP) or QSGDISP(SHARED).
- Use PARM='QMGRname' only if you intend to use functions that require the queue manager to be running, such as COPY and COMMAND.
- Use PARM='QSGname,DSGname,DB2ssid' if you intend to use the SDEFS function with either QSGDISP(GROUP) or QSGDISP(SHARED) specified. This is because CSQUTIL requires access to DB2 to perform the SDEFS function in this situation.
- Use PARM='QMGRname,QSGname,DSGname,DB2ssid' if you intend to combine the previous two functions in one CSQUTIL job.

If you specify a queue manager name as blanks, CSQUTIL uses the name of the default queue manager specified for OS/390 batch programs in CSQBDEFV. The utility then uses this queue manager for the whole job step. When the utility connects to the queue manager, the authorization of the “signed-on user name” is checked to see which functions the invocation is allowed to use.

You specify the functions required by statements in the SYSIN data set according to these rules:

- The data set must have a record length of 80.
- Only columns 1 through 72 are significant. Columns 73 through 80 are ignored.
- Records with an asterisk (\*) in column 1 are interpreted as comments and are ignored.
- Blank records are ignored.
- Each statement must start on a new line.
- A trailing – means continue from column 1 of the next record.
- A trailing + means continue from the first non-blank column of the next record.
- The keywords of statements are not case-sensitive. However, some arguments, such as queue name, are case-sensitive.

The utility statements refer to the default or explicitly named DDnames for input and output. Your job can use the COPY and LOAD functions repeatedly and process different page sets or queues during a single run of the utility.

All output messages are sent to the SYSPRINT data set, which must have a record format VBA and a record length 125.

While running, CSQUTIL uses temporary dynamic queues with names of the form SYSTEM.CSQUTIL.\*

## Return codes

When CSQUTIL returns to the operating system, the return code can be:

- |    |  |
|----|--|
| 0  | All functions completed successfully.  |
| 4  | Some functions completed successfully, some did not or forced a syncpoint.                                 |
| 8  | All the attempted functions failed.  |
| 12 | No functions attempted; there was a syntax error in the statements or the expected data sets were missing. |

In most cases, if a function fails or is forced to take a syncpoint, no further functions are attempted. In this case, the message CSQU147I replaces the normal completion message CSQU148I.

See the usage notes for each function for more information about success or failure.

### Monitoring the progress of the MQSeries utility program

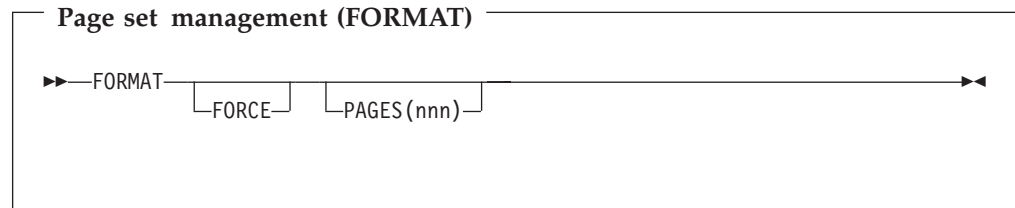
To record the progress of CSQUTIL, every SYSIN statement is echoed to SYSPRINT.

The utility first checks the syntax of the statements in the SYSIN. The requested functions are started only if all the statements are syntactically correct.

Messages giving a commentary on the progress of each function are sent to SYSPRINT. When the processing of the utility is complete, statistics are printed with an indication of how the functions completed.

## Formatting page sets (FORMAT)

Use the FORMAT function to format page sets on all data sets specified by DDnames CSQP0000 through CSQP0099. In this way, you can format up to 100 page sets by invoking the utility program once. Use the FORCE parameter to reuse existing page sets.



## Keywords and parameters

### FORCE

Specifies that existing page sets are to be re-used without having to delete and redefine them first. You must define any page sets you want to re-use with the REUSE attribute in the AMS DEFINE CLUSTER statement. For more information about DEFINE CLUSTER, see the *DFSMS/MVS Access Method Services for VSAM* or the *DFSMS/MVS Access Method Services for the Integrated Catalog Facility* manual.

### PAGES(nnn)

Specifies the minimum number of pages to format in each page set. This enables a data set that spans more than one volume to be formatted.

Formatting of the data set is always done in whole space allocations, as specified as primary or secondary quantities when the data set is defined. The number of space allocations formatted is the minimum necessary to provide the requested number of pages; if there is insufficient data set space available, as many extents as can be obtained are formatted. If an existing page set is being reused (with the FORCE keyword), the whole page set is formatted if that is larger.

The number of pages must be in the range 1 through 1 048 576 (because the maximum page set size is 4 GB (gigabytes)). The default is 1.

The number of pages formatted is reported by message CSQU092I for each page set.

## Example

Figure 57 on page 184 illustrates how the FORMAT command is invoked from CSQUTIL. In this example, two page sets, referenced by CSQP0000 and CSQP0003 respectively, are formatted by CSQUTIL.

## CSQUTIL (FORMAT function)

```
//FORMAT EXEC PGM=CSQUTIL
//STEPLIB DD DISP=SHR,DSN=th1qua1.SCSQANLE
// DD DISP=SHR,DSN=th1qua1.SCSQAUTH
//CSQP0000 DD DISP=OLD,DSN=pageset.dsname0
//CSQP0003 DD DISP=OLD,DSN=pageset.dsname3
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
FORMAT
/*
```

Figure 57. Sample JCL for the *FORMAT* function of *CSQUTIL*

### Usage notes

1. You cannot format page sets that belong to a queue manager that is still running.
2. When you use *FORMAT*, it is not necessary to specify a queue manager name.
3. If you use data set names in which the queue manager name is a high-level qualifier, you can more easily identify which page sets are used by which MQSeries subsystem, if more than one MQSeries subsystem is defined.
4. If there is an error when formatting a page set, it does not prevent other page sets from being formatted, although the *FORMAT* function is considered to have failed.
5. If *FORMAT* fails, no further *CSQUTIL* functions are attempted.



## Expanding a page set (COPYPAGE)

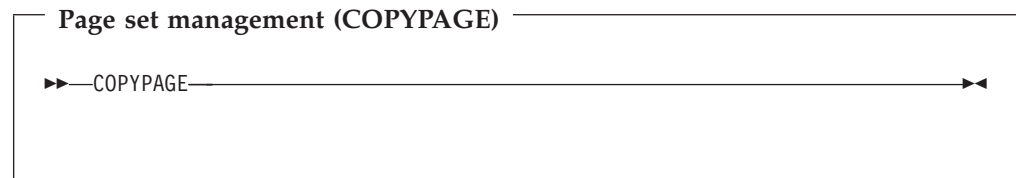
**Note:** The COPYPAGE function is only used for *expanding* page sets. It is not used for making backup copies of page sets. If you want to do this, use AMS REPRO as described in “How to back up and recover page sets” on page 111. When you have used the COPYPAGE function, the page sets cannot be used by a queue manager with a different name, so you must not rename your queue manager.

Use the COPYPAGE function to copy one or more page sets. All queues and messages on the page set are copied. If you copy page set zero, all the MQSeries object definitions are also copied. Each page set is copied to a destination data set that must be formatted as a page set. Copying to a smaller page set is not supported.

If you use this function, you must modify the page set definition in the startup procedure to reflect the change of the name of the data set on which the new page set resides.

To use the COPYPAGE function, define DDnames in the range CSQS0000 through CSQS0099 for the source data sets, and define DDnames for the target data sets from CSQT0000 through CSQT0099 respectively.

For more information, see “Chapter 10. Managing page sets” on page 105.



### Keywords and parameters

There are no keywords or parameters.

### Example

In Figure 58 on page 186, two existing page sets are copied onto two new page sets. The procedure for this is:

1. Set up the required DDnames, where:

**CSQP0005, CSQP0006**

Identify the destination data sets. These DDnames are used by the FORMAT function.

**CSQS0005, CSQS0006**

Identify the source data sets containing the two page sets you want to copy.

**CSQT0005, CSQT0005**

Identify the destination data sets (page sets), but this time for the COPYPAGE function.

2. Format the destination data sets, referenced by DDnames CSQP0005 and CSQP0006, as page sets using the FORMAT function.

## CSQUTIL (COPYPAGE function)

3. Copy the two existing page sets onto the new page sets using the COPYPAGE function.

```
//COPYPAGE EXEC PGM=CSQUTIL
//STEPLIB DD DISP=SHR,DSN=thlqual.SCSQANLE
// DD DISP=SHR,DSN=thlqual.SCSQAUTH
//CSQP0005 DD DISP=OLD,DSN=pageset.newname5
//CSQP0006 DD DISP=OLD,DSN=pageset.newname6
//CSQS0005 DD DISP=OLD,DSN=pageset.oldname5
//CSQS0006 DD DISP=OLD,DSN=pageset.oldname6
//CSQT0005 DD DISP=OLD,DSN=pageset.newname5
//CSQT0006 DD DISP=OLD,DSN=pageset.newname6
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
* Format new data sets (CSQP0005 and CSQP0006) as page sets
FORMAT
* Copy old page sets CSQS0005 and CSQS0006 to new
* page sets CSQT0005 and CSQT0006
COPYPAGE
/*
```

Figure 58. Sample JCL showing the use of the COPYPAGE function

## Usage notes

1. You cannot use COPYPAGE on page sets of a queue manager that is running.
2. Using COPYPAGE involves stopping the queue manager. This will result in the loss of nonpersistent messages.
3. Before you use COPYPAGE, the new data sets must be pre-formatted as page sets. To do this, use the FORMAT function, as shown in Figure 58.
4. Ensure that the new (destination) data sets are larger than the old (source) data sets.
5. You cannot change the page set identifier (PSID) associated with a page set. For example, you cannot 'make' page set 03 become page set 05.
6. Failure of this function does not prevent other page set management functions from being completed.
7. If you attempt to use the COPYPAGE function after MQSeries has terminated abnormally, it is possible that the page sets have not been closed properly. If a page set has not been closed properly, you cannot successfully run the COPYPAGE function against it.

To avoid this problem, run the AMS VERIFY command before using the COPYPAGE function. The AMS VERIFY command might produce error messages. However, it does close the page sets properly, so that the COPYPAGE function can complete successfully.

For more information about the AMS VERIFY command, see the *DFSMS/MVS Access Method Services for VSAM* or the *DFSMS/MVS Access Method Services for the Integrated Catalog Facility* manual.

## Copying a page set and resetting the log (RESETPAGE)

The RESETPAGE function is similar to the COPYPAGE function except that it also resets the log information in the new page sets. RESETPAGE lets you restart MQSeries from a known, valid set of page sets, even if the corresponding log data sets have been corrupted.

The source page sets for RESETPAGE must be in a consistent state. They must be either:

- Page sets that have been through a successful MQSeries shutdown using the MQSeries STOP QMGR command.
- Copies of page sets that have been through a successful stop.

**The RESETPAGE function must not be run against copies of page sets made using fuzzy backup, or against page sets that are from an MQSeries system that has terminated abnormally.**

RESETPAGE either:

- Copies page sets on all data sets referenced by DDnames CSQS0000 through CSQS0099 to new data sets referenced by DDnames CSQT0000 through CSQT0099 respectively. If you use this function, you must modify the page set definition in the startup procedure to reflect the change of the name of the data set on which the new page set resides.
- Resets the log information in the page set referenced by DDnames CSQP0000 through CSQP0099.

For more information, see “Chapter 10. Managing page sets” on page 105.

## Using the RESETPAGE function

You can use the RESETPAGE function to update a set of consistent page sets so that they can be used with a set of new (clean) BSDS and log data sets to start MQSeries. You would only have to do this if both copies of the log have been lost or damaged for some reason; you can restart from backup copies of page sets (and accept the resulting loss of data from the time the copies were made), or from your existing page sets.

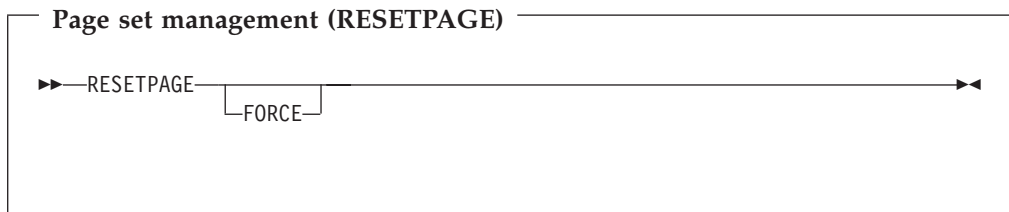
In this situation, you should use the RESETPAGE function on **all** the page sets of the affected queue manager. You must also create new BSDS and log data sets.

**Note:** The RESETPAGE function should not be used on a subset of the page sets known to MQSeries.

If you run the RESETPAGE function against any page sets, but do not provide clean BSDS and log data sets for the MQSeries subsystem, MQSeries will attempt to recover the logs from RBA zero, and will treat the page sets as empty. For example, the following messages would be produced if you attempted to use the RESETPAGE function to generate page sets 0, 1, 2, and 3 without providing a clean set of BSDS and log data sets:

```
CSQI021I +CSQ1 CSQIECUR PAGE SET 0 IS EMPTY. MEDIA RECOVERY STARTED
CSQI021I +CSQ1 CSQIECUR PAGE SET 1 IS EMPTY. MEDIA RECOVERY STARTED
CSQI021I +CSQ1 CSQIECUR PAGE SET 2 IS EMPTY. MEDIA RECOVERY STARTED
CSQI021I +CSQ1 CSQIECUR PAGE SET 3 IS EMPTY. MEDIA RECOVERY STARTED
```

## CSQUTIL (RESETPAGE function)



### Keywords and parameters

#### FORCE

Specifies that the page sets specified by DDnames CSQP0000 through CSQP00nn are to be reset in place.

If FORCE is not specified, the page sets specified by DDnames CSQS0000 through CSQS00nn are copied to new page sets specified by DDnames CSQT0000 through CSQT00nn. This is the default.

### Example

An existing page set, referenced by DDname CSQS0007, is copied to a new data set referenced by DDname CSQT0007. The new data set, which is also referenced by DDname CSQP0007, is already formatted as a page set before the RESETPAGE function is called.

```
//RESTPAGE EXEC PGM=CSQUTIL
//STEPLIB DD DISP=SHR,DSN=thlqual.SCSQANLE
// DD DISP=SHR,DSN=thlqual.SCSQAUTH
//CSQP0007 DD DISP=OLD,DSN=pageset.newname7
//CSQS0007 DD DISP=OLD,DSN=pageset.oldname7
//CSQT0007 DD DISP=OLD,DSN=pageset.newname7
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
* Format new data set, CSQP0007, as page set
FORMAT
* Copy page set CSQS0007 to CSQT0007 and reset it
RESETPAGE
/*
```

Figure 59. Sample JCL showing the use of the RESETPAGE function

### Usage notes

1. You should not attempt to use the RESETPAGE function against page sets after MQSeries has terminated abnormally. It is probable that page sets from an MQSeries system that terminated abnormally contain inconsistent data; using RESETPAGE on page sets in this state leads to data integrity problems.
2. You cannot use RESETPAGE on page sets belonging to a queue manager that is running.
3. Using RESETPAGE involves stopping the queue manager. This will result in the loss of nonpersistent messages.
4. Before you use RESETPAGE, the new data sets must be pre-formatted as page sets. To do this, use the FORMAT function, as shown in Figure 59.
5. Ensure that the new (destination) data sets are larger than the old (source) data sets.

## **CSQUTIL (RESETPAGE function)**

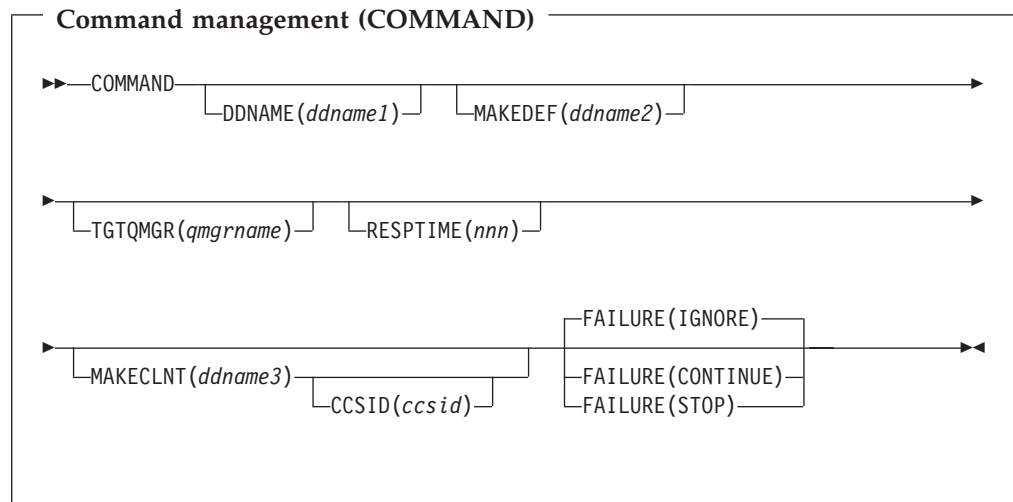
6. You cannot change the page set identifier (PSID) associated with a page set. For example, you cannot 'make' page set 03 become page set 05.
7. Failure of this function does not prevent other page set management functions from being completed.

## Issuing commands to MQSeries (COMMAND)

Use the COMMAND function to:

1. Pass MQSeries commands from an input data set to the queue manager.
2. Produce a list of MQSeries DEFINE commands that describe the objects in an MQSeries subsystem. The statements can be used to keep a record of the object definitions or to regenerate all or part of a queue manager's objects as part of a migration from one MQSeries system to another.
3. Make a client channel definition file.

The queue manager specified in the PARM parameter of the EXEC statement must be running.



### Keywords and parameters

#### DDNAME(*ddname1*)

Specifies that the MQSeries commands are to be read from a named input data set. If this keyword is omitted, the default DDname, CSQUCMD, is used.

*ddname1* specifies the DDname that identifies the input data set from which MQSeries commands are to be read.

#### MAKEDEF(*ddname2*)

Specifies that DEFINE commands are to be generated from any DISPLAY object commands in the input data set.

There is no default if this keyword is omitted.

*ddname2* specifies the DDname that identifies the output data set in which the DEFINE statements are to be stored. The data set should be RECFM=FB, LRECL=80. This data set can then be used as input for a later invocation of the COMMAND function or it can be incorporated into the initialization data sets CSQINP1 and CSQINP2.

#### TGTQMGR(*qmgrname*)

Specifies the name of the queue manager where you want the commands to be performed. You can specify a target queue manager that is not the one you connect to. In this case, you would normally specify the name of a remote

## CSQUTIL (COMMAND function)

queue manager object that provides a queue manager alias definition (the name is used as the *ObjectQMGrName* when opening the command input queue). To do this, you must have suitable queues and channels set up to access the remote queue manager.

The default is that commands are performed on the queue manager to which you are connected, as specified in the PARM field of the EXEC statement.

### RESPTIME(*nnn*)

Specifies the time in seconds to wait for a response to each of the commands, in the range 5 through 999.

The default is 30 seconds.

### MAKECLNT(*ddname3*)

Specifies that a client channel definition file, in binary format suitable for downloading to a client machine, is to be generated from any DISPLAY CHANNEL commands in the input data set that return information about client-connection channels.

If this keyword is omitted, no file is generated.

*ddname3* specifies the DDname that identifies the output data set in which the generated file is to be stored; the data set should be RECFM=U, LRECL=2048, BLKSIZE=2048. The file can then be downloaded as binary data to the client machine by a suitable file transfer program.

### CCSID(*ccsid*)

Specifies the coded character set identifier that is to be used for the data in a client channel definition file. The value must be in the range 1 through 65535; the default is 437. You can only specify CCSID if you also specify MAKECLNT.

**Note:** MQSeries assumes that the data is to be in ASCII, and that the encoding for numeric data is to be MQENC\_INTEGER\_REVERSED.

### FAILURE

Specifies what action to take if an MQSeries command that is issued fails to execute successfully. Values are:

#### IGNORE

Ignore the failure; continue reading and issuing commands, and treat the COMMAND function as being successful. This is the default.

#### CONTINUE

Read and issue any remaining commands in the input data set, but treat the COMMAND function as being unsuccessful.

**STOP** Do not read or issue any more commands, and treat the COMMAND function as being unsuccessful.

## Examples

This section gives examples of using the COMMAND function to do the following:

- “Issuing commands”
- “Making a list of DEFINE commands” on page 192
- “Making a client channel definition file” on page 193

### Issuing commands

In Figure 60 on page 192, the data sets referenced by DDnames CSQUCMD and OTHER contain sets of MQSeries commands. The first COMMAND statement

## CSQUTIL (COMMAND function)

takes MQSeries commands from the default input data set MY.MQSERIES.COMMANDS(COMMAND1) and passes them to the command processor. The second COMMAND statement takes MQSeries commands from the input data set MY.MQSERIES.COMMANDS(OTHER1), which is referenced by DDname OTHER.

```
//COMMAND EXEC PGM=CSQUTIL,PARM='CSQ1'  
//STEPLIB DD DISP=SHR,DSN=thlqua1.SCSQANLE  
// DD DISP=SHR,DSN=thlqua1.SCSQAUTH  
//CSQUCMD DD DSN=MY.MQSERIES.COMMANDS(COMMAND1),DISP=SHR  
//OTHER DD DSN=MY.MQSERIES.COMMANDS(OTHER1),DISP=SHR  
//SYSPRINT DD SYSOUT=*  
//SYSIN DD *  
* NEXT STATEMENT CAUSES COMMANDS TO BE READ FROM CSQUCMD DDNAME  
COMMAND  
* THE NEXT SET OF COMMANDS WILL COME FROM 'OTHER' DDNAME  
COMMAND DDNAME(OTHER)  
* THE NEXT STATEMENT CAUSES COMMANDS TO BE READ FROM CSQUCMD  
* DDNAME AND ISSUED ON QUEUE MANAGER CSQ2 WITH A RESPONSE TIME  
* OF 10 SECONDS  
COMMAND TGTQMGR(CSQ2) RESPTIME(10)  
/*
```

Figure 60. Sample JCL for issuing MQSeries commands using CSQUTIL

## Making a list of DEFINE commands

In Figure 61, the data set referenced by DDname CMDINP contains a set of MQSeries DISPLAY commands. These DISPLAY commands specify generic names for each object type (except the queue manager itself). If you run these commands, a list is produced containing all the MQSeries objects (except the queue manager). In these DISPLAY commands, the ALL keyword is specified to ensure that all the attributes of all the objects are included in the list.

The MAKEDEF keyword causes this list to be converted into a corresponding set of DEFINE commands. These commands are put into a data set referenced by the *ddname2* parameter of the MAKEDEF keyword, that is, OUTPUT1. If you run this set of commands, MQSeries regenerates all the object definitions (except the queue manager) in the MQSeries subsystem.

```
//QDEFS EXEC PGM=CSQUTIL,PARM='CSQ1'  
//STEPLIB DD DISP=SHR,DSN=thlqua1.SCSQANLE  
// DD DISP=SHR,DSN=thlqua1.SCSQAUTH  
//OUTPUT1 DD DISP=OLD,DSN=MY.MQSERIES.COMMANDS(DEFS)  
//SYSPRINT DD SYSOUT=*  
//SYSIN DD *  
COMMAND DDNAME(CMDINP) MAKEDEF(OUTPUT1)  
/*  
//CMDINP DD *  
DISPLAY STGCLASS(*) ALL  
DISPLAY QUEUE(*) ALL  
DISPLAY NAMELIST(*) ALL  
DISPLAY PROCESS(*) ALL  
DISPLAY CHANNEL(*) ALL  
/*
```

Figure 61. Sample JCL for using the MAKEDEF option of the COMMAND function



### Making a client channel definition file

In Figure 62, the data set referenced by DDname CMDCHL contains an MQSeries DISPLAY CHANNEL command. The DISPLAY command specifies a generic name and the ALL keyword is specified to ensure that all the attributes are included.

