

WebSphere MQ



Programmable Command Formats and Administration Interface

Version 6.0

WebSphere MQ



Programmable Command Formats and Administration Interface

Version 6.0

Note!

Before using this information and the product it supports, be sure to read the general information under notices at the back of this book.

Second edition (October 2006)

This edition of the book applies to the following products:

- IBM WebSphere MQ Version 6.0
- IBM WebSphere MQ for z/OS Version 6.0

and to all subsequent releases and modifications until otherwise indicated in new editions.

Unless otherwise stated, the information also applies to the following products:

- MQSeries for Compaq NonStop Kernel, V5.1
- WebSphere MQ for HP OpenVMS, V5.3

© Copyright International Business Machines Corporation 1996, 2006. All rights reserved.

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

Contents

Figures	xii	Authority checking for PCF commands	18
Tables	xiii	WebSphere MQ for iSeries	18
About this book	xv	WebSphere MQ for Windows, and UNIX systems	19
Who this book is for	xv	WebSphere MQ for HP OpenVMS and Compaq	
What you need to know to understand this book	xvi	NonStop Kernel	20
How to use this book	xvi	WebSphere MQ for z/OS	23
Appendices	xvi		
Summary of changes	xvii		
Changes for this edition (SC34-6598-01)	xvii		
Changes for the previous edition (SC34-6598-00)	xvii		
Changes to PCF commands and responses	xvii		
Changes to the MQAI	xxi		
Changes to make this manual easier to use	xxii		
Part 1. Programmable Command Formats	1		
Chapter 1. Introduction to Programmable Command Formats	7		
The problem PCF commands solve	7	PCF commands and responses in groups	30
What PCFs are	8	Authentication Information commands	30
Other administration interfaces	8	Authority Record commands	30
WebSphere MQ for iSeries	8	CF commands	30
WebSphere MQ for z/OS	9	Channel commands	31
MQSeries for Compaq NonStop Kernel, V5.1	9	Cluster commands	31
WebSphere MQ for Windows, UNIX systems and HP OpenVMS	9	Connection commands	31
The WebSphere MQ Administration Interface (MQAI)	9	Escape command	31
Chapter 2. Using Programmable Command Formats	11	Namelist commands	31
PCF command messages	11	Process commands	32
How to issue PCF command messages	11	Queue commands	32
Message descriptor for a PCF command	11	Queue Manager commands	32
Sending user data	13	Security commands	32
Responses	13	Service commands	32
Message descriptor for a response	13	Storage class commands	32
Standard responses	14	System commands	33
OK response	14	Data responses to commands	33
Error response	14	Definitions of Programmable Command Formats	34
Data response	15	Backup CF Structure	34
Extended responses	15	Required parameters	34
Extended responses to Inquire commands	16	Optional parameters	34
Extended responses to commands other than Inquire	16	Change, Copy, and Create Authentication Information Object	35
Extended responses to commands using CommandScope	17	Required parameters (Change authentication information)	35
Rules for naming WebSphere MQ objects	17	Required parameters (Copy authentication information)	35
Name lengths	18	Required parameters (Create authentication information)	36
Generic values	18	Optional parameters	36
		Change, Copy, and Create CF Structure	38
		Required parameters (Change and Create CF Structure)	39
		Required parameters (Copy CF Structure)	39
		Optional parameters	39
		Change, Copy and Create Channel	41
		Required parameters (Change, Create Channel)	43
		Required parameters (Copy Channel)	43
		Optional parameters	44