The MAKECLNT keyword causes this to be converted into a corresponding set of client channel definitions. These are put into a data set referenced by the *ddname3* parameter of the MAKECLNT keyword, that is, OUTCLNT, which is ready to be downloaded to the client machine.

```
//CLIENT EXEC PGM=CSQUTIL,PARM='CSQ1'
//STEPLIB DD DISP=SHR,DSN=thlqua1.SCSQANLE
// DD DISP=SHR,DSN=thlqua1.SCSQAUTH
//OUTCLNT DD DISP=OLD,DSN=MY.MQSERIES.CLIENTS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
COMMAND DDNAME(CMDCHL) MAKECLNT(OUTCLNT)
/*
//CMDCHL DD *
DISPLAY CHANNEL(*) ALL TYPE(CLNTCONN)
/*
```

Figure 62. Sample JCL for using the MAKECLNT option of the COMMAND function

## Usage notes

1. The format of commands issued from the COMMAND function is similar to the MQSeries operator command format. See the *MQSeries MQSC Command Reference* manual for information about the rules for building MQSeries commands.
2. The rules for specifying commands in the input data set are the same as for the initialization data sets:
  - The data set must have a record length of 80.
  - Only columns 1 through 72 are significant. Columns 73 through 80 are ignored.
  - Records with an asterisk (\*) in column 1 are interpreted as comments and are ignored.
  - Blank records are ignored.
  - Each command must start on a new record.
  - A trailing – means continue from column 1 of the next record.
  - A trailing + means continue from the first non-blank column of the next record.
  - The maximum number of characters permitted in a command is 32 762.

With the additional rule:

- A semicolon (;) can be used to terminate a command, the remaining data in the record is ignored.
3. If you specify the MAKEDEF keyword:
    - In the input data set, the DISPLAY commands for objects must contain the ALL parameter so that the complete definition of each object is produced. See Figure 61 on page 192.
    - To obtain a complete definition, you must DISPLAY the following:
      - Queues
      - Namelists

## CSQUTIL (COMMAND function)

Process definitions  
Channels  
Storage classes

**Note:** DEFINE commands are not generated for any local queues that can be identified as dynamic, or for channels that were defined automatically.

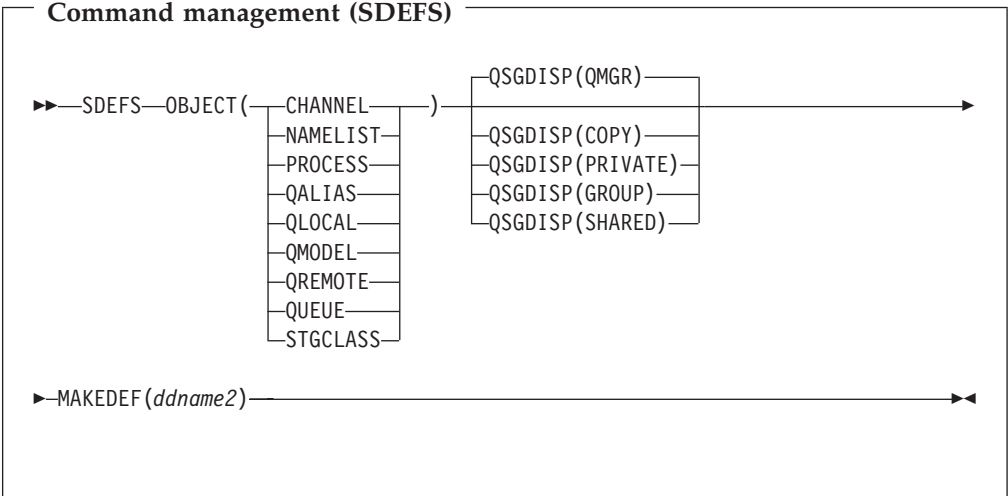
- Do not specify the same MAKEDEF data set for more than one COMMAND function, unless its DD statement specifies a sequential data set with DISP=MOD.
4. Whether or not the MAKEDEF or MAKECLNT keywords are used, the results of these DISPLAY commands are also sent to SYSPRINT.
  5. If you specify the MAKECLNT keyword:
    - In the input data set, the display commands for channels must contain the ALL parameter so that the complete definition of each channel is produced.
    - If the DISPLAY commands return information for a given channel more than once, only the last set of information is used.
    - Do not specify the same client definition file data set for more than one COMMAND function, unless its DD statement specifies a sequential data set with DISP=MOD.
  6. If you specify the FAILURE keyword, a command is considered to execute successfully or not according to the codes returned in message CSQN205I. If the return code is 00000000 and the reason code is 00000000 or 00000004, it is a success; for all other values it is a failure.
  7. The COMMAND function is considered to be successful only if both:
    - All the commands in the input data set are read and issued and get a response from MQSeries, regardless of whether the response indicates successful execution of the command or not.
    - Every command issued executes successfully, if FAILURE(CONTINUE) or FAILURE(STOP) is specified.

If COMMAND fails, no further CSQUTIL functions are attempted.

8. You need the necessary authority to use the command server queues (SYSTEM.COMMAND.INPUT, SYSTEM.COMMAND.REPLY.MODEL, and SYSTEM.CSQUTIL.\*) and to use the MQSC commands that you wish to issue.

## Producing a list of MQSeries define commands (SDEFS)

Use the SDEFS function to produce a list of DEFINE statements describing the objects in your MQSeries subsystem or queue-sharing group.



### Keywords and parameters

**OBJECT**

Specifies the type of object to be listed.

**QSGDISP**

Specifies from where the object definition information is obtained. Depending on how the object has been defined, this information is either:

- On the page set zero referred to by the CSQP0000 DD statement, or
- In a DB2 shared repository.

Permitted values are shown in Table 6.

Table 6. SDEFS QSGDISP parameters and their actions

QSGDISP parameter	What the SDEFS utility does.....
QMGR	Creates DEFINE statements for the specified object type from definitions held on the page set zero referred to by the CSQP0000 DD statement. (1)  Only objects defined with QSGDISP(QMGR) are included.
COPY	Creates DEFINE statements for the specified object type from definitions held on the page set zero referred to by the CSQP0000 DD statement. (1)  Only objects defined with QSGDISP(COPY) are included.
PRIVATE	Creates DEFINE statements for the specified object type from definitions held on the page set zero referred to by the CSQP0000 DD statement. (1)  Both QSGDISP(QMGR) and QSGDISP(COPY) objects are included.

## CSQUTIL (SDEFS function)

Table 6. SDEFS QSGDISP parameters and their actions (continued)

QSGDISP parameter	What the SDEFS utility does.....
GROUP	<p>Creates DEFINE statements for the specified object type from definitions held on DB2 resource definition tables for the specified queue-sharing group.</p> <p>Only objects defined with QSGDISP(GROUP) are included.</p> <p>No CSQP0000 DD statement is required; the DB2 subsystem specified at object definition is accessed. The DB2 library db2qual.SDSNLOAD is required.</p>
SHARED	<p>Creates DEFINE statements for all local queues defined with QSGDISP(SHARED) by accessing the DB2 resource definition table for the specified queue-sharing group.</p> <p><b>This parameter is permitted only with OBJECT(QLOCAL) or OBJECT(Queue).</b></p> <p>No CSQP0000 DD statement is required; the DB2 subsystem specified at object definition is accessed. The DB2 library db2qual.SDSNLOAD is required.</p>
<p><b>Notes:</b></p> <ol style="list-style-type: none"> <li>1. Because only page set zero is accessed, you must ensure that the queue manager is not running.</li> </ol>	

### MAKEDEF(ddname2)

Specifies that define commands generated for the object are to be placed in the output data set identified by the DDname. The data set should be RECFM=FB, LRECL=80. This data set can then be used as input for a later invocation of the COMMAND function or it can be incorporated into the initialization data sets CSQINP1 and CSQINP2.

**Note:** DEFINE commands are not generated for any local queues that can be identified as dynamic, or for channels that were defined automatically.

## Examples

```
//SDEFS EXEC PGM=CSQUTIL
//STEPLIB DD DISP=SHR,DSN=th1qua1.SCSQANLE
// DD DISP=SHR,DSN=th1qua1.SCSQAUTH
//CSQP0000 DD DISP=OLD,DSN=pageset.dsname0
//OUTPUT1 DD DISP=OLD,DSN=MY.MQSERIES.COMMANDS(DEFS)
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
SDEFS OBJECT(Queue) MAKEDEF(OUTPUT1)
/*
```

Figure 63. Sample JCL for the SDEFS function of CSQUTIL

```
//SDEFS EXEC PGM=CSQUTIL
//STEPLIB DD DISP=SHR,DSN=th1qua1.SCSQANLE
// DD DISP=SHR,DSN=th1qua1.SCSQAUTH
// DD DISP=SHR,DSN=db2qua1.SDSNLOAD
//OUTPUT1 DD DISP=OLD,DSN=MY.MQSERIES.COMMANDS(DEFS)
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
SDEFS OBJECT(QLOCAL) QSGDISP(SHARED) MAKEDEF(OUTPUT1)
/*
```

Figure 64. Sample JCL for the SDEFS function of CSQUTIL for objects in the DB2 shared repository

## Usage notes

1. For local queues, you should not use SDEFS for a queue manager that is running because results will be unpredictable. You can avoid doing this accidentally by using DISP=OLD in the CSQP0000 DD statement. For shared or group queue definitions, this does not matter because the information is derived from DB2.
2. When you use SDEFS for local queues it is not necessary to specify a queue manager name. However, for shared and group queue definitions, it is required to access DB2.
3. To use the SDEFS function more than once in a job, specify different DDnames and data sets for each invocation of the function, or specify a sequential data set and DISP=MOD in the DD statements.

## Copying queues into a data set while the queue manager is running (COPY)

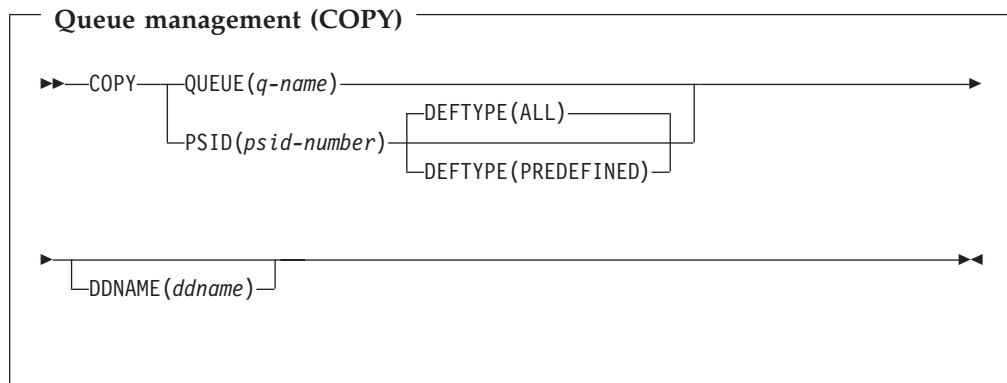
Use the COPY function to copy queued messages to a sequential data set, when the queue manager is running, without destroying any messages in the original queues.

The scope of the COPY function is determined by the keyword that you specify in the first parameter. You can either copy all the messages from a named queue, or all the messages from all the queues on a named page set.

Use the complementary function, LOAD, to restore the messages to their respective queues.

### Notes:

1. If you want to copy the object definitions from the named page set, use COPYPAGE.
2. If you want to copy messages to a data set when the queue manager is not running, use SCOPY.
3. See "Syncpoints" on page 200 for information about how to avoid problems with duplicate messages if this function fails.



## Keywords and parameters

### QUEUE(*q-name*)

QUEUE specifies that messages in the named queue are to be copied. The keyword QUEUE can be abbreviated to Q.

*q-name* specifies the name of the queue to be copied. This name is case-sensitive.

### PSID(*psid-number*)

PSID specifies that all the messages in all the queues in the specified page set are to be copied.

*psid-number* is the page set identifier, which specifies the page set to be used. This identifier is a two-digit integer (whole number) representing a single page set.

### DEFTYPE

Specifies whether to copy dynamic queues:

**ALL** Copy all queues; this is the default.

### PREDEFINED

Do not include dynamic queues; this is the same set of queues that are selected by the COMMAND and SDEFS functions with the MAKEDEF parameter.

### DDNAME

Specifies that the messages are to be copied to a named data set. If this keyword is omitted, the default DDname, CSQUOUT, is used. The keyword DDname can be abbreviated to DD.

*ddname* specifies the DDname of the destination data set, which is used to store the messages. The record format of this data set must be variable block spanned (VBS).

## Example

```
//COPY EXEC PGM=CSQUTIL,PARM='CSQ1'
//STEPLIB DD DISP=SHR,DSN=thlqua1.SCSQANLE
// DD DISP=SHR,DSN=thlqua1.SCSQAUTH
//OUTPUTA DD DSN=SAMPLE.UTILITY.COPYA,DISP=(NEW,CATLG),
// SPACE=(CYL,(5,1),RLSE),UNIT=SYSDA,
// DCB=(RECFM=VBS,BLKSIZE=23200)
//CSQUOUT DD DSN=SAMPLE.UTILITY.COPY3,DISP=(NEW,CATLG),
// SPACE=(CYL,(5,1),RLSE),UNIT=SYSDA,
// DCB=(RECFM=VBS,BLKSIZE=23200)
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
* COPY WHOLE PAGE SET TO 'CSQUOUT'
COPY PSID(03)
* COPY ONE QUEUE TO 'OUTPUT'
COPY QUEUE(ABC123A) DDNAME(OUTPUTA)
/*
```

Figure 65. Sample JCL for the CSQUTIL COPY functions. The sample shows two instances of the COPY function—one COPY to the default DDNAME, CSQUOUT; the other to DDNAME OUTPUTA, which overrides CSQUOUT.

## Usage notes

1. The queues or page set involved must not be in use when the function is invoked.
2. If you want to operate on a range of page sets, you must repeat the COPY function for each page set.
3. The function operates only on local queues.
4. A COPY PSID function is considered successful only if it successfully copies all the queues on the page set.
5. If you try to copy an empty queue (whether explicitly by COPY QUEUE or because there are one or more empty queues on a page set that you are copying), data indicating this is written to the sequential data set, and the copy is considered to be a success. However, if you attempt to copy a non-existent queue, or a page set containing no queues, the COPY function fails, and no data is written to the data set.
6. If COPY fails, no further CSQUTIL functions will be attempted.
7. To use the COPY function more than once in the job, specify different DDnames and data sets for each invocation of the function, or specify a sequential data set and DISP=MOD in the DD statements.

## CSQUTIL (COPY function)

8. You need the necessary authority to use the command server queues (SYSTEM.COMMAND.INPUT, SYSTEM.COMMAND.REPLY.MODEL, and SYSTEM.CSQUTIL.\*), to use the DISPLAY QUEUE and DISPLAY STGCLASS MQSC commands, and to use the MQSeries API to browse messages on the queues that you wish to copy.

### Syncpoints

The queue management functions used when the queue manager is running within a syncpoint so that, if a function fails, its effects can be backed out. The MQSeries entity, MAXSMSGS, specifies the maximum number of messages that a task can get or put within a single unit of recovery.

MAXSMSGS should be greater than:

- The number of messages in the queue – if you are working with a single queue.
- The number of messages in the longest queue in the page set – if you are working with an entire page set.

Otherwise, the utility forcibly takes syncpoints as required and issues the warning message CSQU087I. If the function subsequently fails, the changes already committed will not be backed out. Do not simply re-run the job to correct the problem or you might get duplicate messages on your queues. Instead, use the current depth of the queue to work out, from the utility output, which messages have not been backed out. Then determine the most appropriate course of action. For example, if the function is LOAD you can empty the queue and start again or you can choose to accept duplicate messages on the queues.

Use the DISPLAY QLOCAL command to find out the value of the CURDEPTH attribute, which is the current depth of the queue. To find out the value of MAXSMSGS, use the DISPLAY MAXSMSGS command. See the *MQSeries MQSC Command Reference* manual for more information.



## Copying queues into a data set while the queue manager is not running (SCOPY)

Use the SCOPY function to copy queued messages to a sequential data set when the queue manager is not running, without destroying any messages in the original queues.

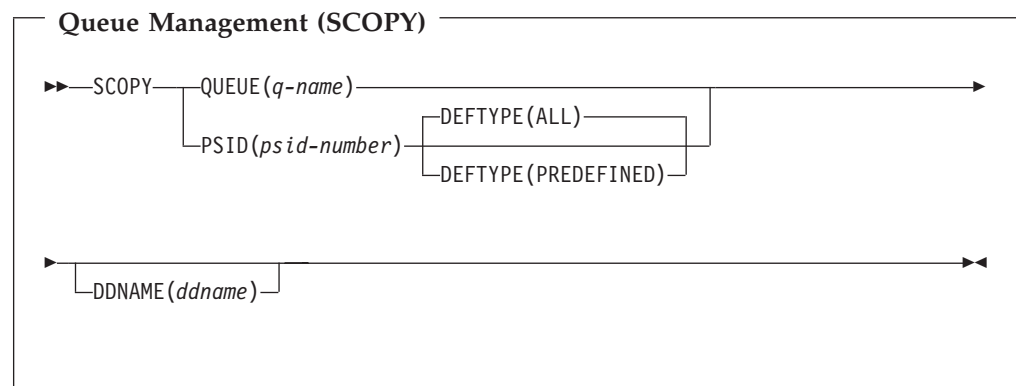
The scope of the SCOPY function is determined by the keyword that you specify in the first parameter. You can either copy all the messages from a named queue, or all the messages from all the queues on a named page set.

Use the complementary function, LOAD, to restore the messages to their respective queues.

To use the SCOPY function, DDname CSQP0000 must specify the data set with page set zero for the subsystem required.

### Notes:

1. The SCOPY function does not operate on shared queues.



## Keywords and parameters

### QUEUE(*q-name*)

QUEUE specifies that messages in the named queue are to be copied. The keyword QUEUE can be abbreviated to Q.

*q-name* specifies the name of the queue to be copied. This name is case-sensitive.

DDname CSQP00*nn* must specify the data set with page set *nn* for the subsystem required, where *nn* is the number of the page set where the queue resides.

### PSID(*psid-number*)

PSID specifies that all the messages in all the queues in the specified page set are to be copied.

*psid-number* is the page set identifier, which specifies the page set to be used. This identifier is a two-digit integer (whole number) representing a single page set.

## CSQUTIL (SCOPY function)

DDname CSQP00 $psid$ -number must specify the data set with the required page set for the subsystem required.

### DEFTYPE

Specifies whether to copy dynamic queues:

**ALL** Copy all queues; this is the default.

### PREDEFINED

Do not include dynamic queues; this is the same set of queues that are selected by the COMMAND and SDEFS functions with the MAKEDEF parameter.

This parameter is only valid if you specify PSID.

### DDNAME

Specifies that the messages are to be copied to a named data set. If this keyword is omitted, the default DDname, CSQUOUT, is used. The keyword DDname can be abbreviated to DD.

*ddname* specifies the DDname of the destination data set, which is used to store the messages. The record format of this data set must be variable block spanned (VBS).

Do not specify the same DDname on more than one SCOPY statement, unless its DD statement specifies a sequential data set with DISP=MOD.

## Example

```
//SCOPY EXEC PGM=CSQUTIL
//STEPLIB DD DISP=SHR,DSN=thlqual.SCSQANLE
// DD DISP=SHR,DSN=thlqual.SCSQAUTH
//OUTPUTA DD DSN=SAMPLE.UTILITY.COPYA,DISP=(NEW,CATLG),
// SPACE=(CYL,(5,1),RLSE),UNIT=SYSDA,
// DCB=(RECFM=VBS,BLKSIZE=23200)
//CSQUOUT DD DSN=SAMPLE.UTILITY.COPY3,DISP=(NEW,CATLG),
// SPACE=(CYL,(5,1),RLSE),UNIT=SYSDA,
// DCB=(RECFM=VBS,BLKSIZE=23200)
//CSQP0000 DD DISP=OLD,DSN=pageset.dsname0
//CSQP0003 DD DISP=OLD,DSN=pageset.dsname3
//CSQP0006 DD DISP=OLD,DSN=pageset.dsname6
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
* COPY WHOLE PAGE SET TO 'CSQUOUT'
SCOPY PSID(03)
* COPY ONE QUEUE TO 'OUTPUT' - QUEUE IS ON PAGE SET 6
SCOPY QUEUE(ABC123A) DDNAME(OUTPUTA)
/*
```

Figure 66. Sample JCL for the CSQUTIL SCOPY functions. The sample shows two instances of the SCOPY function—one SCOPY to the default DDNAME, CSQUOUT; the other to DDNAME OUTPUTA, which overrides CSQUOUT.

## Usage notes

1. You should not use SCOPY for a queue manager that is running because results will be unpredictable. You can avoid doing this accidentally by using DISP=OLD in the page set DD statement.
2. When you use SCOPY, it is not necessary to specify a queue manager name.
3. If you want to operate on a range of page sets, you must repeat the SCOPY function for each page set.

## **CSQUTIL (SCOPY function)**

4. The function operates only on local queues and only for persistent messages.
5. A SCOPY PSID function is considered successful only if it successfully copies all the queues on the page set that have messages; empty queues are ignored. If the page set has no queues with messages, the SCOPY function fails, and no data is written to the data set.
6. If you try to copy an empty queue explicitly by SCOPY QUEUE, data indicating this is written to the sequential data set, and the copy is considered to be a success. However, if you attempt to copy a non-existent queue, the SCOPY function fails, and no data is written to the data set.
7. If the SCOPY function fails, no further CSQUTIL functions are attempted.
8. To use the SCOPY function more than once in the job, specify different DDnames and data sets for each invocation of the function, or specify a sequential data set and DISP=MOD in the DD statements.

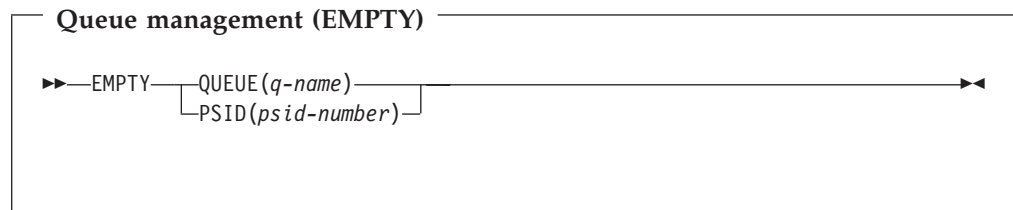
### Emptying a queue of all messages (EMPTY)

Use the EMPTY function to delete all messages from a named queue or all the queues on a page set. The queue manager must be running. The scope of the function is determined by the keyword that you specify in the first parameter.

Use this function with care. You should only delete messages of which copies have already been made.

#### Notes:

1. See “Syncpoints” on page 200 for information about how to avoid problems with duplicate messages if this function fails.



### Keywords and parameters

You must specify the scope of the EMPTY function. Choose one of these:

#### QUEUE(*q-name*)

QUEUE specifies that messages are to be deleted from a named queue. This keyword can be abbreviated to Q.

*q-name* specifies the name of the queue from which messages are to be deleted. This name is case-sensitive.

#### PSID(*psid-number*)

PSID specifies that all the messages are to be deleted from all queues in the named page set.

*psid-number* specifies the page-set identifier. This identifier is a two-digit integer (whole number) representing a single page set.

### Example

```
//EMPTY EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD DISP=SHR,DSN=thlqua1.SCSQANLE
// DD DISP=SHR,DSN=thlqua1.SCSQAUTH
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
EMPTY QUEUE(SPARE)
EMPTY PSID(66)
/*
```

Figure 67. Sample JCL for the CSQUTIL EMPTY function

### Usage notes

1. The queues or page sets involved must not be in use when the function is invoked.
2. This function operates only on local queues.

## CSQUTIL (EMPTY function)

3. If you want to operate on a range of page sets, you must repeat the EMPTY function for each page set.
4. You cannot empty the system-command input queue (SYSTEM.COMMAND.INPUT).
5. An EMPTY PSID function is considered successful only if it successfully empties all the queues on the page set.
6. If you empty a queue that is already empty (whether explicitly by EMPTY QUEUE or because there are one or more empty queues on a page set that you are emptying), the EMPTY function is considered to be a success. However, if you attempt to empty a non-existent queue, or a page set containing no queues, the EMPTY function fails.
7. If EMPTY fails or is forced to take a syncpoint, no further CSQUTIL functions will be attempted.
8. You need the necessary authority to use the command server queues (SYSTEM.COMMAND.INPUT, SYSTEM.COMMAND.REPLY.MODEL, and SYSTEM.CSQUTIL.\*), to use the DISPLAY QUEUE and DISPLAY STGCLASS MQSC commands, and to use the MQSeries API to get messages from the queues that you wish to empty.

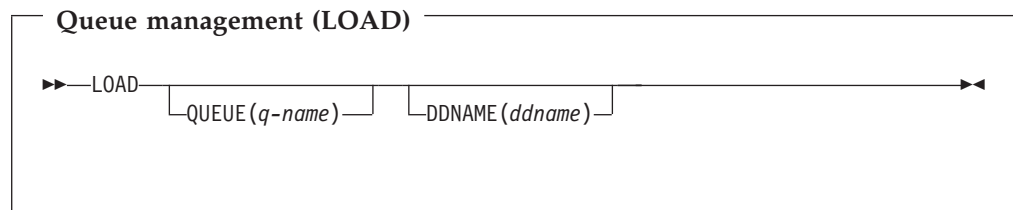
## Restoring messages from a data set to a queue (LOAD)

The LOAD function of CSQUTIL is complementary to the COPY or SCOPY function. LOAD restores messages from the destination data set of an earlier COPY or SCOPY operation. The queue manager must be running.

The data set can contain messages from one queue only if it was created by COPY or SCOPY QUEUE, or from a number of queues if it was created by COPY PSID or several successive COPY or SCOPY QUEUE operations. Messages are restored to queues with the same name as those from which they were copied. You can specify that the first or only queue is loaded to a queue with a different name. (This would normally be used with a data set created with a single COPY queue operation to restore the messages to a queue with a different name.)

### Notes:

1. See “Syncpoints” on page 200 for information about how to avoid problems with duplicate messages if this function fails.



## Keywords and parameters

### QUEUE(*q-name*)

QUEUE specifies that the messages from the first or only queue on the destination data set of a prior COPY or SCOPY operation are to be loaded to a named queue. Messages from any subsequent queues are loaded to queues with the same names as those they came from. The keyword QUEUE can be abbreviated to Q.

*q-name* specifies the name of the queue to which the messages are to be loaded. This name is case-sensitive. It must not be a model queue.

### DDNAME(*ddname*)

DDNAME specifies that messages are to be loaded from a named data set. This keyword can be abbreviated to DD.

*ddname* specifies the DDname that identifies the destination data set of a prior COPY or SCOPY operation—from which the messages are to be loaded. This name is not case-sensitive, and can be up to eight characters long.

If you omit DDname(*ddname*) the default DDname, CSQUINP, is used.

## Example

```
//LOAD EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD DISP=SHR,DSN=thlqua1.SCSQANLE
// DD DISP=SHR,DSN=thlqua1.SCSQAUTH
//OUTPUTA DD DSN=MY.UTILITY.OUTPUTA,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
LOAD QUEUE(ABC123) DDNAME(OUTPUTA)
/*
```

Figure 68. Sample JCL for the CSQUTIL LOAD function

## Usage notes

1. To use the LOAD function, the queues or page sets involved must not be in use when the function is invoked.
2. If the data set contains multiple queues, the LOAD function is considered successful only if it successfully loads all the queues on the data set.
3. If LOAD fails, or is forced to take a syncpoint, no further CSQUTIL functions will be attempted.
4. You need the necessary authority to use the MQSeries API to put messages on the queues that you wish to load.

## CSQUTIL (LOAD function)



---

## Chapter 18. The change log inventory utility (CSQJU003)

The MQSeries change log inventory utility runs as an OS/390 batch job to change the bootstrap data set (BSDS).

Through this utility, you can invoke these functions:

### NEWLOG

Add active or archive log data sets.

### DELETE

Delete active or archive log data sets.

### ARCHIVE

Supply passwords for archive logs.

### CRESTART

Control the next restart of MQSeries.

### CHECKPT

Set checkpoint records.

### HIGHRBA

Update the highest written log RBA.

This utility should be run only when MQSeries is not running. This is because the active log data sets named in the BSDS are dynamically added for exclusive use to MQSeries and remain allocated exclusively to MQSeries until it terminates.

---

### Invoking the CSQJU003 utility

The utility runs as an OS/390 batch program. Figure 69 gives an example of the JCL required.

```
//JU003 EXEC PGM=CSQJU003
//STEPLIB DD DISP=SHR,DSN=thlqua1.SCSQANLE
// DD DISP=SHR,DSN=thlqua1.SCSQAUTH
//SYSPRINT DD SYSOUT=*,DCB=BLKSIZE=629
//SYSUT1 DD DISP=SHR,DSN=bsds.dsname
//SYSIN DD *
NEWLOG DSN=CSQREPAL.A0001187,COPY1VOL=CSQV04,UNIT=SYSDA,
STARTRBA=3A190000,ENDRBA=3A1F0FFF,CATALOG=YES,PASSWORD=PASSWRD
/*
```

Figure 69. Sample JCL to invoke the CSQJU003 utility

### Data definition (DD) statements

CSQJU003 requires DD statements with these DDnames:

#### SYSUT1

This statement is required; it names the BSDS.

#### SYSUT2

This statement is required if you use dual BSDSs; it names the second copy of the BSDS.

#### Dual BSDSs and CSQJU003

Each time you run the CSQJU003 utility, the BSDS time stamp field is updated with the current system time. If you run CSQJU003 separately for each copy of a dual copy BSDS, the time stamp fields are not synchronized

## Change log inventory utility

so that MQSeries fails at startup, issuing error message CSQJ120E. Therefore, if CSQJU003 is used to update dual copy BSDSs, both BSDSs must be updated within a single run of CSQJU003.

### **SYSPRINT**

This statement is required; it names a data set for print output. The logical record length (LRECL) is 125. The block size (BLKSIZE) must be 629.

### **SYSIN**

This statement is required; it names the input data set for statements that specify what the utility is to do. The logical record length (LRECL) is 80.

You can use more than one statement of each type. In each statement, separate the operation name (NEWLOG, DELETE, ARCHIVE, CRESTART) from the first parameter by one or more blanks. You can use parameters in any order; separate them by commas with no blanks. Do not split a parameter description across two SYSIN records.

A statement containing an asterisk in column 1 is considered to be a comment, and is ignored. However, it appears in the output listing. To include a comment or sequence number in a SYSIN record, separate it from the last comma by a blank. When a blank follows a comma, the rest of the record is ignored.

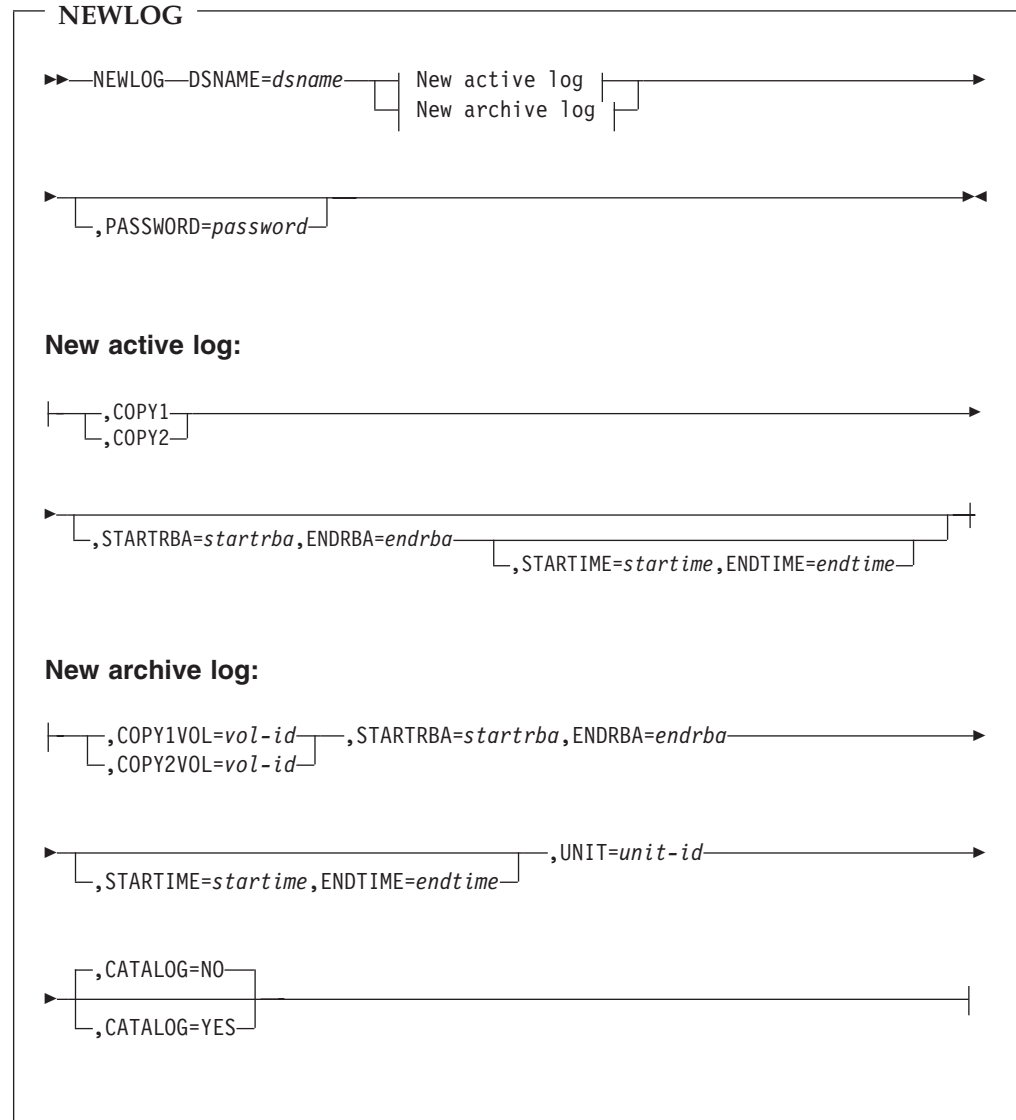
## **Multiple statement operation**

When running CSQJU003, a significant error in any statement causes the control statements for the statement in error and all following statements to be skipped. Therefore, BSDS updates cannot occur for any operation specified in the statement in error, or any following statements. However, all the remaining statements are checked for syntax errors.

## Adding information about a data set to the BSDS (NEWLOG)

The NEWLOG function declares one of these data sets:

- A VSAM data set that is available for use as an active log data set.  
Use the keywords DSNAME, COPY1, COPY2, and PASSWORD.
- An active log data set that is replacing one that encountered an I/O error.  
Use the keywords DSNAME, COPY1, COPY2, STARTRBA, ENDRBA, and PASSWORD.
- An archive log data set volume.  
Use the keywords DSNAME, COPY1VOL, COPY2VOL, STARTRBA, ENDRBA, UNIT, CATALOG, and PASSWORD.



## Keywords and parameters

**DSNAME**=*dsname*

Names a log data set. *dsname* can be up to 44 characters long.

**PASSWORD**=*password*

Assigns a password to the data set. It is stored in the BSDS and subsequently used in any access to the active or archive log data sets.

The password is a data set password, and should follow standard VSAM convention: 1 through 8 alphanumeric characters (A through Z, 0 through 9) or special characters (& \* + - . ; ' /).

We recommend that you use an ESM such as RACF to provide your data set security requirements.

**COPY1**

Makes the data set an active log copy-1 data set.

**COPY2**

Makes the data set an active log copy-2 data set.

**STARTRBA**=*startrba*

Gives the log RBA (relative byte address within the log) of the beginning of the replacement active log data set or the archive log data set volume specified by DSNAME. *startrba* is a hexadecimal number of up to 12 characters. The value must end with 000. If you use fewer than 12 characters, leading zeros are added. The RBA can be obtained from messages or by printing the log map.

**ENDRBA**=*endrba*

Gives the log RBA (relative byte address within the log) of the end of the replacement active log data set or the archive log data set volume specified by DSNAME. *endrba* is a hexadecimal number of up to 12 characters. The value must end with FFF. If you use fewer than 12 characters, leading zeros are added.

**STARTIME**=*starttime*

Lets you record the start time of the RBA in the BSDS. This is an optional field. The time stamp format (with valid values in parentheses) is yyyydddhhmmsst, where:

<b>yyyy</b>	Indicates the year (1993 through 2099)
<b>ddd</b>	Indicates the day of the year (0 through 365; 366 in leap years)
<b>hh</b>	Indicates the hour (0 through 23)
<b>mm</b>	Indicates the minutes (0 through 59)
<b>ss</b>	Indicates the seconds (0 through 59)
<b>t</b>	Indicates tenths of a second

If fewer than 14 digits are specified for the STARTIME and ENDTIME parameter, then trailing zeros will be added.

STARTRBA is required when STARTIME is specified.

**ENDTIME**=*endtime*

Enables you to record the end time of the RBA in the BSDS. This is an optional field. For time stamp format, see the STARTIME option. The ENDTIME value must be greater than or equal to the value of STARTIME.

**COPY1VOL**=*vol-id*

The volume serial of the copy-1 archive log data set named after DSNAME.

**COPY2VOL**=*vol-id*

The volume serial of the copy-2 archive log data set named after DSNAME.

**UNIT**=*unit-id*

The device type of the archive log data set named after DSNAME.

**CATALOG**

Tells whether the archive log data set is cataloged:

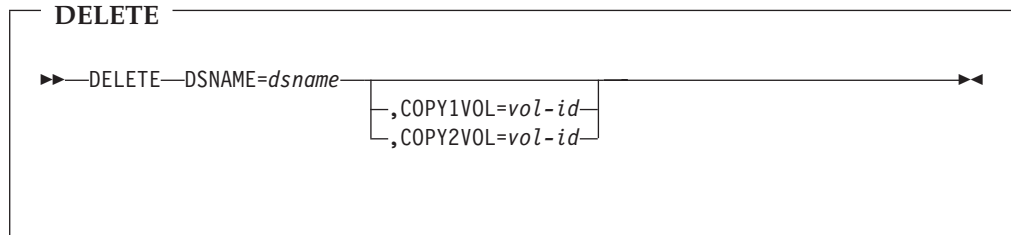
**NO** Indicates that the archive log data set is not cataloged. All subsequent allocations of the data set are made using the unit and volume information specified on the function. The default is NO.

**YES** Indicates that the archive log data set is cataloged. A flag is set in the BSDS indicating this, and all subsequent allocations of the data set are made using the catalog.

MQSeries requires that all archive log data sets on DASD be cataloged. Select CATALOG=YES if the archive log data set is on DASD.

## Deleting information about a data set from the BSDS (DELETE)

Use the DELETE function to delete all information about a specified log data set or data set volume from the bootstrap data sets. For example, you can use this function to delete outdated archive log data sets.



### Keywords and parameters

**DSNAME=*dsname***

Specifies the name of the log data set. *dsname* can be up to 44 characters long.

**COPY1VOL=*vol-id***

The volume serial number of the copy-1 archive log data set named after DSNAME.

**COPY2VOL=*vol-id***

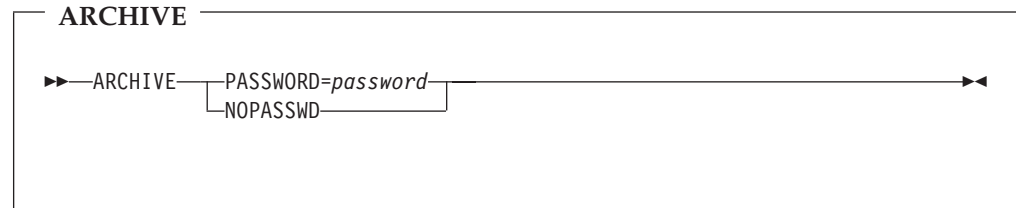
The volume serial number of the copy-2 archive log data set named after DSNAME.

## Supplying a password for archive log data sets (ARCHIVE)

Use the ARCHIVE function to give a password to all archive data sets created after this operation. This password is added to the installation's OS/390 password data set each time a new archive log data set is created.

Use the NOPASSWD keyword to remove the password protection for all archives created after the archive operation.

**Note:** You should normally use an ESM, such as RACF, if you want to implement security on any MQSeries data sets.



## Keywords and parameters

### **PASSWORD=***password*

PASSWORD specifies that a password is to be assigned to the archive log data sets.

*password* specifies the password, which is a data set password and it must follow the standard VSAM convention; that is, 1 through 8 alphanumeric characters (A through Z, 0 through 9) or special characters (& \* + - . ; ' /).

### **NOPASSWD**

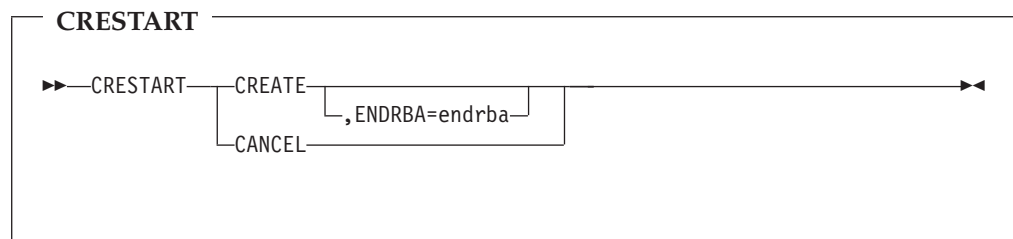
Specifies that archive password protection is not to be active for all archives created after this operation. No other keyword can be used with NOPASSWD.

## Controlling the next restart (CRESTART)

Use the CRESTART function to control the next restart of MQSeries, either by creating a new conditional restart control record or by cancelling the one currently active. These records limit the scope of the log data that will be used during restart. Any existing conditional restart control record governs every restart until one of these events occurs:

- A restart operation completes
- A CRESTART CANCEL is issued
- A new conditional restart control record is created

**Attention:** This can override MQSeries efforts to maintain data in a consistent state. You would normally only use this function when implementing the disaster recovery process described in “Alternative site recovery” on page 127, or under the guidance of IBM service.



### Keywords and parameters

#### CREATE

Creates a new conditional restart control record. When the new record is created, the previous control record becomes inactive.

#### CANCEL

Makes the currently active conditional restart control record inactive. The record remains in the BSDS as historical information.

No other keyword can be used with CANCEL.

#### ENDRBA=*endrba*

Gives the last RBA of the log to be used during restart, and the starting RBA of the next active log to be written after restart. Any log information in the bootstrap data set and the active logs, with an RBA greater than *endrba*, is discarded. If you omit this option, MQSeries determines the end of the log range.

*endrba* is a hexadecimal number of up to 12 digits. If you use fewer than 12 digits, leading zeros are added.

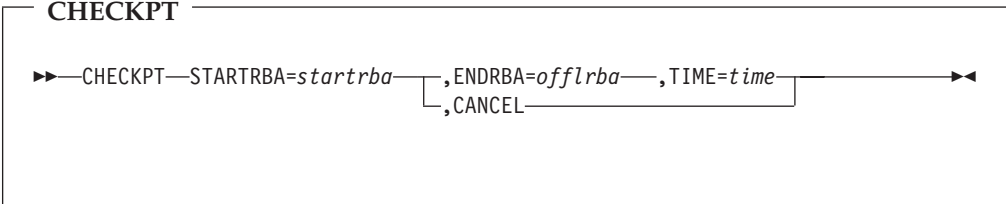
The value of ENDRBA must be a multiple of 4096. (The hexadecimal value must end in 000.)



## Setting checkpoint records (CHECKPT)

Use the CHECKPT function to add or delete a record in the BSDS checkpoint queue. Use the STARTRBA and ENDRBA keywords to add a record, or the STARTRBA and CANCEL keywords to delete a record.

**Attention:** This can override MQSeries efforts to maintain data in a consistent state. You would normally only use this function when implementing the disaster recovery process described in “Alternative site recovery” on page 127, or under the guidance of IBM service.



### Keywords and parameters

**STARTRBA=startrba**

Indicates the start checkpoint log record.

*startrba* is a hexadecimal number of up to 12 digits. If you use fewer than 12 digits, leading zeros are added. The RBA can be obtained from messages or by printing the log map.

**ENDRBA=endrba**

Indicates the end checkpoint log record corresponding to the start checkpoint record.

*endrba* is a hexadecimal number of up to 12 digits. If you use fewer than 12 digits, leading zeros are added. The RBA can be obtained from messages or by printing the log map.

**TIME=time**

Gives the time the start checkpoint record was written. The time stamp format (with valid values in parentheses) is `yyydddhhmmsst`, where:

- yyyy** Indicates the year (1993 through 2099)
- ddd** Indicates the day of the year (0 through 365; 366 in leap years)
- hh** Indicates the hour (0 through 23)
- mm** Indicates the minutes (0 through 59)
- ss** Indicates the seconds (0 through 59)
- t** Indicates tenths of a second

If fewer than 14 digits are specified for the TIME parameter, then trailing zeros are added.

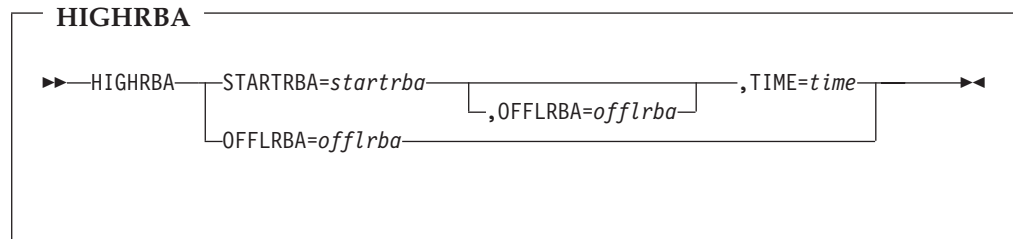
**CANCEL**

Deletes the checkpoint queue record containing a starting RBA that matches the RBA specified by STARTRBA.

## Updating the highest written log RBA (HIGHRBA)

Use the HIGHRBA function to update the highest written log RBA recorded in the BSDS for either the active or archive log data sets. Use the STARTRBA keyword to update the active log, and the OFFLRBA keyword to update the archive log.

**Attention:** This can override MQSeries efforts to maintain data in a consistent state. You would normally only use this function when implementing the disaster recovery process described in "Alternative site recovery" on page 127, or under the guidance of IBM service.



### Keywords and parameters

#### STARTRBA=*starttrba*

Indicates the log RBA of the highest written log record in the active log data set.

*starttrba* is a hexadecimal number of up to 12 digits. If you use fewer than 12 digits, leading zeros are added. The RBA can be obtained from messages or by printing the log map.

#### TIME=*time*

Specifies when the log record with the highest RBA was written to the log. The time stamp format (with valid values in parentheses) is yyyydddhhmmsst, where:

- yyyy Indicates the year (1993 through 2099)
- ddd Indicates the day of the year (0 through 365; 366 in leap years)
- hh Indicates the hour (0 through 23)
- mm Indicates the minutes (0 through 59)
- ss Indicates the seconds (0 through 59)
- t Indicates tenths of a second

If fewer than 14 digits are specified for the TIME parameter, then trailing zeros will be added.

#### OFFLRBA=*offlrba*

Specifies the highest off-loaded RBA in the archive log.

*offlrba* is a hexadecimal number of up to 12 digits. If you use fewer than 12 digits, leading zeros are added. The value must end with hexadecimal 'FFF'.

---

## Chapter 19. The print log map utility (CSQJU004)

The MQSeries print log map utility runs as an OS/390 batch program to list this information:

- Log data set name and log RBA association for both copies of all active and archive log data sets
- Active log data sets available for new log data
- Contents of the queue of checkpoint records in the bootstrap data set (BSDS)
- Contents of the quiesce history record
- System and utility time stamps
- Passwords for the active and archive log data sets, if provided

The CSQJU004 program can be run regardless of whether MQSeries is running. However, if MQSeries is running, consistent results from the utility can be ensured only if both the utility and the MQSeries subsystem are running under control of the same OS/390 system.

To use this utility, the user ID of the job must have the requisite security authorization, or, if the BSDS is password protected, the appropriate VSAM password for the data set.

---

### Invoking the CSQJU004 utility

Figure 70 shows an example of the JCL used to invoke the CSQJU004 utility:

```
//JU004 EXEC PGM=CSQJU004
//STEPLIB DD DISP=SHR,DSN=thlqua1.SCSQANLE
// DD DISP=SHR,DSN=thlqua1.SCSQAUTH
//SYSPRINT DD SYSOUT=*,DCB=BLKSIZE=629
//SYSUT1 DD DISP=SHR,DSN=bsds.dsname
```

Figure 70. Sample JCL to invoke the CSQJU004 utility

### Data definition statements

The CSQJU004 utility requires DD statements with the following DDnames:

#### **SYSUT1**

This statement is required to specify and allocate the bootstrap data set. If the BSDS must be shared with a concurrently executing MQSeries online subsystem, use DISP=SHR on the DD statement.

#### **SYSPRINT**

This statement is required to specify a data set or print spool class for print output. The logical record length (LRECL) is 125. The block size (BLKSIZE) must be 629.

“Finding out what the BSDS contains” on page 97 describes the output.



---

## Chapter 20. The log print utility (CSQ1LOGP)

You can use this utility to print information contained in the logs or the BSDS.

---

### Invoking the CSQ1LOGP utility

You run the MQSeries log print utility as an OS/390 batch program. You can specify:

- A bootstrap data set (BSDS)
- Active logs (with no BSDS)
- Archive logs (with no BSDS)

Sample JCL to invoke the CSQ1LOGP utility is shown in figures 71, 72, and 73.

These DD statements should be provided:

#### **SYSPRINT**

All error messages, exception conditions and the detail report are written to this data set. The logical record length (LRECL) is 131.

#### **SYSIN**

Input selection criteria can be specified in this data set (see “Input control parameters” on page 222 for more information).

#### **SYSSUMRY**

If a summary report is requested, the output is written to this data set. The logical record length (LRECL) is 131.

**BSDS** Name of the bootstrap data set (BSDS).

#### **ACTIVEn**

Name of an active log data set you want to print (n=number).

#### **ARCHIVE**

Name of an archive log data set you want to print.

**Note:** The utility will not run if MQSeries is active and you are trying to process active logs (using a BSDS or the active logs directly).

```
//PRTLOG EXEC PGM=CSQ1LOGP
//STEPLIB DD DISP=SHR,DSN=thlqual.SCSQANLE
// DD DISP=SHR,DSN=thlqual.SCSQLOAD
//BSDS DD DSN=bsds.dsname,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSSUMRY DD SYSOUT=*
//SYSIN DD *
insert your input control statements here
/*
```

Figure 71. Sample JCL to invoke the CSQ1LOGP utility using a BSDS

## Log print utility

```
//PRTLOG EXEC PGM=CSQ1LOGP
//STEPLIB DD DISP=SHR,DSN=thlqua1.SCSQANLE
//          DD DISP=SHR,DSN=thlqua1.SCSQLOAD
//ACTIVE1 DD DSN=bsds.logcopy1.ds01,DISP=SHR
//ACTIVE2 DD DSN=bsds.logcopy1.ds02,DISP=SHR
//ACTIVE3 DD DSN=bsds.logcopy1.ds03,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSSUMRY DD SYSOUT=*
//SYSIN DD *
        insert your input control statements here
/*
```

Figure 72. Sample JCL to invoke the CSQ1LOGP utility using active log data sets

```
//PRTLOG EXEC PGM=CSQ1LOGP
//STEPLIB DD DISP=SHR,DSN=thlqua1.SCSQANLE
//          DD DISP=SHR,DSN=thlqua1.SCSQLOAD
//ARCHIVE DD DSN=bsds.archive1.ds01,DISP=SHR
//          DD DSN=bsds.archive1.ds02,DISP=SHR
//          DD DSN=bsds.archive1.ds03,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSSUMRY DD SYSOUT=*
//SYSIN DD *
        insert your input control statements here
/*
```

Figure 73. Sample JCL to invoke the CSQ1LOGP utility using archive logs

---

## Input control parameters

The keywords that you can use in the SYSIN data set are described below:

### **RBASTART(hexadecimal-constant)**

Specifies the log RBA from which to begin processing. If you are using a BSDS, this parameter must be specified.

Normally you are only interested in the most recent additions to the log. Therefore, do not specify a value of zero. If you do, you create an enormous amount of data, most of which is of no interest to you.

You can also use the forms STARTRBA or ST. Specify this keyword only once.

### **RBAEND(hexadecimal-constant)**

Specifies the last valid log RBA that is to be processed. If this keyword is omitted, processing continues to the end of the log (FFFFFFFFFFFF).

You can also use the forms ENDRBA or EN. Specify this keyword only once.

### **PAGESET(decimal-integer)**

Specifies a page set identifier. The number should be in the range 00 through 99. Only log records associated with the page set you specify will be processed.

**URID (hexadecimal-constant)**

Specifies a hexadecimal unit of recovery identifier. Changes to data occur in the context of an MQSeries unit of recovery. A unit of recovery is identified on the log via a BEGIN UR record. The log RBA of that BEGIN UR record is the URID value you must use. If you know the URID for a given UR that you are interested in, you can limit the extraction of information from the MQSeries log to that URID.

The hexadecimal constant can consist of 1 through 12 characters (6 bytes), and leading zeros are not required.

You can specify a maximum of 10 URID keywords in any given CSQ1LOGP job. To narrow the search, you can specify URID keywords in a job that contains other keywords.

**RM (resource\_manager)**

Specifies a particular resource manager. Only records associated with this resource manager will be processed. Valid values for this keyword are:

**RECOVERY**

Recovery log manager

**DATA** Data manager

**BUFFER**

Buffer manager

**XCF** IMS bridge

**SUMMARY(YES|NO|ONLY)**

Specifies whether a summary report is to be produced or not:

**YES** Produce a summary report in addition to the detail report.

**NO** Do not produce a summary report.

**ONLY** Produce only a summary report (no detail report).

---

## Output

The detail report begins by echoing the input selection criteria specified via SYSIN, and then prints each valid log record encountered. Definitions of keywords in the detail report are as follows:

**RM** Resource manager that wrote the log record.

**TYPE** Type of log record.

**URID** BEGIN UR for this unit of recovery, see the description above.

**LRID** Logical record identifier in the form:

AAAAAAAA.BBBBBBCC

where:

**AAAAAAAA**

Is the page set number.

**BBBBBB**

Is the relative page number in the page set.

**CC**

Is the relative record number on the page.

**SUBTYPE**

Subtype of the log record type.

**CHANGE LENGTH**

Length of the logged change.

## **Log print utility**

### **CHANGE OFFSET**

Start position of the change.

### **BACKWARD CHAIN**

Pointer to the previous page.

### **FORWARD CHAIN**

Pointer to the next page.

### **RECORD LENGTH**

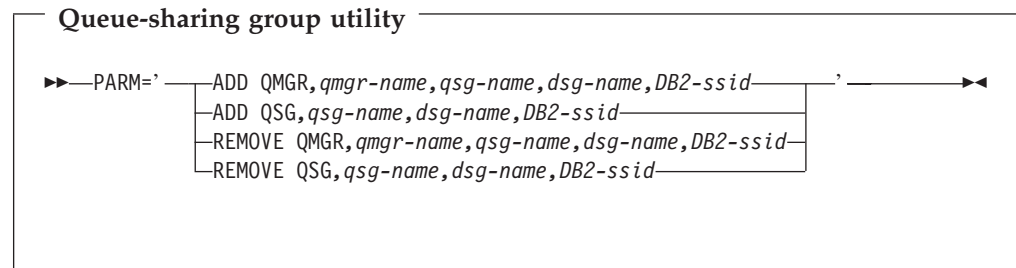
Length of the inserted record.



---

## Chapter 21. The queue-sharing group utility (CSQ5PQSG)

Use the CSQ5PQSG utility program to add queue-sharing group and queue manager definitions to the MQSeries DB2 tables, and to remove them.



---

### Invoking the queue-sharing group utility

Figure 74 shows an example of the JCL used to invoke the CSQ5PQSG utility.

```
//S001 EXEC PGM=CSQ5PQSG,REGION=4M,
//      PARM='function,function parameters'
//STEPLIB DD DSN=th1qua1.SCSQANLE,DISP=SHR
//        DD DSN=th1qua1.SCSQAUTH,DISP=SHR
//        DD DSN=db2qua1.SDSNLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
```

Figure 74. Sample JCL to invoke the CSQ5PQSG utility

### Data definition statements

The CSQ5PQSG utility requires data definition statements with the following DDnames:

#### SYSPRINT

This statement is required; it names the data set for print output. The logical record length (LRECL) is 125.

---

### Keywords and parameters

#### PARM

This field contains the function request *function* followed by the function-specific parameters *function parameters*. These are described below:

#### ADD QMGR

Add a queue manager record into the CSQ.ADMIN\_B\_QMGR table. This will only complete successfully if a corresponding queue-sharing group record already exists in the CSQ.ADMIN\_B\_QSG table and the queue manager entry does not already exist in the CSQ.ADMIN\_B\_QMGR table as the member of a different queue-sharing group.

*qmgr-name*            The queue manager name

*qsg-name*            The queue-sharing group name

## CSQ5PQSG utility

<i>dsg-name</i>	The DB2 data-sharing group name
<i>DB2-ssid</i>	The DB2 subsystem ID

### ADD QSG

Add a queue-sharing group record into the CSQ.ADMIN\_B\_QSG table.

<i>qsg-name</i>	The queue-sharing group name
<i>dsg-name</i>	The DB2 data-sharing group name
<i>DB2-ssid</i>	The DB2 subsystem ID

### REMOVE QMGR

Remove a queue manager record from the CSQ.ADMIN\_B\_QMGR table. This will only complete successfully if the queue manager has either never been started, or terminated normally from its last execution.

<i>qmgr-name</i>	The queue manager name
<i>qsg-name</i>	The queue-sharing group name
<i>dsg-name</i>	The DB2 data-sharing group name
<i>DB2-ssid</i>	The DB2 subsystem ID

### REMOVE QSG

Remove a queue-sharing group record from the CSQ.ADMIN\_B\_QSG table. This will only complete successfully if no queue managers are defined to the queue-sharing group.

<i>qsg-name</i>	The queue-sharing group name
<i>dsg-name</i>	The DB2 data-sharing group name
<i>DB2-ssid</i>	The DB2 subsystem ID

---

## Example

The following sample JCL adds an entry for queue manager QM01 into queue-sharing group QSG1. It specifies a connection to DB2 subsystem DB2A, which is a member of DB2 data-sharing group DSN510PG.

```
//S001 EXEC PGM=CSQ5PQSG,REGION=4M,  
// PARM='ADD QMGR,QM01,QSG1,DSN510PG,DB2A'  
//STEPLIB DD DSN=th1qua1.SCSQANLE,DISP=SHR  
// DD DSN=th1qua1.SCSQAUTH,DISP=SHR  
// DD DSN=db2qua1.SDSNLOAD,DISP=SHR  
//SYSPRINT DD SYSOUT=*
```

Figure 75. Using the queue-sharing group utility to add a queue manager into a queue-sharing group

---

## Chapter 22. The dead-letter queue handler utility (CSQUDLQH)

A *dead-letter queue* (DLQ) is a holding queue for messages that cannot be delivered to their destination queues. Every queue manager in a network should have an associated DLQ.

Queue managers, message channel agents, and applications can put messages on the DLQ. All messages on the DLQ should be prefixed with a *dead-letter header* structure, MQDLH. Messages put on the DLQ by a queue manager or by a message channel agent always have a dead-letter header; you are strongly recommended to ensure that applications putting messages on the DLQ supply a dead-letter header as well. The *Reason* field of the MQDLH structure contains a reason code that identifies why the message is on the DLQ.

There should be a routine that runs regularly to process messages on the DLQ. MQSeries supplies a default *dead-letter queue handler* (DLQ handler) called CSQUDLQH. A user-written *rules table* supplies instructions to the DLQ handler, for processing messages on the DLQ. That is, the DLQ handler matches messages on the DLQ against entries in the rules table. When a DLQ message matches an entry in the rules table, the DLQ handler performs the action associated with that entry.

---

### Invoking the DLQ handler

The CSQUDLQH utility program runs as an OS/390 batch program. You need to specify the name of the dead-letter queue that you want to process and the queue manager on which it resides. You can do this in one of the following two ways (in these examples, the dead-letter queue is called CSQ1.DEAD.QUEUE and the queue manager is called CSQ1):

1. The names can be specified as positional parameters in the PARM parameter of the EXEC statement within the submitted JCL, for example:

```
//READQ EXEC PGM=CSQUDLQH,  
// PARM='CSQ1.DEAD.QUEUE CSQ1'
```

Figure 76. Specifying the queue manager and dead-letter queue names for the dead-letter queue handler in the JCL

2. The names can be specified in the rules table, for example:

```
INPUTQ(CSQ1.DEAD.QUEUE) INPUTQM(CSQ1)
```

Figure 77. Specifying the queue manager and dead-letter queue names for the dead-letter queue handler in the rules table

Any parameters that you specify in the PARM parameter override those in the rules table. If you specify only one parameter in the PARM statement, this is used as the name of the dead-letter queue. The rules table is taken from the SYSIN data set.

## CSQUDLQH utility

### Data definition statements

CSQUDLQH requires DD statements with these DDnames:

#### SYSOUT

This statement is required; it names the data set for print output. You can specify the logical record length (LRECL) and block size (BLKSIZE) for this output data set.

#### SYSIN

This statement is required; it names the input data set containing the rules table that specifies what the utility is to do. The logical record length (LRECL) is 80.

### Sample JCL

```
//READQ EXEC PGM=CSQUDLQH,  
//      PARM='CSQ1.DEAD.QUEUE CSQ1'  
//STEPLIB DD DSN=th1qua1.SCSQAUTH,DISP=SHR  
//      DD DSN=th1qua1.SCSQLOAD,DISP=SHR  
//      DD DSN=th1qua1.SCSQANLE,DISP=SHR  
//SYSOUT DD SYSOUT=*  
//SYSIN   DD *  
INPUTQM(CSQ2) INPUTQ('CSQ2.DEAD.QUEUE')  
ACTION(RETRY)  
/*
```

Figure 78. Sample JCL to invoke the CSQUDLQH utility. In this example, queue manager CSQ1 and dead-letter queue CSQ1.DEAD.QUEUE are used because the values specified in the PARM statement override the values specified in the SYSIN data set.

---

## The DLQ handler rules table

The DLQ handler rules table defines how the DLQ handler is to process messages that arrive on the DLQ. There are two types of entry in a rules table:

- The first entry in the table, which is optional, contains *control data*.
- All other entries in the table are *rules* for the DLQ handler to follow. Each rule consists of a *pattern* (a set of message characteristics) that a message is matched against, and an *action* to be taken when a message on the DLQ matches the specified pattern. There must be at least one rule in a rules table.

Each entry in the rules table comprises one or more keywords.

See “Rules table conventions” on page 233 for information about the syntax of the rules table.

## Control data

This section describes the keywords that you can include in a control-data entry in a DLQ handler rules table.

- All keywords are optional.
- If a control-data entry is included in the rules table, it *must* be the first entry in the table.
- The default value for a keyword, if any, is underlined.
- The vertical line (|) separates alternatives. You can specify only one of these.

### INPUTQ (QueueName|' \_')

Specifies the name of the DLQ that you want to process:

1. If you specify a queue name in the PARM parameter of the EXEC statement, this overrides any INPUTQ value in the rules table.
2. If you do not specify a queue name in the PARM parameter of the EXEC statement, the INPUTQ value in the rules table is used.
3. If you do not specify a queue name in the PARM parameter of the EXEC statement or the rules table, the dead-letter queue named *qmgr-name*.DEAD.QUEUE is used if it has been defined. If this queue does not exist, the program fails and returns error message CSQU224E, giving the reason code for the error.

### INPUTQM (QueueManagerName|' \_')

Specifies the name of the queue manager that owns the DLQ named on the INPUTQ keyword.

1. If you specify a queue manager name in the PARM parameter of the EXEC statement, this overrides any INPUTQM value in the rules table.
2. If you do not specify a queue manager name in the PARM parameter of the EXEC statement, the INPUTQM value in the rules table is used.
3. If you do not specify a queue manager name in the PARM parameter of the EXEC statement or the rules table, the default queue manager is used (if one has been defined using CSQBDEFV). If not, the program fails and returns error message CSQU220E, giving the reason code for the error.

### RETRYINT (Interval|60)

Specifies the interval, in seconds, at which the DLQ handler should attempt to reprocess messages on the DLQ that could not be processed at the first attempt, and for which repeated attempts have been requested.

The default is 60 seconds.

### WAIT (YES|NO|nnn)

Specifies whether the DLQ handler should wait for further messages to arrive on the DLQ when it detects that there are no further messages that it can process.

**YES** Causes the DLQ handler to wait indefinitely.

**NO** Causes the DLQ handler to terminate when it detects that the DLQ is either empty or contains no messages that it can process.

*nnn* Causes the DLQ handler to wait for *nnn* seconds for new work to arrive after it detects that the queue is either empty or contains no messages that it can process, before terminating.

Specify a value in the range 1 through 999 999.

## CSQUDLQH utility

You are recommended to specify WAIT (YES) for busy DLQs, and WAIT (NO) or WAIT (*nnn*) for DLQs that have a low level of activity. If the DLQ handler is allowed to terminate, you can use triggering to invoke it when needed.

## Rules (patterns and actions)

Figure 79 shows an example rule from a DLQ handler rules table.

```
PERSIST(MQPER_PERSISTENT) REASON (MQRC_PUT_INHIBITED) +  
ACTION (RETRY) RETRY (3)
```

Figure 79. An example rule from a DLQ handler rules table. This rule instructs the DLQ handler to make three attempts to deliver to its destination queue any persistent message that was put on the DLQ because MQPUT and MQPUT1 were inhibited.

This section describes the keywords that you can include in a rules table. It begins with a description of the pattern-matching keywords (those against which messages on the DLQ are matched). It then describes the action keywords (those that determine how the DLQ handler is to process a matching message).

- All keywords except ACTION are optional.
- The default value for a keyword, if any, is underlined. For most keywords, the default value is \* (asterisk), which matches any value.
- The vertical line (|) separates alternatives. You can specify only one of these.

### The pattern-matching keywords

The pattern-matching keywords, are described below. You use these to specify values against which messages on the DLQ are matched. All pattern-matching keywords are optional.

#### **APPLIDAT** (*ApplIdentityData* | \*)

The *ApplIdentityData* value of the message on the DLQ, specified in the message descriptor, MQMD.

#### **APPLNAME** (*PutApplName* | \*)

The name of the application that issued the MQPUT or MQPUT1 call, as specified in the *PutApplName* field of the message descriptor, MQMD, of the message on the DLQ.

#### **APPLTYPE** (*PutApplType* | \*)

The *PutApplType* value specified in the message descriptor, MQMD, of the message on the DLQ.

#### **DESTQ** (*QueueName* | \*)

The name of the message queue for which the message is destined.

#### **DESTQM** (*QueueManagerName* | \*)

The queue manager name for the message queue for which the message is destined.

#### **FEEDBACK** (*Feedback* | \*)

Describes the nature of the report when the *MsgType* value is MQMT\_REPORT.

You can use symbolic names. For example, you can use the symbolic name MQFB\_COA to identify those messages on the DLQ that require confirmation of their arrival on their destination queues. A few symbolic names are not accepted by the utility and lead to a syntax error. In these cases, you can use the corresponding numeric value.

**FORMAT** (*Format* | \*)

The name that the sender of the message uses to describe the format of the message data.

**MSGTYPE** (*MsgType* | \*)

The message type of the message on the DLQ.

You can use symbolic names. For example, you can use the symbolic name MQMT\_REQUEST to identify those messages on the DLQ that require replies.

**PERSIST** (*Persistence* | \*)

The persistence value of the message. (The persistence of a message determines whether it survives restarts of the queue manager.)

You can use symbolic names. For example, you can use the symbolic name MQPER\_PERSISTENT to identify those messages on the DLQ that are persistent.

**REASON** (*ReasonCode* | \*)

The reason code that describes why the message was put to the DLQ.

You can use symbolic names. For example, you can use the symbolic name MQRC\_Q\_FULL to identify those messages placed on the DLQ because their destination queues were full. A few symbolic names are not accepted by the utility and lead to a syntax error. In these cases, you can use the corresponding numeric value.

**REPLYQ** (*QueueName* | \*)

The reply-to queue name specified in the message descriptor, MQMD, of the message on the DLQ.

**REPLYQM** (*QueueManagerName* | \*)

The queue manager name of the reply-to queue specified in the REPLYQ keyword.

**USERID** (*UserIdentifier* | \*)

The user ID of the user who originated the message on the DLQ, as specified in the message descriptor, MQMD.

**The action keywords**

The action keywords are described below. You use these to describe how a matching message is processed.

**ACTION** (**DISCARD** | **IGNORE** | **RETRY** | **FWD**)

The action taken for any message on the DLQ that matches the pattern defined in this rule.

**DISCARD**

Causes the message to be deleted from the DLQ.

**IGNORE**

Causes the message to be left on the DLQ.

**RETRY**

Causes the DLQ handler to try again to put the message on its destination queue.

**FWD**

Causes the DLQ handler to forward the message to the queue named on the FWDQ keyword.

## CSQUDLQH utility

You must specify the ACTION keyword. The number of attempts made to implement an action is governed by the RETRY keyword. The RETRYINT keyword of the control data controls the interval between attempts.

### FWDQ (*QueueName* | &DESTQ | &REPLYQ)

The name of the message queue to which the message is forwarded when you select the ACTION keyword.

#### *QueueName*

This parameter is the name of a message queue. FWDQ(' ') is not valid.

#### &DESTQ

Takes the queue name from the *DestQName* field in the MQDLH structure.

#### &REPLYQ

Takes the name from the *ReplyToQ* field in the message descriptor, MQMD. You can specify REPLYQ (?\*) in the message pattern to avoid error messages, when a rule specifying FWDQ (&REPLYQ), matches a message with a blank *ReplyToQ* field.

### FWDQM (*QueueManagerName* | &DESTQM | &REPLYQM | ' ')

The queue manager of the queue to which a message is forwarded.

#### *QueueManagerName*

This parameter defines the queue manager name for the queue to which the message is forwarded when you select the ACTION (FWD) keyword.

#### &DESTQM

Takes the queue manager name from the *DestQMGrName* field in the MQDLH structure.

#### &REPLYQM

Takes the name from the *ReplyToQMGr* field in the message descriptor, MQMD.

' ' The local queue manager.

### HEADER (YES | NO)

Whether the MQDLH should remain on a message for which ACTION (FWD) is requested. By default, the MQDLH remains on the message. The HEADER keyword is not valid for actions other than FWD.

### PUTAUT (DEF | CTX)

The authority with which messages should be put by the DLQ handler:

**DEF** Puts messages with the authority of the DLQ handler itself.

**CTX** Causes the messages to be put with the authority of the user ID in the message context. You must be authorized to assume the identity of other users, if you specify PUTAUT (CTX).

### RETRY (*RetryCount* | 1)

The number of times that an action should be attempted (at the interval specified on the RETRYINT keyword of the control data). Specify a value in the range 1 through 999 999 999.

**Note:** The count of attempts made by the DLQ handler to implement any particular rule is specific to the current instance of the DLQ handler; the count does not persist across restarts. If you restart the DLQ handler, the count of attempts made to apply a rule is reset to zero.



## Rules table conventions

The rules table must adhere to the following conventions regarding its syntax, structure, and contents:

- A rules table must contain at least one rule.
- Keywords can occur in any order.
- A keyword can be included once only in any rule.
- Keywords are not case sensitive.
- A keyword and its parameter value can be separated from other keywords by at least one blank or comma.
- Any number of blanks can occur at the beginning or end of a rule, and between keywords, punctuation, and values.
- Each rule must begin on a new line.
- For reasons of portability, the significant length of a line should not be greater than 72 characters.
- Use the plus sign (+) as the last nonblank character on a line to indicate that the rule continues from the first nonblank character in the next line. Use the minus sign (-) as the last nonblank character on a line to indicate that the rule continues from the start of the next line. Continuation characters can occur within keywords and parameters.

For example:

```
APPLNAME('ABC+
D')
```

results in 'ABCD'.

```
APPLNAME('ABC-
D')
```

results in 'ABC D'.

- Comment lines, which begin with an asterisk (\*), can occur anywhere in the rules table.
- Blank lines are ignored.

Each entry in the DLQ handler rules table comprises one or more keywords and their associated parameters. The parameters must follow these syntax rules:

- Each parameter value must include at least one significant character. The delimiting quotation marks in quoted values are not considered significant. For example, these parameters are valid:
 

FORMAT('ABC')	3 significant characters
FORMAT(ABC)	3 significant characters
FORMAT('A')	1 significant character
FORMAT(A)	1 significant character
FORMAT(' ')	1 significant character

These parameters are not valid because they contain no significant characters:

- FORMAT('')
- FORMAT( )
- FORMAT()
- FORMAT

## CSQUDLQH utility

- Wildcard characters are supported. You can use the question mark (?) in place of any single character, except a trailing blank. You can use the asterisk (\*) in place of zero or more adjacent characters. The asterisk (\*) and the question mark (?) are *always* interpreted as wildcard characters in parameter values.
- You cannot include wildcard characters in the parameters of these keywords: ACTION, HEADER, RETRY, FWDQ, FWDQM, and PUTAUT.
- Trailing blanks in parameter values, and in the corresponding fields in the message on the DLQ, are not significant when performing wildcard matches. However, leading and embedded blanks within strings in quotation marks are significant to wildcard matches.
- Numeric parameters cannot include the question mark (?) wildcard character. You can include the asterisk (\*) in place of an entire numeric parameter, but the asterisk cannot be included as part of a numeric parameter. For example, these are valid numeric parameters:  
MSGTYPE(2)      Only reply messages are eligible  
MSGTYPE(\*)      Any message type is eligible  
MSGTYPE(' \* ')      Any message type is eligible

However, MSGTYPE('2\*') is not valid, because it includes an asterisk (\*) as part of a numeric parameter.

- Numeric parameters must be in the range 0 through 999 999 999 unless otherwise stated. If the parameter value is in this range, it is accepted, even if it is not currently valid in the field to which the keyword relates. You can use symbolic names for numeric parameters.
- If a string value is shorter than the field in the MQDLH or MQMD to which the keyword relates, the value is padded with blanks to the length of the field. If the value, excluding asterisks, is longer than the field, an error is diagnosed. For example, these are all valid string values for an 8-character field:  
'ABCDEFGH'                      8 characters  
'A\*C\*E\*G\*I'                      5 characters excluding asterisks  
'\*A\*C\*E\*G\*I\*K\*M\*O\*'              8 characters excluding asterisks
- Strings that contain blanks, lowercase characters, or special characters other than period (.), forward slash (/), underscore (\_), and percent sign (%) must be enclosed in single quotation marks. Lowercase characters not enclosed in quotation marks are folded to uppercase. If the string includes a quotation, two single quotation marks must be used to denote both the beginning and the end of the quotation. When the length of the string is calculated, each occurrence of double quotation marks is counted as a single character.

## Processing the rules table

The DLQ handler searches the rules table for a rule whose pattern matches a message on the DLQ. The search begins with the first rule in the table, and continues sequentially through the table. When a rule with a matching pattern is found, the rules table attempts the action from that rule. The DLQ handler increments the retry count for a rule by 1 whenever it attempts to apply that rule. If the first attempt fails, the attempt is repeated until the count of attempts made matches the number specified on the `RETRY` keyword. If all attempts fail, the DLQ handler searches for the next matching rule in the table.

This process is repeated for subsequent matching rules until an action is successful. When each matching rule has been attempted the number of times specified on its `RETRY` keyword, and all attempts have failed, `ACTION (IGNORE)` is assumed. `ACTION (IGNORE)` is also assumed if no matching rule is found.

### Notes:

1. Matching rule patterns are sought only for messages on the DLQ that begin with an MQDLH. If the dead-letter queue handler encounters one or more messages that are not prefixed by an MQDLH, it issues an information message to report this. Messages that do not contain an MQDLH are not processed by the DLQ handler and remain on the dead-letter queue until dealt with by another method.
2. All pattern keywords can default, so that a rule may consist of an action only. Note, however, that action-only rules are applied to all messages on the queue that have MQDLHs and that have not already been processed in accordance with other rules in the table.
3. The rules table is validated when the DLQ handler starts, and errors flagged at that time. You can make changes to the rules table at any time, but those changes do not come into effect until the DLQ handler is restarted.
4. The DLQ handler does not alter the content of messages, of the MQDLH, or of the message descriptor. The DLQ handler always puts messages to other queues with the message option `MQPMO_PASS_ALL_CONTEXT`.
5. Consecutive syntax errors in the rules table may not be recognized because the validation of the rules table is designed to eliminate the generation of repetitive errors.
6. The DLQ handler opens the DLQ with the `MQOO_INPUT_AS_Q_DEF` option.
7. You should not run applications that perform `MQGET` calls against the queue at the same time as the DLQ handler. This includes multiple instances of the DLQ handler. It is more usual for there to be a one-to-one relationship between the dead-letter queue and the DLQ handler.

## Ensuring that all DLQ messages are processed

The DLQ handler keeps a record of all messages on the DLQ that have been seen but not removed. If you use the DLQ handler as a filter to extract a small subset of the messages from the DLQ, the DLQ handler still keeps a record of those messages on the DLQ that it did not process. Also, the DLQ handler cannot guarantee that new messages arriving on the DLQ will be seen, even if the DLQ is defined as first-in first-out (FIFO). Therefore, if the queue is not empty, the DLQ is periodically rescanned to check all messages. For these reasons, you should try to ensure that the DLQ contains as few messages as possible. If messages that cannot be discarded or forwarded to other queues (for whatever reason) are allowed to accumulate on the queue, the workload of the DLQ handler increases and the DLQ itself is in danger of filling up.

You can take specific measures to enable the DLQ handler to empty the DLQ. For example, try not to use ACTION (IGNORE), which simply leaves messages on the DLQ. (Remember that ACTION (IGNORE) is assumed for messages that are not explicitly addressed by other rules in the table.) Instead, for those messages that you would otherwise ignore, use an action that moves the messages to another queue. For example:

```
ACTION (FWD) FWDQ (IGNORED.DEAD.QUEUE) HEADER (YES)
```

Similarly, the final rule in the table should be a catchall to process messages that have not been addressed by earlier rules in the table. For example, the final rule in the table could be something like this:

```
ACTION (FWD) FWDQ (REALLY.DEAD.QUEUE) HEADER (YES)
```

This causes messages that fall through to the final rule in the table to be forwarded to the queue REALLY.DEAD.QUEUE, where they can be processed manually. If you do not have such a rule, messages are likely to remain on the DLQ indefinitely.

## An example DLQ handler rules table

Here is an example rules table that contains a single control-data entry and several rules:

```

*****
*           An example rules table for the CSQUDLQH utility           *
*****
* Control data entry
* -----
* If no queue manager name is supplied as an explicit parameter to CSQUDLQH,
* use the default queue manager.
* If no queue name is supplied as an explicit parameter to CSQUDLQH, use the
* DLQ defined for the queue manager.
*
inputqm(' ') inputq(' ')

* Rules
* ----

* The first check deals with attempted security violations.
* If a message was placed on the DLQ because the putter did not have the
* appropriate authority for the target queue, forward the message to a queue
* for manual inspection.

REASON(MQRC_NOT_AUTHORIZED) ACTION(FWD) +
FWDQ(DEADQ.MANUAL.SECURITY)

* The next set of rules with ACTION (RETRY) try to deliver the message to the
* intended destination.

* If a message is placed on the DLQ because its destination queue is full,
* attempt to forward the message to its destination queue. Make 5 attempts at
* approximately 60-second intervals (the default value for RETRYINT).

REASON(MQRC_Q_FULL) ACTION(RETRY) RETRY(5)

* If a message is placed on the DLQ because of a put inhibited condition, attempt
* to forward the message to its destination queue. Make 5 attempts at
* approximately 60-second intervals (the default value for RETRYINT).

REASON(MQRC_PUT_INHIBITED) ACTION(RETRY) RETRY(5)

* The AAAA corporation often send messages with incorrect addresses. When we find
* a request from the AAAA corporation, we return it to the DLQ (DEADQ) of the
* reply-to queue manager (&REPLYQM). The AAAA DLQ handler attempts to
* redirect the message.

MSGTYPE(MQMT_REQUEST) REPLYQM(AAAA.*) +
ACTION(FWD) FWDQ(DEADQ) FWDQM(&REPLYQM)

* The BBBB corporation requests that we try sending messages to queue manager
* BBB2 if queue manager BBB1 is unavailable.

DESTQM(BBB1) +
ACTION(FWD) FWDQ(&DESTQ) FWDQM(BBB2) HEADER(NO)

* The CCCC corporation is very security conscious, and believes that none of its
* messages will ever end up on one of our DLQs. If we do see a message from a
* CCCC queue manager on our DLQ, we send it to a special destination in the CCCC
* organization where the problem is investigated.

REPLYQM(CCCC.*) +
ACTION(FWD) FWDQ(ALARM) FWDQM(CCCC.SYSTEM)

* Messages that are not persistent risk being lost when a queue manager terminates.
* If an application is sending nonpersistent messages, it will be able to cope with

```

## CSQUDLQH utility

| \* the message being lost, so we can afford to discard the message.

| PERSIST(MQPER\_NOT\_PERSISTENT) ACTION(DISCARD)

| \* For performance and efficiency reasons, we like to keep the number of messages on  
| \* the DLQ small. If we receive a message that has not been processed by an earlier  
| \* rule in the table, we assume that it requires manual intervention to resolve the  
| \* problem.

| \* Some problems are best solved at the node where the problem was detected, and  
| \* others are best solved where the message originated. We do not have the message  
| \* origin, but we can use the REPLYQM to identify a node that has some interest  
| \* in this message. Attempt to put the message onto a manual intervention queue  
| \* at the appropriate node. If this fails, put the message on the manual  
| \* intervention queue at this node.

| REPLYQM('?\*') +  
|     ACTION(FWD) FWDQ(DEADQ.MANUAL.INTERVENTION) FWDQM(&REPLYQM)

| ACTION(FWD) FWDQ(DEADQ.MANUAL.INTERVENTION)

---

## Part 7. Appendixes





## Appendix A. User messages on start-up

When you start MQSeries successfully, it produces a set of start up messages similar to the ones in Figure 80.

### Notes:

1. If you are starting MQSeries for the first time, the messages are slightly different.
2. If any of the values in message CSQR004I is not zero, message CSQR007I is issued to provide the restart status table.
3. Messages CSQP018I and CSQP019I are issued every time a checkpoint is taken. At checkpoint time, all pages that have not been changed for the two checkpoints are written out to DASD. Message CSQP019I is issued for each buffer pool, giving the number of pages written. You can use this information when balancing page sets in buffer pools.

If you want to suppress these messages, see the *MQSeries for OS/390 System Setup Guide*.

4. There might be periods during startup when no messages are produced; for example, if you are using indexed queues, no messages are produced while the queue indexes are being rebuilt.

```
$HASP373 CSQ1MSTR STARTED
IEF403I CSQ1MSTR - STARTED - TIME=13.35.20
CSQY000I +CSQ1 IBM MQSeries for OS/390 - V5.2
CSQY001I +CSQ1 SUBSYSTEM STARTING, USING PARAMETER MODULE CSQZPARM
CSQY100I +CSQ1 System parameters ...
CSQY101I +CSQ1 CTHREAD=300, IDBACK=20, IDFORE=100, LOGLOAD=10000
CSQY102I +CSQ1 CMDUSER=CSQOPR, QMCCSID=500, ROUTCDE=( 1)
CSQY103I +CSQ1 SMFACCT=NO (00000000), SMFSTAT=NO (00000000), STATIME=30
CSQY104I +CSQ1 OTMACON=( , ,DFSYDRU0,2147483647,CSQ)
CSQY105I +CSQ1 TRACSTR=( 1), TRACTBL=500, RESAUDIT=NO
CSQY106I +CSQ1 EXITTCB=8, EXITLIM=30, WLMTIME=30
CSQY107I +CSQ1 QSGDATA=(SQ05,DSN510MQ,DDMQ,0)
CSQY110I +CSQ1 Logging parameters ...
CSQY111I +CSQ1 INBUFF=28, OUTBUFF=400, MAXRTU=2, MAXARCH=500
CSQY112I +CSQ1 TWOACTV=YES, TWOARCH=YES, TWOBSDS=YES
CSQY113I +CSQ1 OFFLOAD=YES, WRTHRS=20, DEALLCT=(0,0)
CSQY120I +CSQ1 Archive parameters ...
CSQY121I +CSQ1 UNIT=TAPE, UNIT2=, ALCUNIT=BLK, PRIQTY=4320, SECQTY=540, BLKSIZE=20480
CSQY122I +CSQ1 ARCPFX1=CSQARC1, ARCPFX2=CSQARC2, TSTAMP=NO
CSQY123I +CSQ1 ARCRETN=9999, ARCWTOR=YES, ARCWRTC=( 1 ,3 ,4)
CSQY124I +CSQ1 CATALOG=NO, COMPACT=NO, PROTECT=NO, QUIESCE=5
CSQY201I +CSQ1 CSQYSTRT ARM REGISTER for element SYSMQMGRCSQ1 type SYSMQMGR successful
CSQJ127I +CSQ1 SYSTEM TIME STAMP FOR BSDS=2000-01-19 13:33:19.82
CSQJ001I +CSQ1 CSQJW007 CURRENT COPY 1 ACTIVE LOG
DATA SET IS DSNAME=VICY.CSQ1.LOGCOPY1.DS01,
STARTRBA=000000000000,ENDRBA=00000021BFFF
CSQJ001I +CSQ1 CSQJW007 CURRENT COPY 2 ACTIVE LOG
DATA SET IS DSNAME=VICY.CSQ1.LOGCOPY2.DS01,
STARTRBA=000000000000,ENDRBA=00000021BFFF
CSQJ099I +CSQ1 LOG RECORDING TO COMMENCE WITH STARTRBA=000000150000
CSQ5001I +CSQ1 CSQ5CONN Connected to DB2 M141
```

Figure 80. MQSeries startup messages for subsystem CSQ1 (Part 1 of 3)

## Startup messages

```
CSQH024I +CSQ1 CSQHINSQ SUBSYSTEM security switch set ON,  
profile 'SQ05.NO.SUBSYS.SECURITY' not found  
CSQH022I +CSQ1 CSQHINSQ QMGR security switch set ON,  
profile 'CSQ1.YES.QMGR.CHECKS' found  
CSQH021I +CSQ1 CSQHINSQ QSG security switch set OFF,  
profile 'SQ05.NO.QSG.CHECKS' found  
CSQH021I +CSQ1 CSQHIS1C CONNECTION security switch set OFF,  
profile 'CSQ1.NO.CONNECT.CHECKS' found  
CSQH024I +CSQ1 CSQHIS1C COMMAND security switch set ON,  
profile 'CSQ1.NO.CMD.CHECKS' not found  
CSQH021I +CSQ1 CSQHIS1C CONTEXT security switch set OFF,  
profile 'CSQ1.NO.CONTEXT.CHECKS' found  
CSQH024I +CSQ1 CSQHIS1C ALTERNATE USER security switch set ON,  
profile 'CSQ1.NO.ALTERNATE.USER.CHECKS' not found  
CSQH021I +CSQ1 CSQHIS1C COMMAND RESOURCES security switch set OFF,  
profile 'CSQ1.NO.CMD.RESC.CHECKS' found  
CSQH024I +CSQ1 CSQHIS1C PROCESS security switch set ON,  
profile 'CSQ1.NO.PROCESS.CHECKS' not found  
CSQH024I +CSQ1 CSQHIS1C NAMELIST security switch set ON,  
profile 'CSQ1.NO.NLIST.CHECKS' not found  
CSQH024I +CSQ1 CSQHIS1C QUEUE security switch set ON,  
profile 'CSQ1.NO.QUEUE.CHECKS' not found  
CSQV452I +CSQ1 CSQVXLR Cluster workload exits not available  
CSQR001I +CSQ1 RESTART INITIATED  
CSQR003I +CSQ1 RESTART - PRIOR CHECKPOINT RBA=00000014EA08  
CSQR004I +CSQ1 RESTART - UR STATUS COUNTS  
IN COMMIT=0, INDOUBT=0, INFLIGHT=0, IN BACKOUT=0  
CSQI049I +CSQ1 Page set 0 has media recovery  
RBA=00000014EA08, checkpoint RBA=00000014EA08  
CSQI049I +CSQ1 Page set 1 has media recovery  
RBA=00000014EA08, checkpoint RBA=00000014EA08  
CSQI049I +CSQ1 Page set 2 has media recovery  
RBA=00000014EA08, checkpoint RBA=00000014EA08  
CSQI049I +CSQ1 Page set 3 has media recovery  
RBA=00000014EA08, checkpoint RBA=00000014EA08  
CSQI049I +CSQ1 Page set 4 has media recovery  
RBA=00000014EA08, checkpoint RBA=00000014EA08  
CSQI049I +CSQ1 Page set 5 has media recovery  
RBA=00000014EA08, checkpoint RBA=00000014EA08  
IXL014I IXLCONN REQUEST FOR STRUCTURE SQ05CSQ_ADMIN  
WAS SUCCESSFUL. JOBNAME: CSQ1MSTR ASID: 0157  
CONNECTOR NAME: CSQESQ05CSQ101 CFNAME: SICF01  
CSQE005I +CSQ1 Structure SQ05CSQ_ADMIN connected as  
CSQESQ05CSQ101, version=B35661B9D2CB1F04 0001008A  
CSQR030I +CSQ1 Forward recovery log range  
from RBA=00000014EA08 to RBA=00000014F768  
CSQR005I +CSQ1 RESTART - COUNTS AFTER FORWARD RECOVERY  
IN COMMIT=0, INDOUBT=0  
CSQR032I +CSQ1 Backward recovery log range  
from RBA=00000014F768 to RBA=00000014F768  
CSQR006I +CSQ1 RESTART - COUNTS AFTER BACKWARD RECOVERY  
INFLIGHT=0, IN BACKOUT=0  
CSQR002I +CSQ1 RESTART COMPLETED  
CSQP018I +CSQ1 CSQPBACK CHECKPOINT STARTED FOR ALL BUFFER POOLS  
+CSQ1 DISPLAY THREAD(*) TYPE(INDOUBT)  
CSQP019I +CSQ1 CSQP1DWP CHECKPOINT COMPLETED FOR BUFFER POOL 2, 2 PAGES WRITTEN  
CSQP019I +CSQ1 CSQP1DWP CHECKPOINT COMPLETED FOR BUFFER POOL 3, 2 PAGES WRITTEN  
CSQP019I +CSQ1 CSQP1DWP CHECKPOINT COMPLETED FOR BUFFER POOL 1, 5 PAGES WRITTEN  
CSQP019I +CSQ1 CSQP1DWP CHECKPOINT COMPLETED FOR BUFFER POOL 0, 18 PAGES WRITTEN
```

Figure 80. MQSeries startup messages for subsystem CSQ1 (Part 2 of 3)

```
CSQP021I +CSQ1 Page set 0 new media recovery
RBA=000000151506, checkpoint RBA=000000151506
CSQP021I +CSQ1 Page set 1 new media recovery
RBA=000000151506, checkpoint RBA=000000151506
CSQP021I +CSQ1 Page set 2 new media recovery
RBA=000000151506, checkpoint RBA=000000151506
CSQP021I +CSQ1 Page set 3 new media recovery
RBA=000000151506, checkpoint RBA=000000151506
CSQP021I +CSQ1 Page set 4 new media recovery
RBA=000000151506, checkpoint RBA=000000151506
CSQP021I +CSQ1 Page set 5 new media recovery
RBA=000000151506, checkpoint RBA=000000151506
CSQV401I +CSQ1 DISPLAY THREAD REPORT FOLLOWS -
CSQV420I +CSQ1 NO INDOUBT THREADS FOUND
CSQ9022I +CSQ1 CSQVDT ' DISPLAY THREAD' NORMAL COMPLETION
CSQY022I +CSQ1 QUEUE MANAGER INITIALIZATION COMPLETE
CSQ9022I +CSQ1 CSQYASCP 'START QMGR' NORMAL COMPLETION
```

*Figure 80. MQSeries startup messages for subsystem CSQ1 (Part 3 of 3)*

|



---

## Appendix B. Notices

This information was developed for products and services offered in the United States. IBM may not offer the products, services, or features discussed in this information 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 information. The furnishing of this information 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.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation  
Licensing  
2-31 Roppongi 3-chome, Minato-ku  
Tokyo 106, Japan

**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 information. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this information 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.

## Notices

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM United Kingdom Laboratories,  
Mail Point 151,  
Hursley Park,  
Winchester,  
Hampshire,  
England  
SO21 2JN.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

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.

---

## Programming interface information

This book is intended to help you to administer and operate MQSeries for OS/390.

This book also documents General-use Programming Interface and Associated Guidance Information and Product-sensitive Programming Interface and Associated Guidance Information provided by MQSeries for OS/390.

General-use programming interfaces allow the customer to write programs that obtain the services of MQSeries for OS/390.

General-use Programming Interface and Associated Guidance Information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

### **Start of General-use programming interface**

General-use Programming Interface and Associated Guidance Information...

### **End of General-use programming interface**

Product-sensitive programming interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning of MQSeries for OS/390. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM software product. Product-sensitive programming interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

Product-sensitive Programming Interface and Associated Guidance Information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

### **Start of Product-sensitive programming interface**

Product-sensitive Programming Interface and Associated Guidance Information...

### **End of Product-sensitive programming interface**

## Trademarks

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

BookManager	CICS	CUA
DB2	IBM	IMS
IMS/ESA	MQSeries	MVS
MVS/DFP	MVS/ESA	OS/390
RACF		

Lotus, Freelance, and Word Pro are trademarks of Lotus Development Corporation in the United States, or other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

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

UNIX is a registered trademark in the United States and/or other countries licensed exclusively through X/Open Company Limited.

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



---

## Glossary of terms and abbreviations

This glossary defines MQSeries terms and abbreviations used in this book. If you do not find the term you are looking for, see the Index or the *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994.

This glossary includes terms and definitions from the *American National Dictionary for Information Systems*, ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies may be purchased from the American National Standards Institute, 11 West 42 Street, New York, New York 10036. Definitions are identified by the symbol (A) after the definition.

### A

**abend reason code.** A 4-byte hexadecimal code that uniquely identifies a problem with MQSeries for OS/390. A complete list of MQSeries for OS/390 abend reason codes and their explanations is contained in the *MQSeries for OS/390 Messages and Codes* manual.

**active log.** See *recovery log*.

**adapter.** An interface between MQSeries for OS/390 and TSO, IMS, CICS, or batch address spaces. An adapter is an attachment facility that enables applications to access MQSeries services.

**address space.** The area of virtual storage available for a particular job.

**address space identifier (ASID).** A unique, system-assigned identifier for an address space.

**administrator commands.** MQSeries commands used to manage MQSeries objects, such as queues, processes, and namelists.

**affinity.** An association between objects that have some relationship or dependency upon each other.

**alert.** A message sent to a management services focal point in a network to identify a problem or an impending problem.

**alert monitor.** In MQSeries for OS/390, a component of the CICS adapter that handles unscheduled events occurring as a result of connection requests to MQSeries for OS/390.

**alias queue object.** An MQSeries object, the name of which is an alias for a base queue defined to the local queue manager. When an application or a queue

manager uses an alias queue, the alias name is resolved and the requested operation is performed on the associated base queue.

**allied address space.** See *ally*.

**ally.** An OS/390 address space that is connected to MQSeries for OS/390.

**alternate user security.** A security feature in which the authority of one user ID can be used by another user ID; for example, to open an MQSeries object.

**APAR.** Authorized program analysis report.

**application environment.** The software facilities that are accessible by an application program. On the OS/390 platform, CICS and IMS are examples of application environments.

**application queue.** A queue used by an application.

**archive log.** See *recovery log*.

**ARM.** Automatic Restart Management

**ASID.** Address space identifier.

**asynchronous messaging.** A method of communication between programs in which programs place messages on message queues. With asynchronous messaging, the sending program proceeds with its own processing without waiting for a reply to its message. Contrast with *synchronous messaging*.

**attribute.** One of a set of properties that defines the characteristics of an MQSeries object.

**authorization checks.** Security checks that are performed when a user tries to issue administration commands against an object, for example to open a queue or connect to a queue manager.

**authorized program analysis report (APAR).** A report of a problem caused by a suspected defect in a current, unaltered release of a program.

**Automatic Restart Management (ARM).** An OS/390 recovery function that can improve the availability of specific batch jobs or started tasks, and therefore result in faster resumption of productive work.

### B

**backout.** An operation that reverses all the changes made during the current unit of recovery or unit of work. After the operation is complete, a new unit of recovery or unit of work begins. Contrast with *commit*.

## Glossary

**basic mapping support (BMS).** An interface between CICS and application programs that formats input and output display data and routes multiple-page output messages without regard for control characters used by various terminals.

**BMS.** Basic mapping support.

**bootstrap data set (BSDS).** A VSAM data set that contains:

- An inventory of all active and archived log data sets known to MQSeries for OS/390
- A wrap-around inventory of all recent MQSeries for OS/390 activity

The BSDS is required if the MQSeries for OS/390 subsystem has to be restarted.

**browse.** In message queuing, to use the MQGET call to copy a message without removing it from the queue. See also *get*.

**browse cursor.** In message queuing, an indicator used when browsing a queue to identify the message that is next in sequence.

**BSDS.** Bootstrap data set.

**buffer pool.** An area of main storage used for MQSeries for OS/390 queues, messages, and object definitions. See also *page set*.

## C

**call back.** In MQSeries, a requester message channel initiates a transfer from a sender channel by first calling the sender, then closing down and awaiting a call back.

**CCF.** Channel control function.

**CCSID.** Coded character set identifier.

**CDF.** Channel definition file.

**channel.** See *message channel*.

**channel control function (CCF).** In MQSeries, a program to move messages from a transmission queue to a communication link, and from a communication link to a local queue, together with an operator panel interface to allow the setup and control of channels.

**channel definition file (CDF).** In MQSeries, a file containing communication channel definitions that associate transmission queues with communication links.

**channel event.** An event indicating that a channel instance has become available or unavailable. Channel events are generated on the queue managers at both ends of the channel.

**checkpoint.** A time when significant information is written on the log. Contrast with *syncpoint*.

**CI.** Control interval.

**CL.** Control Language.

**client.** A run-time component that provides access to queuing services on a server for local user applications. The queues used by the applications reside on the server. See also *MQSeries client*.

**client application.** An application, running on a workstation and linked to a client, that gives the application access to queuing services on a server.

**client connection channel type.** The type of MQI channel definition associated with an MQSeries client. See also *server connection channel type*.

**cluster.** A network of queue managers that are logically associated in some way.

**cluster queue.** A queue that is hosted by a cluster queue manager and made available to other queue managers in the cluster.

**cluster queue manager.** A queue manager that is a member of a cluster. A queue manager may be a member of more than one cluster.

**cluster transmission queue.** A transmission queue that transmits all messages from a queue manager to any other queue manager that is in the same cluster. The queue is called SYSTEM.CLUSTER.TRANSMIT.QUEUE.

**coded character set identifier (CCSID).** The name of a coded set of characters and their code point assignments.

**command.** In MQSeries, an administration instruction that can be carried out by the queue manager.

**command prefix (CPF).** In MQSeries for OS/390, a character string that identifies the queue manager to which MQSeries for OS/390 commands are directed, and from which MQSeries for OS/390 operator messages are received.

**command processor.** The MQSeries component that processes commands.

**command server.** The MQSeries component that reads commands from the system-command input queue, verifies them, and passes valid commands to the command processor.

**commit.** An operation that applies all the changes made during the current unit of recovery or unit of work. After the operation is complete, a new unit of recovery or unit of work begins. Contrast with *backout*.

**completion code.** A return code indicating how an MQI call has ended.

**connect.** To provide a queue manager connection handle, which an application uses on subsequent MQI calls. The connection is made either by the MQCONN or MQCONNX call, or automatically by the MQOPEN call.

**connection handle.** The identifier or token by which a program accesses the queue manager to which it is connected.

**context.** Information about the origin of a message.

**context security.** In MQSeries, a method of allowing security to be handled such that messages are obliged to carry details of their origins in the message descriptor.

**control interval (CI).** A fixed-length area of direct access storage in which VSAM stores records and creates distributed free spaces. The control interval is the unit of information that VSAM transmits to or from direct access storage.

**controlled shutdown.** See *quiesced shutdown*.

**CPF.** Command prefix.

**Cross Systems Coupling Facility (XCF).** Provides the OS/390 coupling services that allow authorized programs in a multisystem environment to communicate with programs on the same or different OS/390 systems.

**coupling facility.** On OS/390, a special logical partition that provides high-speed caching, list processing, and locking functions in a parallel sysplex.

## D

**datagram.** The simplest message that MQSeries supports. This type of message does not require a reply.

**DCI.** Data conversion interface.

**dead-letter queue (DLQ).** A queue to which a queue manager or application sends messages that it cannot deliver to their correct destination.

**default object.** A definition of an object (for example, a queue) with all attributes defined. If a user defines an object but does not specify all possible attributes for that object, the queue manager uses default attributes in place of any that were not specified.

**deferred connection.** A pending event that is activated when a CICS subsystem tries to connect to MQSeries for OS/390 before MQSeries for OS/390 has been started.

**dequeue.** To remove a message from a queue. Contrast with *enqueue*.

**distributed application.** In message queuing, a set of application programs that can each be connected to a different queue manager, but that collectively constitute a single application.

**distributed queue management (DQM).** In message queuing, the setup and control of message channels to queue managers on other systems.

**DLQ.** Dead-letter queue.

**DQM.** Distributed queue management.

**dual logging.** A method of recording MQSeries for OS/390 activity, where each change is recorded on two data sets, so that if a restart is necessary and one data set is unreadable, the other can be used. Contrast with *single logging*.

**dual mode.** See *dual logging*.

**dynamic queue.** A local queue created when a program opens a model queue object. See also *permanent dynamic queue* and *temporary dynamic queue*.

## E

**enqueue.** To put a message on a queue. Contrast with *dequeue*.

**environment.** See *application environment*.

**ESM.** External security manager.

**event.** See *channel event*, *instrumentation event*, *performance event*, and *queue manager event*.

**event data.** In an event message, the part of the message data that contains information about the event (such as the queue manager name, and the application that gave rise to the event). See also *event header*.

**event header.** In an event message, the part of the message data that identifies the event type of the reason code for the event.

**event message.** Contains information (such as the category of event, the name of the application that caused the event, and queue manager statistics) relating to the origin of an instrumentation event in a network of MQSeries systems.

**event queue.** The queue onto which the queue manager puts an event message after it detects an event. Each category of event (queue manager, performance, or channel event) has its own event queue.

## Glossary

**external security manager (ESM).** A security product that is invoked by the OS/390 System Authorization Facility. RACF is an example of an ESM.

## F

**FIFO.** First-in-first-out.

**first-in-first-out (FIFO).** A queuing technique in which the next item to be retrieved is the item that has been in the queue for the longest time. (A)

**forced shutdown.** A type of shutdown of the CICS adapter where the adapter immediately disconnects from MQSeries for OS/390, regardless of the state of any currently active tasks. Contrast with *quiesced shutdown*.

## G

**GCPC.** Generalized command preprocessor.

**generalized command preprocessor (GCPC).** An MQSeries for OS/390 component that processes MQSeries commands and runs them.

**Generalized Trace Facility (GTF).** An OS/390 service program that records significant system events, such as supervisor calls and start I/O operations, for the purpose of problem determination.

**get.** In message queuing, to use the MQGET call to remove a message from a queue.

**global trace.** An MQSeries for OS/390 trace option where the trace data comes from the entire MQSeries for OS/390 subsystem.

| **globally-defined object.** On OS/390, an object whose  
| definition is stored in the shared repository. The object  
| is available to all queue managers in the queue-sharing  
| group. See also *locally-defined object*.

**GTF.** Generalized Trace Facility.

## H

**handle.** See *connection handle* and *object handle*.

**hardened message.** A message that is written to auxiliary (disk) storage so that the message will not be lost in the event of a system failure. See also *persistent message*.

## I

**immediate shutdown.** In MQSeries, a shutdown of a queue manager that does not wait for applications to disconnect. Current MQI calls are allowed to complete,

but new MQI calls fail after an immediate shutdown has been requested. Contrast with *quiesced shutdown* and *preemptive shutdown*.

| **inbound channel.** A channel that listens for and  
| receives messages from another queue manager. See  
| also *shared inbound channel*.

**in-doubt unit of recovery.** In MQSeries, the status of a unit of recovery for which a syncpoint has been requested but not yet confirmed.

**initialization input data sets.** Data sets used by MQSeries for OS/390 when it starts up.

**initiation queue.** A local queue on which the queue manager puts trigger messages.

**input/output parameter.** A parameter of an MQI call in which you supply information when you make the call, and in which the queue manager changes the information when the call completes or fails.

**input parameter.** A parameter of an MQI call in which you supply information when you make the call.

**instrumentation event.** A facility that can be used to monitor the operation of queue managers in a network of MQSeries systems. MQSeries provides instrumentation events for monitoring queue manager resource definitions, performance conditions, and channel conditions. Instrumentation events can be used by a user-written reporting mechanism in an administration application that displays the events to a system operator. They also allow applications acting as agents for other administration networks to monitor reports and create the appropriate alerts.

**Interactive Problem Control System (IPCS).** A component of OS/390 that permits online problem management, interactive problem diagnosis, online debugging for disk-resident abend dumps, problem tracking, and problem reporting.

**Interactive System Productivity Facility (ISPF).** An IBM licensed program that serves as a full-screen editor and dialog manager. It is used for writing application programs, and provides a means of generating standard screen panels and interactive dialogues between the application programmer and terminal user.

**IPCS.** Interactive Problem Control System.

**ISPF.** Interactive System Productivity Facility.

## L

**listener.** In MQSeries distributed queuing, a program that monitors for incoming network connections.

**local definition.** An MQSeries object belonging to a local queue manager.

**local definition of a remote queue.** An MQSeries object belonging to a local queue manager. This object defines the attributes of a queue that is owned by another queue manager. In addition, it is used for queue-manager aliasing and reply-to-queue aliasing.

**local queue.** A queue that belongs to the local queue manager. A local queue can contain a list of messages waiting to be processed. Contrast with *remote queue*.

**local queue manager.** The queue manager to which a program is connected and that provides message queuing services to the program. Queue managers to which a program is not connected are called *remote queue managers*, even if they are running on the same system as the program.

| **locally-defined object.** On OS/390, an object whose  
| definition is stored on page set zero. The definition can  
| be accessed only by the queue manager that defined it.  
| Also known as a *privately-defined object*.

**log.** In MQSeries, a file recording the work done by queue managers while they receive, transmit, and deliver messages, to enable them to recover in the event of failure.

**logical unit of work (LUW).** See *unit of work*.

## M

**machine check interrupt.** An interruption that occurs as a result of an equipment malfunction or error. A machine check interrupt can be either hardware recoverable, software recoverable, or nonrecoverable.

**MCA.** Message channel agent.

**MCI.** Message channel interface.

**message.** In message queuing applications, a communication sent between programs. In system programming, information intended for the terminal operator or system administrator.

**message channel.** In distributed message queuing, a mechanism for moving messages from one queue manager to another. A message channel comprises two message channel agents (a sender at one end and a receiver at the other end) and a communication link. Contrast with *MQI channel*.

**message channel agent (MCA).** A program that transmits prepared messages from a transmission queue to a communication link, or from a communication link to a destination queue. See also *message queue interface*.

**message channel interface (MCI).** The MQSeries interface to which customer- or vendor-written programs that transmit messages between an MQSeries queue manager and another messaging system must conform. A part of the MQSeries Framework.

**message descriptor.** Control information describing the message format and presentation that is carried as part of an MQSeries message. The format of the message descriptor is defined by the MQMD structure.

**message priority.** In MQSeries, an attribute of a message that can affect the order in which messages on a queue are retrieved, and whether a trigger event is generated.

**message queue.** Synonym for *queue*.

**message queue interface (MQI).** The programming interface provided by the MQSeries queue managers. This programming interface allows application programs to access message queuing services.

**message queuing.** A programming technique in which each program within an application communicates with the other programs by putting messages on queues.

**message sequence numbering.** A programming technique in which messages are given unique numbers during transmission over a communication link. This enables the receiving process to check whether all messages are received, to place them in a queue in the original order, and to discard duplicate messages.

**messaging.** See *synchronous messaging* and *asynchronous messaging*.

**model queue object.** A set of queue attributes that act as a template when a program creates a dynamic queue.

**MQI.** Message queue interface.

**MQI channel.** Connects an MQSeries client to a queue manager on a server system, and transfers only MQI calls and responses in a bidirectional manner. Contrast with *message channel*.

**MQSC.** MQSeries commands.

**MQSeries.** A family of IBM licensed programs that provides message queuing services.

## N

**namelist.** An MQSeries object that contains a list of names, for example, queue names.

**nonpersistent message.** A message that does not survive a restart of the queue manager. Contrast with *persistent message*.

**null character.** The character that is represented by X'00'.

## Glossary

### O

**object.** In MQSeries, an object is a queue manager, a queue, a process definition, a channel, a namelist, or a storage class (OS/390 only).

**object descriptor.** A data structure that identifies a particular MQSeries object. Included in the descriptor are the name of the object and the object type.

**object handle.** The identifier or token by which a program accesses the MQSeries object with which it is working.

**off-loading.** In MQSeries for OS/390, an automatic process whereby a queue manager's active log is transferred to its archive log.

**Open Transaction Manager Access (OTMA).** A transaction-based, connectionless client/server protocol. It functions as an interface for host-based communications servers accessing IMS TM applications through the OS/390 Cross Systems Coupling Facility (XCF). OTMA is implemented in an OS/390 sysplex environment. Therefore, the domain of OTMA is restricted to the domain of XCF.

**OTMA.** Open Transaction Manager Access.

| **outbound channel.** A channel that takes messages  
| from a transmission queue and sends them to another  
| queue manager. See also *shared outbound channel*.

**output log-buffer.** In MQSeries for OS/390, a buffer that holds recovery log records before they are written to the archive log.

**output parameter.** A parameter of an MQI call in which the queue manager returns information when the call completes or fails.

### P

**page set.** A VSAM data set used when MQSeries for OS/390 moves data (for example, queues and messages) from buffers in main storage to permanent backing storage (DASD).

**pending event.** An unscheduled event that occurs as a result of a connect request from a CICS adapter.

**percolation.** In error recovery, the passing along a preestablished path of control from a recovery routine to a higher-level recovery routine.

**performance event.** A category of event indicating that a limit condition has occurred.

**performance trace.** An MQSeries trace option where the trace data is to be used for performance analysis and tuning.

**permanent dynamic queue.** A dynamic queue that is deleted when it is closed only if deletion is explicitly requested. Permanent dynamic queues are recovered if the queue manager fails, so they can contain persistent messages. Contrast with *temporary dynamic queue*.

**persistent message.** A message that survives a restart of the queue manager. Contrast with *nonpersistent message*.

**ping.** In distributed queuing, a diagnostic aid that uses the exchange of a test message to confirm that a message channel or a TCP/IP connection is functioning.

**platform.** In MQSeries, the operating system under which a queue manager is running.

**point of recovery.** In MQSeries for OS/390, the term used to describe a set of backup copies of MQSeries for OS/390 page sets and the corresponding log data sets required to recover these page sets. These backup copies provide a potential restart point in the event of page set loss (for example, page set I/O error).

**preemptive shutdown.** In MQSeries, a shutdown of a queue manager that does not wait for connected applications to disconnect, nor for current MQI calls to complete. Contrast with *immediate shutdown* and *quiesced shutdown*.

| **privately-defined object.** In OS/390, an object whose  
| definition is stored on page set zero. The definition can  
| be accessed only by the queue manager that defined it.  
| Also known as a *locally-defined object*.

**process definition object.** An MQSeries object that contains the definition of an MQSeries application. For example, a queue manager uses the definition when it works with trigger messages.

### Q

**queue.** An MQSeries object. Message queuing applications can put messages on, and get messages from, a queue. A queue is owned and maintained by a queue manager. Local queues can contain a list of messages waiting to be processed. Queues of other types cannot contain messages—they point to other queues, or can be used as models for dynamic queues.

**queue manager.** A system program that provides queuing services to applications. It provides an application programming interface so that programs can access messages on the queues that the queue manager owns. See also *local queue manager* and *remote queue manager*. An MQSeries object that defines the attributes of a particular queue manager.

**queue manager event.** An event that indicates:

- An error condition has occurred in relation to the resources used by a queue manager. For example, a queue is unavailable.
- A significant change has occurred in the queue manager. For example, a queue manager has stopped or started.

**queue-sharing group.** In MQSeries for OS/390, a group of queue managers in the same sysplex that can access a single set of object definitions stored in the shared repository, and a single set of shared queues stored in the coupling facility. See also *shared queue*.

**queuing.** See *message queuing*.

**quiesced shutdown.** In MQSeries, a shutdown of a queue manager that allows all connected applications to disconnect. Contrast with *immediate shutdown* and *preemptive shutdown*. A type of shutdown of the CICS adapter where the adapter disconnects from MQSeries, but only after all the currently active tasks have been completed. Contrast with *forced shutdown*.

**quiescing.** In MQSeries, the state of a queue manager prior to it being stopped. In this state, programs are allowed to finish processing, but no new programs are allowed to start.

## R

**RBA.** Relative byte address.

**reason code.** A return code that describes the reason for the failure or partial success of an MQI call.

**receiver channel.** In message queuing, a channel that responds to a sender channel, takes messages from a communication link, and puts them on a local queue.

**recovery log.** In MQSeries for OS/390, data sets containing information needed to recover messages, queues, and the MQSeries subsystem. MQSeries for OS/390 writes each record to a data set called the *active log*. When the active log is full, its contents are off-loaded to a DASD or tape data set called the *archive log*. Synonymous with *log*.

**relative byte address (RBA).** The displacement in bytes of a stored record or control interval from the beginning of the storage space allocated to the data set to which it belongs.

**remote queue.** A queue belonging to a remote queue manager. Programs can put messages on remote queues, but they cannot get messages from remote queues. Contrast with *local queue*.

**remote queue manager.** To a program, a queue manager that is not the one to which the program is connected.

**remote queue object.** See *local definition of a remote queue*.

**remote queuing.** In message queuing, the provision of services to enable applications to put messages on queues belonging to other queue managers.

**reply message.** A type of message used for replies to request messages. Contrast with *request message* and *report message*.

**reply-to queue.** The name of a queue to which the program that issued an MQPUT call wants a reply message or report message sent.

**report message.** A type of message that gives information about another message. A report message can indicate that a message has been delivered, has arrived at its destination, has expired, or could not be processed for some reason. Contrast with *reply message* and *request message*.

**repository.** A collection of information about the queue managers that are members of a cluster. This information includes queue manager names, their locations, their channels, what queues they host, and so on.

**repository queue manager.** A queue manager that hosts the *full repository* of information about a cluster.

**requester channel.** In message queuing, a channel that may be started remotely by a sender channel. The requester channel accepts messages from the sender channel over a communication link and puts the messages on the local queue designated in the message. See also *server channel*.

**request message.** A type of message used to request a reply from another program. Contrast with *reply message* and *report message*.

**RESLEVEL.** In MQSeries for OS/390, an option that controls the number of CICS user IDs checked for API-resource security in MQSeries for OS/390.

**resolution path.** The set of queues that are opened when an application specifies an alias or a remote queue on input to an MQOPEN call.

**resource.** Any facility of the computing system or operating system required by a job or task. In MQSeries for OS/390, examples of resources are buffer pools, page sets, log data sets, queues, and messages.

**resource manager.** An application, program, or transaction that manages and controls access to shared resources such as memory buffers and data sets. MQSeries, CICS, and IMS are resource managers.

**Resource Recovery Services (RRS).** An OS/390 facility that provides 2-phase syncpoint support across participating resource managers.

## Glossary

**responder.** In distributed queuing, a program that replies to network connection requests from another system.

**resynch.** In MQSeries, an option to direct a channel to start up and resolve any in-doubt status messages, but without restarting message transfer.

**return codes.** The collective name for completion codes and reason codes.

**rollback.** Synonym for *back out*.

**RRS.** Resource Recovery Services.

## S

**SAF.** System Authorization Facility.

**security enabling interface (SEI).** The MQSeries interface to which customer- or vendor-written programs that check authorization, supply a user identifier, or perform authentication must conform. A part of the MQSeries Framework.

**SEI.** Security enabling interface.

**sender channel.** In message queuing, a channel that initiates transfers, removes messages from a transmission queue, and moves them over a communication link to a receiver or requester channel.

**sequential delivery.** In MQSeries, a method of transmitting messages with a sequence number so that the receiving channel can reestablish the message sequence when storing the messages. This is required where messages must be delivered only once, and in the correct order.

**sequential number wrap value.** In MQSeries, a method of ensuring that both ends of a communication link reset their current message sequence numbers at the same time. Transmitting messages with a sequence number ensures that the receiving channel can reestablish the message sequence when storing the messages.

**server.** (1) In MQSeries, a queue manager that provides queue services to client applications running on a remote workstation. (2) The program that responds to requests for information in the particular two-program, information-flow model of client/server. See also *client*.

**server channel.** In message queuing, a channel that responds to a requester channel, removes messages from a transmission queue, and moves them over a communication link to the requester channel.

**server connection channel type.** The type of MQI channel definition associated with the server that runs a queue manager. See also *client connection channel type*.

**service interval.** A time interval, against which the elapsed time between a put or a get and a subsequent get is compared by the queue manager in deciding whether the conditions for a service interval event have been met. The service interval for a queue is specified by a queue attribute.

**service interval event.** An event related to the service interval.

| **session ID.** In MQSeries for OS/390, the CICS-unique  
| identifier that defines the communication link to be  
| used by a message channel agent when moving  
| messages from a transmission queue to a link.

**shared inbound channel.** In MQSeries for OS/390, a channel that was started by a listener using the group port. The channel definition of a shared channel can be stored either on page set zero (private) or in the shared repository (global).

**shared outbound channel.** In MQSeries for OS/390, a channel that moves messages from a shared transmission queue. The channel definition of a shared channel can be stored either on page set zero (private) or in the shared repository (global).

| **shared queue.** In MQSeries for OS/390, a type of local  
| queue. The messages on the queue are stored in the  
| *coupling facility* and can be accessed by one or more  
| queue managers in a *queue-sharing group*. The definition  
| of the queue is stored in the *shared repository*.

| **shared repository.** In MQSeries for OS/390, a shared  
| DB2 database that is used to hold object definitions that  
| have been defined globally.

**shutdown.** See *immediate shutdown*, *preemptive shutdown*, and *quiesced shutdown*.

**signaling.** In MQSeries for OS/390 and MQSeries for Windows® 2.1, a feature that allows the operating system to notify a program when an expected message arrives on a queue.

**single logging.** A method of recording MQSeries for OS/390 activity where each change is recorded on one data set only. Contrast with *dual logging*.

**single-phase backout.** A method in which an action in progress must not be allowed to finish, and all changes that are part of that action must be undone.

**single-phase commit.** A method in which a program can commit updates to a queue without coordinating those updates with updates the program has made to resources controlled by another resource manager. Contrast with *two-phase commit*.

**SIT.** System initialization table.



**storage class.** In MQSeries for OS/390, a storage class defines the page set that is to hold the messages for a particular queue. The storage class is specified when the queue is defined.

**store and forward.** The temporary storing of packets, messages, or frames in a data network before they are retransmitted toward their destination.

**subsystem.** In OS/390, a group of modules that provides function that is dependent on OS/390. For example, MQSeries for OS/390 is an OS/390 subsystem.

**supervisor call (SVC).** An OS/390 instruction that interrupts a running program and passes control to the supervisor so that it can perform the specific service indicated by the instruction.

**SVC.** Supervisor call.

**switch profile.** In MQSeries for OS/390, a RACF profile used when MQSeries starts up or when a refresh security command is issued. Each switch profile that MQSeries detects turns off checking for the specified resource.

**synchronous messaging.** A method of communication between programs in which programs place messages on message queues. With synchronous messaging, the sending program waits for a reply to its message before resuming its own processing. Contrast with *asynchronous messaging*.

**syncpoint.** An intermediate or end point during processing of a transaction at which the transaction's protected resources are consistent. At a syncpoint, changes to the resources can safely be committed, or they can be backed out to the previous syncpoint.

**sysplex.** A multiple OS/390-system environment that allows multiple-console support (MCS) consoles to receive console messages and send operator commands across systems.

**System Authorization Facility (SAF).** An OS/390 facility through which MQSeries for OS/390 communicates with an external security manager such as RACF.

**system.command.input queue.** A local queue on which application programs can put MQSeries commands. The commands are retrieved from the queue by the command server, which validates them and passes them to the command processor to be run.

**system control commands.** Commands used to manipulate platform-specific entities such as buffer pools, storage classes, and page sets.

**system initialization table (SIT).** A table containing parameters used by CICS on start up.

## T

**target library high-level qualifier (thlqual).** High-level qualifier for OS/390 target data set names.

**task control block (TCB).** An OS/390 control block used to communicate information about tasks within an address space that are connected to an OS/390 subsystem such as MQSeries for OS/390 or CICS.

**task switching.** The overlapping of I/O operations and processing between several tasks. In MQSeries for OS/390, the task switcher optimizes performance by allowing some MQI calls to be executed under subtasks rather than under the main CICS TCB.

**TCB.** Task control block.

**temporary dynamic queue.** A dynamic queue that is deleted when it is closed. Temporary dynamic queues are not recovered if the queue manager fails, so they can contain nonpersistent messages only. Contrast with *permanent dynamic queue*.

**termination notification.** A pending event that is activated when a CICS subsystem successfully connects to MQSeries for OS/390.

**thlqual.** Target library high-level qualifier.

**thread.** In MQSeries, the lowest level of parallel execution available on an operating system platform.

**time-independent messaging.** See *asynchronous messaging*.

**TMI.** Trigger monitor interface.

**trace.** In MQSeries, a facility for recording MQSeries activity. The destinations for trace entries can include GTF and the system management facility (SMF).

**tranid.** See *transaction identifier*.

**transaction identifier.** In CICS, a name that is specified when the transaction is defined, and that is used to invoke the transaction.

**transmission program.** See *message channel agent*.

**transmission queue.** A local queue on which prepared messages destined for a remote queue manager are temporarily stored.

**trigger event.** An event (such as a message arriving on a queue) that causes a queue manager to create a trigger message on an initiation queue.

**triggering.** In MQSeries, a facility allowing a queue manager to start an application automatically when predetermined conditions on a queue are satisfied.

**trigger message.** A message containing information about the program that a trigger monitor is to start.

## Glossary

**trigger monitor.** A continuously-running application serving one or more initiation queues. When a trigger message arrives on an initiation queue, the trigger monitor retrieves the message. It uses the information in the trigger message to start a process that serves the queue on which a trigger event occurred.

**trigger monitor interface (TMI).** The MQSeries interface to which customer- or vendor-written trigger monitor programs must conform. A part of the MQSeries Framework.

**two-phase commit.** A protocol for the coordination of changes to recoverable resources when more than one resource manager is used by a single transaction. Contrast with *single-phase commit*.

## U

**undo/redo record.** A log record used in recovery. The redo part of the record describes a change to be made to an MQSeries object. The undo part describes how to back out the change if the work is not committed.

**unit of recovery.** A recoverable sequence of operations within a single resource manager. Contrast with *unit of work*.

**unit of work.** A recoverable sequence of operations performed by an application between two points of consistency. A unit of work begins when a transaction starts or after a user-requested syncpoint. It ends either at a user-requested syncpoint or at the end of a transaction. Contrast with *unit of recovery*.

**utility.** In MQSeries, a supplied set of programs that provide the system operator or system administrator with facilities in addition to those provided by the MQSeries commands. Some utilities invoke more than one function.

## X

**XCF.** Cross Systems Coupling Facility.

---

## Bibliography

This section describes the documentation available for all current MQSeries products.

---

### MQSeries cross-platform publications

Most of these publications, which are sometimes referred to as the MQSeries “family” books, apply to all MQSeries Level 2 products. The latest MQSeries Level 2 products are:

- MQSeries for AIX, V5.1
- MQSeries for AS/400, V5.1
- MQSeries for AT&T GIS UNIX V2.2
- MQSeries for Compaq (DIGITAL) OpenVMS, V2.2.1.1
- MQSeries for Compaq Tru64 UNIX, V5.1
- MQSeries for HP-UX, V5.1
- MQSeries for OS/2 Warp, V5.1
- MQSeries for OS/390, V5.2
- MQSeries for SINIX and DC/OSx, V2.2
- MQSeries for Sun Solaris, V5.1
- MQSeries for Sun Solaris, Intel Platform Edition, V5.1
- MQSeries for Tandem NonStop Kernel, V2.2.0.1
- MQSeries for VSE/ESA V2.1
- MQSeries for Windows V2.0
- MQSeries for Windows V2.1
- MQSeries for Windows NT, V5.1

The MQSeries cross-platform publications are:

- *MQSeries Brochure*, G511-1908
- *An Introduction to Messaging and Queuing*, GC33-0805
- *MQSeries Intercommunication*, SC33-1872
- *MQSeries Queue Manager Clusters*, SC34-5349
- *MQSeries Clients*, GC33-1632
- *MQSeries System Administration*, SC33-1873
- *MQSeries MQSC Command Reference*, SC33-1369
- *MQSeries Event Monitoring*, SC34-5760
- *MQSeries Programmable System Management*, SC33-1482
- *MQSeries Administration Interface Programming Guide and Reference*, SC34-5390
- *MQSeries Messages*, GC33-1876
- *MQSeries Application Programming Guide*, SC33-0807

- *MQSeries Application Programming Reference*, SC33-1673
- *MQSeries Programming Interfaces Reference Summary*, SX33-6095
- *MQSeries Using C++*, SC33-1877
- *MQSeries Using Java™*, SC34-5456
- *MQSeries Application Messaging Interface*, SC34-5604

---

### MQSeries platform-specific publications

Each MQSeries product is documented in at least one platform-specific publication, in addition to the MQSeries family books.

#### MQSeries for AIX, V5.1

*MQSeries for AIX Quick Beginnings*, GC33-1867

#### MQSeries for AS/400, V5.1

*MQSeries for AS/400® Quick Beginnings*, GC34-5557

*MQSeries for AS/400 System Administration*, SC34-5558

*MQSeries for AS/400 Application Programming Reference (ILE RPG)*, SC34-5559

#### MQSeries for AT&T GIS UNIX V2.2

*MQSeries for AT&T GIS UNIX® System Management Guide*, SC33-1642

#### MQSeries for Compaq (DIGITAL) OpenVMS, V2.2.1.1

*MQSeries for Digital OpenVMS System Management Guide*, GC33-1791

#### MQSeries for Compaq Tru64 UNIX, V5.1

*MQSeries for Compaq Tru64 UNIX Quick Beginnings*, GC34-5684

#### MQSeries for HP-UX, V5.1

*MQSeries for HP-UX Quick Beginnings*, GC33-1869

#### MQSeries for OS/2 Warp, V5.1

*MQSeries for OS/2 Warp Quick Beginnings*, GC33-1868

## Bibliography

### MQSeries for OS/390, V5.2

*MQSeries for OS/390 Concepts and Planning Guide*, GC34-5650

*MQSeries for OS/390 System Setup Guide*, SC34-5651

*MQSeries for OS/390 System Administration Guide*, SC34-5652

*MQSeries for OS/390 Problem Determination Guide*, GC34-5892

*MQSeries for OS/390 Messages and Codes*, GC34-5891

*MQSeries for OS/390 Licensed Program Specifications*, GC34-5893

*MQSeries for OS/390 Program Directory*

### MQSeries link for R/3 Version 1.2

*MQSeries link for R/3 User's Guide*, GC33-1934

### MQSeries for SINIX and DC/OSx, V2.2

*MQSeries for SINIX and DC/OSx System Management Guide*, GC33-1768

### MQSeries for Sun Solaris, V5.1

*MQSeries for Sun Solaris Quick Beginnings*, GC33-1870

### MQSeries for Sun Solaris, Intel Platform Edition, V5.1

*MQSeries for Sun Solaris, Intel Platform Edition Quick Beginnings*, GC34-5851

### MQSeries for Tandem NonStop Kernel, V2.2.0.1

*MQSeries for Tandem NonStop Kernel System Management Guide*, GC33-1893

### MQSeries for VSE/ESA V2.1

*MQSeries for VSE/ESA Version 2 Release 1 Licensed Program Specifications*, GC34-5365

*MQSeries for VSE/ESA™ System Management Guide*, GC34-5364

### MQSeries for Windows V2.0

*MQSeries for Windows User's Guide*, GC33-1822

### MQSeries for Windows V2.1

*MQSeries for Windows User's Guide*, GC33-1965

### MQSeries for Windows NT, V5.1

*MQSeries for Windows NT Quick Beginnings*, GC34-5389

*MQSeries for Windows NT® Using the Component Object Model Interface*, SC34-5387

*MQSeries LotusScript Extension*, SC34-5404

---

## Softcopy books

Most of the MQSeries books are supplied in both hardcopy and softcopy formats.

## HTML format

Relevant MQSeries documentation is provided in HTML format with these MQSeries products:

- MQSeries for AIX, V5.1
- MQSeries for AS/400, V5.1
- MQSeries for Compaq Tru64 UNIX, V5.1
- MQSeries for HP-UX, V5.1
- MQSeries for OS/2 Warp, V5.1
- MQSeries for OS/390, V5.2
- MQSeries for Sun Solaris, V5.1
- MQSeries for Windows NT, V5.1 (compiled HTML)
- MQSeries link for R/3 V1.2

The MQSeries books are also available in HTML format from the MQSeries product family Web site at:

<http://www.ibm.com/software/mqseries/>

## Portable Document Format (PDF)

PDF files can be viewed and printed using the Adobe Acrobat Reader.

If you need to obtain the Adobe Acrobat Reader, or would like up-to-date information about the platforms on which the Acrobat Reader is supported, visit the Adobe Systems Inc. Web site at:

<http://www.adobe.com/>

PDF versions of relevant MQSeries books are supplied with these MQSeries products:

- MQSeries for AIX, V5.1
- MQSeries for AS/400, V5.1
- MQSeries for Compaq Tru64 UNIX, V5.1
- MQSeries for HP-UX, V5.1
- MQSeries for OS/2 Warp, V5.1
- MQSeries for OS/390, V5.2
- MQSeries for Sun Solaris, V5.1
- MQSeries for Windows NT, V5.1
- MQSeries link for R/3 V1.2

PDF versions of all current MQSeries books are also available from the MQSeries product family Web site at:

<http://www.ibm.com/software/mqseries/>

## BookManager® format

The MQSeries library is supplied in IBM BookManager format on a variety of online library collection kits, including the *Transaction Processing and Data* collection kit, SK2T-0730. You can view the softcopy books in IBM BookManager format using the following IBM licensed programs:

- BookManager READ/2
- BookManager READ/6000
- BookManager READ/DOS
- BookManager READ/MVS
- BookManager READ/VM
- BookManager READ for Windows

## PostScript format

The MQSeries library is provided in PostScript (.PS) format with many MQSeries Version 2 products. Books in PostScript format can be printed on a PostScript printer or viewed with a suitable viewer.

## Windows Help format

The *MQSeries for Windows User's Guide* is provided in Windows Help format with MQSeries for Windows Version 2.0 and MQSeries for Windows Version 2.1.

---

## MQSeries information available on the Internet

The MQSeries product family Web site is at:

<http://www.ibm.com/software/mqseries/>

By following links from this Web site you can:

- Obtain latest information about the MQSeries product family.
- Access the MQSeries books in HTML and PDF formats.
- Download MQSeries SupportPacs.

## Related publications

For information about other products that are referred to in this book, see the following books:

### OS/390

- *MVS Setting up a Sysplex* GC28-1779
- *MVS Planning: APPC/MVS Management*, GC28-1807
- *OS/390 SecureWay CS IP Configuration*, SC31-8513

### CICS Transaction Server for OS/390

- *Resource Definition Guide*, SC33-1684
- *Operations and Utilities Guide*, SC33-1685
- *CICS-Supplied Transactions*, SC33-1686

### CICS for MVS/ESA Version 4

- *Resource Definition Guide*, SC33-1166
- *Operations and Utilities Guide*, SC33-1167
- *CICS-Supplied Transactions*, SC33-1168

### DB2

- *DB2 for OS/390 Administration Guide*, SC26-8957

### IMS

- *Customization Guide*, SC26-8020
- *Operator's Reference*, SC26-8030
- *Messages and Codes*, SC26-8028
- *Failure Analysis Structure Tables (FAST) for Dump Analysis*, LY27-9621

### DFSMS/MVS

- *Access Method Services for VSAM*, SC26-4905
- *Access Method Services for the Integrated Catalog Facility*, SC26-4906
- *Macro Instructions for Data Sets*, SC26-4913

## Related publications

---

# Index

## A

abend  
  application option of SSM entry 80  
  CICS transaction disconnecting 69  
  starting after 23  
  U3042 83

abnormal termination, restarting  
  after 125

access method services (AMS)  
  commands 102  
  deleting damaged BSDS 163  
  new active log definition 99  
  renaming damaged BSDS 163  
  REPRO 102, 112

accounting, what's new for this  
  release xiv

ACTION keyword, DLQ handler 231

action queue manager 10

active log  
  data set  
    copying 99  
    copying with AMS REPRO  
      statement 112  
  date 97  
  defining in BSDS 99  
  delays in off-loading 155  
  deleting from BSDS 100  
  enlarging 100  
  log print utility (CSQ1LOGP) 221  
  out of space 155  
  printing (CSQ1LOGP) 221  
  recording existing in BSDS 100  
  recovery plan, problems 151  
  status 98  
  stopped data set effect 152, 155  
  time 97

active log problem  
  both copies are damaged 152  
  delays in off-loading 155  
  dual logging lost 151  
  log stopped 151  
  one copy is damaged 152  
  out of space 155  
  read I/O errors 153  
  write I/O errors 153

active threads 140

address space  
  abend 22  
  canceling for MQSeries 24

administering by writing programs 27

administration programs 27, 28

alternative site recovery 127

AMS (access method services)  
  commands 102  
  deleting damaged BSDS 163  
  new active log definition 99  
  renaming damaged BSDS 163  
  REPRO 102, 112

AMS REPRO, backing up and recovering  
  page sets 112

API-crossing exit, enable or disable 56

Application Messaging Interface  
  (AMI) xv

application program  
  command format 48  
  CQKC DISPLAY 66  
  issuing commands from 27

application programming, what's new for  
  this release xv

APPLIDAT keyword, DLQ handler 230

APPLNAME keyword, DLQ  
  handler 230

APPLTYPE keyword, DLQ handler 230

ARCHIVE, utility function  
  (CSQJU003) 215

archive log  
  adding information to BSDS  
    (NEWLOG) 211  
  data set  
    password 215  
  date 97  
  deleting 94  
  deleting information from the  
    BSDS 101, 214  
  discarding records 94  
  log print utility (CSQ1LOGP) 221  
  password, changing 101  
  printing (CSQ1LOGP) 221  
  recording in BSDS 100  
  recovery plan 157  
  restarting 93  
  time 98

ARCHIVE LOG command 91

archive log problem  
  allocation problems 157  
  insufficient DASD for off-load 158  
  read I/O errors during restart 159  
  write I/O errors during off-load 157

archiving 91

ARM (Automatic Restart Manager)  
  activating a policy 135  
  clusters 137  
  couple data sets 134  
  defining a policy 134  
  introduction 133  
  LU6.2 136  
  network considerations 136  
  policy sample 134  
  queue-sharing groups 137  
  registering with 135  
  TCP/IP 137

Automatic Restart Manager (ARM)  
  activating a policy 135  
  clusters 137  
  couple data sets 134  
  defining a policy 134  
  introduction 133  
  LU 6.2 136  
  network considerations 136  
  policy sample 134  
  queue-sharing groups 137  
  registering with 135

Automatic Restart Manager (ARM)  
  (continued)  
  TCP/IP 137

## B

backing up page sets 111, 112

bibliography 259

blank fields in operations and control  
  panels 6

BMP (batch message program) 80

BookManager 261

bootstrap data set (BSDS)  
  adding an active log 99, 211  
  adding an archive log 100, 211  
  both copies are damaged 161  
  change log inventory utility  
    (CSQJU003) 99, 209  
  changing for active logs 99  
  changing for archive logs 100  
  changing log inventory utility  
    (CSQJU003) 101  
  deleting active log information 100  
  deleting archive log information 101  
  determining log inventory  
    contents 97  
  does not agree with log 161  
  error while opening 160  
  errors 160  
  I/O error 163  
  log print utility (CSQ1LOGP) 221  
  managing 97, 101  
  out of synchronization 162  
  print log map utility (CSQJU004) 219  
  recover log inventory 94  
  recovery 102  
  restoring from the archive log 102  
  single recovery 102  
  time stamps 97  
  unequal time stamps 162

BSDS (bootstrap data set)  
  adding an active log 99, 211  
  adding an archive log 100, 211  
  both copies are damaged 161  
  change log inventory utility  
    (CSQJU003) 99, 209  
  changing for active logs 99  
  changing for archive logs 100  
  changing log inventory utility  
    (CSQJU003) 101  
  deleting active log information 100  
  deleting archive log information 101  
  determining log inventory  
    contents 97  
  does not agree with log 161  
  error while opening 160  
  errors 160  
  I/O error 163  
  log print utility (CSQ1LOGP) 221  
  managing 97, 101  
  out of synchronization 162

- BSDS (bootstrap data set) *(continued)*
  - print log map utility (CSQJU004) 219
  - recover log inventory 94
  - recovery 102
  - restoring from the archive log 102
  - single recovery 102
  - time stamps 97
  - unequal time stamps 162
- buffer pool, associating with page sets 105
- building messages 30

## C

- canceling MQSeries address space 24
- CARTs 4
- CCSID keyword of COMMAND function 191
- CF (Coupling Facility)
  - adding a structure 122
  - load balancing 118
  - managing 122
  - moving a queue to another structure 118
  - recovering from failure 168
  - removing a structure 122
- CF structure
  - adding 122
  - load balancing 118
  - moving a queue to another 118
  - removing 122
- change log inventory utility (CSQJU003) 209
  - adding new active log 99, 155
  - ARCHIVE 215
  - change BSDS 97, 99
  - changes for active logs 99
  - changes for archive logs 100
  - CHECKPT 217
  - CRESTART 216
  - DELETE 214
  - functions
    - ARCHIVE 103
    - NEWLOG 99, 100
    - HIGHRBA 218
    - invoking 209
    - multiple statement operation 210
    - NEWLOG 211
    - time stamp in BSDS 162
- CHANGE SUBSYS, command of IMS 75, 79
- channel disposition 10
- channel initiator
  - restarting with ARM 136
  - what's new for this release xii
- checkpoint records, setting 217
- CHECKPT, utility function (CSQJU003) 217
- Chinese language feature 176
- CICS
  - definition of term xii
  - related publications 261
  - terminating 68
  - units of recovery 143
- CICS adapter
  - commands 47
  - connect program (CSQCQCON) 53
  - control panels 50

- CICS adapter *(continued)*
  - disconnect program 55
  - displaying CICS tasks 59
  - displaying connection details 59
  - displaying status 66
  - forced shutdown 68, 69
  - operation of
    - control panels 50
    - displaying current tasks 65
    - displaying instances of CKTI 64
    - lowercase queue names 52
    - modifying a connection 56
    - starting a connection 51
    - starting CKTI 60
    - stopping CKTI 62
  - orderly shutdown 68
  - passing parameters 48
  - quiesced shutdown 68
  - restart, what happens 141
  - shutting down a connection 68
  - starting a connection 51
  - task initiation program (CSQCSSQ) 61
  - terminating 68
- CICS bridge
  - starting 71
  - stopping 72
  - tuning considerations 72
- CKQC
  - DISPLAY command 66
  - MODIFY command 57
  - START command 52
  - STARTCKTI command 61
  - STOP command 55
  - STOPCKTI command 63
- CKQQ, transient data queue 49
- CKTI transaction
  - automating starting of 61
  - displaying 64
  - starting 60
  - stopping 62, 63
- client channel definition file 191, 193
- clusters and ARM 137
- CMDSCOPE, user messages from commands with 40
- cold start 130
- COMMAND, CSQUTIL function
  - MAKECLNT keyword 193
  - MAKEDEF keyword 192
- command and response tokens 4
- command line, using with operations and control panels 12
- command prefix string (CPF) 4
- command scope 10
- command server
  - restart 30
  - sending commands to 30
  - starting 30
  - stopping 30
- commands
  - action queue manager 10
  - choosing a queue manager 10
  - command prefix string (CPF) 4
  - command scope 10
  - examples of 37
  - for the CICS adapter 47
  - in request messages 30

- commands *(continued)*
  - issuing 3
    - from CSQUTIL 4, 179
    - from system-command input queue 27
    - from the OS/390 console 4
  - issuing through CSQUTIL 190
  - no reply to 43
  - operator 4
  - processor 30
  - remote queue manager 29
  - STOP QMGR 24
  - target queue manager 10
  - user messages
    - from DEFINE 37
    - from DEFINE THREAD 38
    - from DELETE 37
    - from DISPLAY commands 36
    - from DISPLAY QLOCAL 38
  - what's new for this release xiii
- conditional restart 216
- connections
  - controlling IMS 75
  - displaying 139
  - displaying details of
    - CICS 59
    - IMS 80
  - monitoring the activity on 80
  - starting from
    - CICS adapter control panel 51
    - CICS application program 53
    - CICS command line 52
    - IMS 76
  - stopping from
    - CICS adapter control panel 54
    - CICS application program 55
    - CICS command line 55
    - IMS 75
  - to IMS, monitoring activity 80, 81
- console, issuing commands from 3
- control panels for the CICS adapter 50
- COPY, CSQUTIL function 198
- COPY object disposition 10
- copying
  - messages from a queue (COPY) 179
  - page sets
    - RESETPAGE function 187
  - queues to a data set (COPY) 198
  - queues to a data set (SCOPY) 201
- COPYPAGE, CSQUTIL function 185
- CorrelId field, administration programs 33
- COUNT field, user messages 35
- couple data sets, ARM 134
- Coupling Facility (CF)
  - adding a structure 122
  - load balancing 118
  - managing 122
  - moving a queue to another structure 118
  - recovering from failure 168
  - removing a structure 122
- CPF (command prefix string) 4
- CRESTART, utility function (CSQJU003) 216
- CSQ1LOGP (log print utility)
  - finding start RBA with 144



- CSQ1LOGP (log print utility) *(continued)*
    - invoking 221
    - output 223
    - what it does 221
  - CSQ2020E message 87
  - CSQ4BSDS sample 102, 163
  - CSQ5PQSG (queue-sharing group utility)
    - invoking 225
    - what it is 225
  - CSQCDSC CICS adapter disconnect
    - program 55
  - CSQCQCOC CICS adapter connect
    - program 53
  - CSQCRST CICS adapter reset
    - program 58
  - CSQCSSQ CICS adapter task initiation
    - program 61, 63
  - CSQI063E message 150
  - CSQI004I message 151
  - CSQI030E message 151
  - CSQI100E message 160
  - CSQI102E message 152, 161
  - CSQI103E message 157
  - CSQI105E message 153
  - CSQI106E message 153
  - CSQI107E message 163
  - CSQI108E message 163
  - CSQI110E message 155
  - CSQI111A message 155
  - CSQI114I message 158
  - CSQI115E message 157
  - CSQI120E message 162
  - CSQI122E message 162
  - CSQI124E message 153
  - CSQI126E message 163
  - CSQI128E message 158
  - CSQI232E message 152
  - CSQJU003 (change log inventory utility) 209
    - adding new active log 99, 155
    - ARCHIVE 215
    - change BSDS 97, 99
    - changes for active logs 99
    - changes for archive logs 100
    - CHECKPT 217
    - CRESTART 216
    - DELETE 214
    - functions
      - ARCHIVE 103
      - NEWLOG 99, 100
      - HIGHRBA 218
    - invoking 209
    - multiple statement operation 210
    - NEWLOG 211
    - time stamp in BSDS 162
  - CSQJU004 (print log map utility)
    - BSDS time stamps 97
    - introduction 219
    - invoking 219
    - listing BSDS contents using 97
  - CSQP004E message 164
  - CSQP018I message 241
  - CSQP019I message 241
  - CSQQTRMN
    - starting 83
    - stopping 83
  - CSQQxxx messages 170
  - CSQUDLQH (dead-letter queue handler utility)
    - actions 230
    - conventions 233
    - example 237
    - invoking 227
    - patterns 230
    - processing 235
    - rules table 228
    - what it is 227
  - CSQUTIL (MQSeries utility program)
    - COMMAND 190
    - COPY 198
    - COPYPAGE 185
    - EMPTY 204
    - FORMAT 183
    - introduction 179
    - invoking 180
    - LOAD 206
    - monitoring progress 182
    - PARM parameters 180
    - RESETPAGE 187
    - return codes 181
    - SCOPY 201
    - SDEFS 195
    - unit of recovery, maximum number of messages 200
  - CSQV401 message 143
  - CSQV406 message 143
  - CSQZPARM, specifying an alternative 22
  - CTL (IMS control region) 76, 80
- ## D
- data sets
    - copying messages from queues 198
    - copying messages from queues (offline) 201
    - dump and restore 113
    - page set I/O error 164
    - restart on losing 130
    - restoring messages from 206
  - data-sharing group
    - recovering from failure 167
    - resynchronizing with MQSeries 167
  - DB2
    - adding a queue-sharing group 117
    - recovering from system failure 166
    - related publications 261
    - removing a queue-sharing group 118
    - resynchronizing with MQSeries 167
  - DB2 tables
    - adding a queue manager 225
    - adding a queue-sharing group 226
    - removing a queue manager 226
    - removing a queue-sharing group 226
  - dead-letter header, MQDLH 227
  - dead-letter queue
    - finding out its name 37
    - what's new for this release xv
  - dead-letter queue handler utility (CSQUDLQH)
    - actions 230
    - conventions 233
    - example 237
    - invoking 227
    - patterns 230
  - dead-letter queue handler utility (CSQUDLQH) *(continued)*
    - processing 235
    - rules table 228
    - what it is 227
  - DEFINE commands, user messages 37
  - DELETE, utility function (CSQJU003) 214
  - DELETE commands, user messages 37
  - deleting
    - active information log from
      - BSDS 100
      - archive logs 94, 95
      - IMS Tpipes 88
      - log information from BSDS 214
      - messages from a queue 204
    - dependent region, IMS 80
    - disconnecting from 81
    - DEQUEUE TMEMBER, command of IMS 88
    - DESTQ keyword, DLQ handler 230
    - DESTQM keyword, DLQ handler 230
    - DFS3611 message 170
    - DFS555I message 170
    - DFSMS, related publications 261
    - disaster recovery 127
    - discarded messages 34
    - disconnecting
      - from CICS 54
      - from IMS 82
    - display CKQC transaction 66
    - DISPLAY commands, user messages 38
    - DISPLAY OASN command of IMS 79
    - displaying
      - function key settings 11
      - units of recovery in CICS 143
      - units of recovery in IMS 78, 146
    - disposition, object 10
    - DLQ handler utility 227
    - DNS (Domain Name System) 138
    - Domain Name System (DNS) 138
    - dual logging , losing 151
- ## E
- editing namelists 20
  - EMCS 4
  - EMPTY, utility function (CSQUTIL) 204
  - errors, hardware 172
  - example ARM policy 134
  - example recovery scenarios
    - active log problems
      - both copies are damaged 152
      - delays in off-loading 155
      - dual logging lost 151
      - log stopped 151
      - one copy is damaged 152
      - out of space 155
      - read I/O errors 153
      - write I/O errors 153
    - archive log problems
      - allocation problems 157
      - insufficient DASD for
        - off-load 158
      - read I/O errors during
        - restart 159
      - write I/O errors during
        - off-load 157

example recovery scenarios (*continued*)

- BSDS problems
  - both copies are damaged 161
  - BSDS recovery 102
  - does not agree with log 161
  - error while opening 160
  - I/O error 163
  - out of synchronization 162
  - unequal time stamps 162
- hardware problems 172
- IMS problems
  - application terminates 170
  - IMS not operational 171
  - unable to connect to MQSeries 170
- page set problems
  - I/O error 164
  - page set full 165

EXEC CICS LINK

- INPUTMSG option 48
  - linking to the CICS adapter 53
- expanding page sets 185
- extended console support 4

## F

- FAILURE keyword of COMMAND function 191
- FEEDBACK keyword, DLQ handler 230
- finding archive log data sets to be deleted 95
- FORCE keyword of FORMAT 183
- FORCE keyword of RESETPAGE 188
- FORMAT, utility function (CSQUTIL) 183
- FORMAT keyword, DLQ handler 231
- function keys
  - changing namelists 20
  - operations and control panels 11
  - showing 11
  - using 11
- functions, return codes from CSQUTIL 181
- FWDQ keyword, DLQ handler 232
- FWDQM keyword, DLQ handler 232

## G

- glossary 249
- GROUP object disposition 10
- group objects, managing 122

## H

- hardware errors 172
- HEADER keyword, DLQ handler 232
- help
  - CICS adapter 50
  - operations and control panels 12
- HIGHRBA, utility function (CSQJU003) 218
- HTML (Hypertext Markup Language) 260
- Hypertext Markup Language (HTML) 260

## I

- I/O error
  - marks active log as TRUNCATED 98

I/O error (*continued*)

- queues 164
- IMS
  - abend U3042 83
  - commands
    - CHANGE SUBSYS 75, 79
    - DEQUEUE TMEMBER 88
    - DISPLAY OASN 79
    - DISPLAY OASN SUBSYS 75
    - DISPLAY SUBSYS 81
    - START REGION 81
    - START SUBSYS 75
    - START TMEMBER 86
    - STOP REGION 81
    - STOP SUBSYS 75, 82
    - STOP TMEMBER 86
    - TRACE 75
  - connection status 81
  - definition of term xii
  - deleting Tpipes 88
  - disconnecting from dependent region 81
  - in-doubt units of recovery 145
  - related problems 170
  - related publications 261
  - resynchronizing the bridge 87
- IMS adapter
  - connection status 81
  - control region 76
  - controlling dependent region connections 80
  - dependent regions of IMS 80
  - displaying in-doubt units of recovery 78
  - IMSID option 76
  - initializing 76
  - logical terminal (LTERM) 76
  - residual recovery entry (RRE) 79
  - restart, what happens 145
  - starting CSQQTRMN 83
  - stopping CSQQTRMN 83
  - thread 77
  - threads, displaying 78
- IMS bridge
  - Commit mode, synchronization 87
  - controlling queues 86
  - deleting messages 88
  - resynchronizing 87

IMS problem

- application terminates 170
  - IMS not operational 171
  - unable to connect to MQSeries 170
- IMS.PROCLIB library 76, 80
- in-doubt threads 140
- in-doubt units of recovery
  - CICS 141
  - IMS 146
- INPUTQ keyword, DLQ handler 229
- INPUTQM keyword, DLQ handler 229
- interpreting replies to messages 35
- ISPF, showing keys 11
- issuing commands 3, 179

## J

- Japanese language feature 176

## K

- KEYLIST, ISPF command 11

## L

- listener, restarting with ARM 136
- LOAD, utility function 206
- load balancing
  - CF structures 118
  - page set 107
  - sample job for a CF structure 119
  - sample job for a page set 108
- locating archive log data sets to be deleted 95
- log
  - archiving 91
  - change log inventory utility (CSQJU003) 209
  - determining inventory contents 97
  - error recovery procedures 151
  - log print utility (CSQ1LOGP) 221
  - managing 91
  - off-load, cancelling 93
  - optimizing tape reading 93
  - print log map utility (CSQJU004) 219
  - recovering from problems
    - active log 151
    - archive log 157
  - recovery 93
  - restarting archive process 93
  - what's new for this release xiv
- log data sets
  - restart on losing 126
- log inventory, change 209
- log print utility (CSQ1LOGP) 221
  - extract log records 221
  - finding start RBA with 144
  - invoking 221
  - output 223
  - print log records 93, 221
  - what it does 221
- log RBA, updating the highest written 218
- log RBA value, modifying 209
- lowercase queue names
  - CICS adapter 52
  - operations and control panels 6
- LU 6.2 and ARM 136

## M

- MAKECLNT, keyword of COMMAND function 191, 193
- MAKEDEF, keyword of COMMAND function 190, 192
- managing
  - BSDS 97, 99
  - MQSeries log 91
  - page sets 105
  - queue-sharing groups 117
  - shared queues 118
- maximum number of uncommitted messages 200
- MAXSMGS 200
- media recovery 172
- message processing program (MPP) 80
- messages
  - CICS adapter 49
  - discarded 34
  - incorporating MQSeries commands 30

- messages (*continued*)
  - interpreting replies to MQSeries
    - commands 35
    - maximum number of uncommitted 200
  - on the system-command input queue 32
  - user 11, 28
  - waiting for replies to 33
- MGCR and MGCRCRE 3
- migrating queues from non-shared to shared 121
- modifying an MQSeries-CICS connection 56
- monitoring
  - CICS connection activity 59
  - IMS connection activity 80
- moving queues
  - non-shared queue 107
  - shared queue 118
- MPP (message processing program)
  - connection control 80
- MQDLH, dead-letter header 227
- MQGET in administration programs 28
- MQPUT in administration programs 28
- MQSeries commands
  - DISPLAY THREAD 143
  - remote queue manager 29
  - RESOLVE INDOUBT 143
- MQSeries publications 259
- MQSeries utility program (CSQUTIL) 209
  - COMMAND 190
  - COPY 198
  - COPYPAGE 185
  - EMPTY 204
  - FORMAT 183
  - introduction 179
  - invoking 180
  - issuing commands from 4
  - LOAD 206
  - monitoring progress 182
  - PARM parameters 180
  - RESETPAGE 187
  - return codes 181
  - SCOPY 201
  - SDEFS 195
  - syntax checking 182
  - unit of recovery, maximum number of messages 200
- MsgId field, administration programs 33
- MSGTYPE keyword, DLQ handler 231

## N

- namelists 20
  - disposition 10
- network considerations for ARM 136
- new function xii
- NEWLOG, utility function (CSQJU003) 99, 100, 211
- NID (network ID) 143, 146
- nonpersistent messages 30

## O

- objects
  - altering 19

- objects (*continued*)
  - backing up definitions 113
  - defining 14
  - deleting 19
  - displaying 19
  - disposition 10
  - group, managing 122
  - operations and control panels 14
- off-loading, errors during 98
- opening the system-command input queue 29
- operating, basic operations 3
- operations and control panels
  - changing the subsystem ID 13
  - example of 13
  - function keys 11
  - invoking 5
  - queue manager default 11
  - rules for using 6
  - user messages 11
  - using 5
  - using the command line 12
  - what's new for this release xv
  - working with object definitions 19
- operator commands
  - CICS adapter 67
  - IMS adapter 75
  - issuing 3
  - operations and control panels 5
  - orderly shutdown, CICS adapter 68
  - out of space on active log 155

## P

- page set problem
  - I/O error 164
  - page set full 165
- page sets
  - adding 105
  - AMS REPRO 112
  - backing up 111, 112
  - copying 185, 187
  - COPYPAGE 185
  - creating a point of recovery 111
  - display usage 106
  - expanding 109, 185
  - formatting 183
  - full 106, 165
  - load balancing 107
  - managing 105
  - problems 164
  - recovery 113
  - reducing the size 110
  - RESETPAGE 187
  - resetting the log 187
  - restart on losing 125
  - utility functions 179
- PAGES keyword of FORMAT 183
- panels
  - blank fields in 6
  - operations and control 5, 13
  - rules for using 6
- PARM option, START QMGR command 22
- passwords
  - archive log data set 101
  - data sets 212
  - supply for archive log 215

- PDF (Portable Document Format) 260
- PERSIST keyword, DLQ handler 231
- PFSHOW, ISPF command 11
- point of recovery, creating 111
- Portable Document Format (PDF) 260
- PostScript format 261
- print log map utility (CSQJU004)
  - BSDS time stamps 97
  - introduction 219
  - invoking 219
  - listing BSDS contents using 97
- PRIVATE object disposition 10
- processes, disposition 10
- program, administration 27
- publications
  - MQSeries 259
  - related 261
- PUTAUT keyword, DLQ handler 232

## Q

- QMGR object disposition 10
- QSGDISP, user messages from commands with 42
- queue management utility functions 179
- queue managers
  - adding a page set 105
  - adding to a queue-sharing group 117
  - adding to DB2 tables 225
  - cold start 130
  - expanding a page set 109
  - reducing a page set 110
  - removing from a queue-sharing group 117
  - removing from DB2 tables 226
  - restarting 125
  - starting 22
  - stopping 24
- queue-sharing group utility (CSQ5PQSG)
  - invoking 225
  - what it is 225
- queue-sharing groups
  - adding a queue manager 117
  - adding to DB2 tables 117, 226
  - and ARM 137
  - cold start 131
  - load balancing 118
  - managing 117
  - moving a queue 118
  - recovering units of recovery 148
  - reinitializing queue managers 131
  - removing a queue manager 117
  - removing from DB2 tables 118, 226
  - user messages from commands 40
  - What's new for this release xii
- queues
  - copying 185, 198
  - copying (offline) 201
  - defining local 15
  - disposition 10
  - emptying 204
  - LOAD function 206
  - migrating non-shared to shared 121
  - moving a non-shared queue 107
  - moving a shared queue 118
  - moving non-shared to shared 120
  - reply-to model 28
  - restoring messages 206

queues (*continued*)  
  system-command input 29  
  system-command reply-to model 28  
QUIESCE, stop mode 24  
QUIESCE MODE of ARCHIVE LOG 91

## R

RBA (relative byte address), range  
  specified in active log 99  
REASON keyword, DLQ handler 231  
recovering shared queues 118  
recovery  
  active log problems 151  
  alternative site 127  
  basic operations 3  
  BSDS  
    errors 160  
    log inventory 94  
  CICS  
    manually recovering units of  
    recovery 143  
  COPY 115  
  creating a point of 111  
  description 113  
  example scenarios 149  
  IMS  
    manually recovering units of  
    recovery 145  
    resolving in-doubt units of  
    recovery 78  
    resynchronizing the bridge 87  
  IMS units of recovery 145  
  logs 93  
  long-running UOW 169  
  MQSeries-related problems  
    active log problems 151  
    archive log problems 157, 159  
    BSDS 105, 160  
    page set problems 164  
  page sets 112  
  point of 111  
  RRS  
    manually recovering units of  
    recovery 147  
  single BSDS 102  
  starting 21, 23  
  tokens 143  
reducing the size of a page set 110  
region error options (REO) 80  
registering with ARM 135  
reinitializing MQSeries 130  
relative byte address (RBA), range  
  specified in active log 99  
REO (region error options) 80  
replies, examples 37  
reply message descriptor 34  
reply messages 33  
reply-to queue  
  attributes 28  
  defining 28  
  opening 29  
REPLYQ keyword, DLQ handler 231  
REPLYQM keyword, DLQ handler 231  
REPRO command of access method  
  services 102, 112  
request message 32  
resetting page sets 187

RESOLVE INDOUBT command, free  
  locked resources 143  
resolving  
  in-doubt units of recovery 78  
  units of recovery 143, 147  
Resource Recovery Services (RRS), units  
  of recovery 147  
RESPTIME, keyword of COMMAND  
  function 191  
restart  
  after abnormal termination 125  
  after losing data sets 130  
  after losing logs 126  
  after losing page sets 125  
  CICS adapter 141  
  cold start 130  
  conditional 216  
  IMS adapter 145  
  long-running UOW 169  
  OS/390 Automatic Restart  
  Manager 133  
  user messages 142  
  with ARM 133  
restarting MQSeries 125  
restoring messages to a queue 179  
retention period, archive logs  
  (ARCRTN) 94  
RETRY keyword, DLQ handler 232  
RETRYINT keyword, DLQ handler 229  
return codes, from utility functions 181  
RRE (residual recovery entry) 79  
RRS (Resource Recovery Services), units  
  of recovery 147  
rules for using the operations and control  
  panels 6

## S

sample ARM policy 134  
SCOPY, CSQUTIL function 201  
security, what's new for this release xiv  
shared channels after DB2 failure 167  
SHARED object disposition 10  
shared queues  
  load balancing 118  
  managing 118  
  moving a queue 118  
  moving to non-shared 120  
  recovering 118  
  recovering units of recovery 148  
  user messages from commands 40  
shutting down the CICS bridge 72  
softcopy books 260  
SSM (subsystem member)  
  contains control information 76  
  error options 80  
  specified on EXEC parameter 80  
START CMDSERV command 30  
start options for MQSeries 22  
START QMGR command  
  from OS/390 console 21  
  options 22  
START REGION, command of IMS 81  
START SUBSYS, command of IMS 75  
START TMEMBER, command of IMS 86  
start-up messages (MQSeries) 241  
starting  
  after an abend 23

starting (*continued*)  
  CICS bridge 71  
  CICS-MQSeries connection  
    from a CICS program 53  
    from the command line 52  
    using the CICS adapter control  
    panels 51  
  command server 30  
  IMS-MQSeries connection 76  
  MQSeries 21, 22, 23  
  OS/390 Automatic Restart  
  Manager 133  
  with ARM 133  
statistics, what's new for this release xiv  
STOP CMDSERV command 30  
STOP QMGR command  
  MODE(FORCE) 24  
  MODE(QUIESCE) 24  
  MODE(RESTART) 24  
STOP REGION, command of IMS 81  
STOP SUBSYS, command of IMS 75, 82  
STOP TMEMBER, command of IMS 86  
stopping the CICS bridge 72  
storage classes, disposition 10  
storage management subsystem  
  (SMS) 94  
storage medium full 106  
  recovery scenario 166  
structure (Coupling Facility)  
  adding 122  
  load balancing 118  
  moving a queue to another 118  
  removing 122  
subsystem ID, changing 13  
SupportPac 261  
system administration  
  MQSeries commands 3, 21  
  using application programs 27  
system-command input queue 4  
  defining 28  
  opening 29  
  putting messages on 32  
system-command reply-to model  
  queue 28  
system control commands for starting  
  MQSeries 21  
system object samples, what's new for  
  this release xiv  
system parameter module  
  (CSQZPARM) 22  
system parameters, what's new for this  
  release xiii

## T

tape device, optimizing reading for  
  log 93  
target queue manager 10  
tasks, displaying CICS 65  
TCP/IP and ARM 137  
terminating  
  MQSeries 24  
  MQSeries-CICS connection 68  
    from a CICS program 55  
    from the CICS adapter control  
    panels 54  
    from the CICS command line 55  
  MQSeries-IMS connection 82

- terminology used in this book 249
- TGTQMGR, keyword of COMMAND
  - function 190
- thlqual, definition of term xii
- threads
  - active 140
  - attachment in IMS 77
  - CICS adapter termination 68
  - displaying 139
  - displaying, IMS adapter 78
  - IMS termination 82
  - in-doubt 140
  - stopping MQSeries 24
- time stamps 97
  - from BSDS 97
  - unequal in BSDS 162
- Tpipe, deleting 88
- TRACE, command of IMS 75
- trace, what's new for this release xiv
- transient data queue (TDQ), CKQQ 49
- tuning the CICS bridge 72

## U

- U.S. English language features 176
- U3042 abend (IMS) 83
- uncommitted messages, maximum number 200
- unit of recovery
  - CICS, recovering manually 143
  - displaying in-doubt 143
  - IMS
    - in-doubt resolution 78
    - recovering manually 145
  - in-doubt
    - displaying in IMS 78
    - recovering in IMS 78
  - maximum number of messages
    - in 200
  - recovering on another queue manager 148
  - RRS, recovering manually 147
- unit of work
  - CICS adapter 141
  - long running 169
  - RESOLVE INDOUBT command 78
- UOW 141
- user messages 28
  - at start up 241
  - COUNT field 35
  - displaying from panels 11
  - from MQSeries commands, replies 37
- USERID keyword, DLQ handler 231
- utilities
  - time stamp 97
- utility program (CSQUTIL)
  - COMMAND 190
  - COPY 198
  - COPYPAGE 185
  - EMPTY 204
  - FORMAT 183
  - introduction 179
  - invoking 180
  - LOAD 206
  - monitoring progress 182
  - PARM parameters 180
  - RESETPAGE 187

- utility program (CSQUTIL) (*continued*)
  - return codes 181
  - SCOPY 201
  - SDEFS 195
  - unit of recovery, maximum number of messages 200
- utility programs
  - change log inventory utility (CSQJU003) 209
  - dead-letter queue handler utility (CSQUDLQH) 227
  - log print utility (CSQ1LOGP) 221
  - MQSeries utility program (CSQUTIL) 179
  - print log map utility (CSQJU004) 219
  - queue-sharing group utility (CSQ5PQSG) 225
  - summary table 175

## V

- volume dump and restore 113
- VSAM (virtual storage access method) 211

## W

- WAIT keyword, DLQ handler 229
- waiting for replies to messages 33
- What's new for this release xii
- Windows Help 261
- work, units of 143
- writing programs to administer MQSeries 27
- WTOR, MQSeries-related 24



---

## Sending your comments to IBM

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book.

Please limit your comments to the information in this book and the way in which the information is presented.

**To make comments about the functions of IBM products or systems, talk to your IBM representative or to your IBM authorized remarketer.**

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate, without incurring any obligation to you.

You can send your comments to IBM in any of the following ways:

- By mail, to this address:

User Technologies Department (MP095)  
IBM United Kingdom Laboratories  
Hursley Park  
WINCHESTER,  
Hampshire  
SO21 2JN  
United Kingdom

- By fax:
  - From outside the U.K., after your international access code use 44-1962-870229
  - From within the U.K., use 01962-870229
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink™: HURSLEY(IDRCF)
  - Internet: idrcf@hursley.ibm.com

Whichever method you use, ensure that you include:

- The publication title and order number
- The topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.









Printed in the United States of America  
on recycled paper containing 10%  
recovered post-consumer fiber.

SC34-5652-00