Error codes	67
Change, Copy, and Create Channel Listener	70
Required parameters (Change and Create Channel Listener)	70
Required parameters (Copy Channel Listener) . .	70
Optional parameters	71
Change, Copy, and Create Namelist	72
Required parameter (Change and Create Namelist)	73
Required parameters (Copy Namelist)	73
Optional parameters	73
Change, Copy, and Create Process	76
Required parameters (Change and Create Process)	76
Required parameters (Copy Process)	76
Optional parameters	77
Change, Copy, and Create Queue	80
Required parameters (Change and Create Queue)	82
Required parameters (Copy Queue)	82
Required parameters (all commands)	83
Optional parameters	83
Error codes	99
Change Queue Manager	99
Optional parameters	100
Error codes	123
Change Security	124
Optional parameters	124
Change, Copy, and Create Service	125
Required parameter (Change and Create Service)	125
Required parameters (Copy Service)	125
Optional parameters	126
Change, Copy, and Create Storage Class	128
Required parameters (Change and Create Storage Class)	128
Required parameters (Copy Storage Class) . .	128
Optional parameters	129
Clear Queue.	131
Required parameters	131
Optional parameters	131
Error codes	132
Delete Authentication Information Object	132
Required parameters	132
Optional parameters	133
Delete Authority Record.	134
Required parameters	134
Optional parameters	135
Error codes	135
Delete CF Structure	135
Required parameters	136
Delete Channel	136
Required parameters	136
Optional parameters	136
Error codes	137
Delete Channel Listener	138
Required parameters	138
Delete Namelist	138
Required parameters	138
Optional parameters	139
Delete Process	140
Required parameters	140
Optional parameters	140
Delete Queue	141
Required parameters	141
Optional parameters	142
Error codes	143
Delete Service	144
Required parameters	144
Delete Storage Class	144
Required parameters	144
Optional parameters	144
Escape.	145
Required parameters	146
Error codes	146
Escape (Response).	146
Parameters	147
Inquire Archive.	147
Optional parameters	147
Inquire Archive (Response).	148
Response data - archive parameter information	149
Response data - tape unit status information	151
Inquire Authentication Information Object. .	152
Required parameters	152
Optional parameters	152
Inquire Authentication Information Object (Response)	154
Response data	154
Inquire Authentication Information Object Names	156
Required parameters	156
Optional parameters	156
Inquire Authentication Information Object Names (Response)	157
Response data	158
Inquire Authority Records	158
Required parameters	158
Optional parameters	160
Error codes	161
Inquire Authority Records (Response)	161
Response data	162
Inquire Authority Service	164
Required parameters	164
Optional parameters	164
Error codes	165
Inquire Authority Service (Response)	165
Response data	165
Inquire CF Structure	166
Required parameters	166
Optional parameters	166
Inquire CF Structure (Response)	167
Response data	167
Inquire CF Structure Names	168
Required parameters	169
Inquire CF Structure Names (Response) . . .	169
Response data	169
Inquire CF Structure Status.	169
Required parameters	170
Optional parameters	170
Inquire CF Structure Status (Response) . . .	171
Response data	171
Inquire Channel	174
Required parameters	174
Optional parameters	175

Error codes	180
Inquire Channel (Response)	181
Response data	181
Inquire Channel Initiator	189
Optional parameters	189
Inquire Channel Initiator (Response)	190
Response data - channel initiator information	190
Response data - listener information	191
Inquire Channel Listener	192
Required parameters	193
Optional parameters	193
Inquire Channel Listener (Response)	195
Response data	195
Inquire Channel Listener Status	197
Required parameters	197
Optional parameters	197
Error codes	199
Inquire Channel Listener Status (Response)	199
Response data	200
Inquire Channel Names	202
Required parameters	202
Optional parameters	202
Error codes	204
Inquire Channel Names (Response)	204
Response data	204
Inquire Channel Status	205
Required parameters	207
Optional parameters	207
Error codes	215
Inquire Channel Status (Response)	216
Response data	217
Inquire Cluster Queue Manager	226
Required parameters	226
Optional parameters	226
Inquire Cluster Queue Manager (Response)	231
Response data	231
Inquire Connection	239
Required parameters	239
Optional parameters	240
Error codes	243
Inquire Connection (Response)	243
Response data	244
Inquire Entity Authority	248
Required parameters	249
Optional parameters	250
Error codes	250
Inquire Entity Authority (Response)	251
Response data	251
Inquire Group	253
Inquire Group (Response)	253
Response data relating to the queue manager	254
Response data relating to obsolete DB2 messages	255
Inquire Log	255
Optional parameters	256
Inquire Log (Response)	256
Response data - log parameter information	257
Response data - to log status information	258
Inquire Namelist	259
Required parameters	260
Optional parameters	260
Inquire Namelist (Response)	262
Response data	262
Inquire Namelist Names	264
Required parameters	264
Optional parameters	264
Inquire Namelist Names (Response)	265
Response data	265
Inquire Process	266
Required parameters	266
Optional parameters	266
Inquire Process (Response)	268
Response data	268
Inquire Process Names	270
Required parameters	270
Optional parameters	270
Inquire Process Names (Response)	271
Response data	272
Inquire Queue	272
Required parameters	272
Optional parameters	272
Error codes	280
Inquire Queue (Response)	281
Response data	281
Inquire Queue Manager	291
Optional parameters	291
Inquire Queue Manager (Response)	299
Response data	300
Inquire Queue Manager Status	319
Optional parameters	319
Inquire Queue Manager Status (Response)	320
Response data	320
Inquire Queue Names	322
Required parameters	322
Optional parameters	322
Inquire Queue Names (Response)	324
Response data	324
Inquire Queue Status	325
Required parameters	325
Optional parameters	325
Error codes	330
Inquire Queue Status (Response)	330
Response data if StatusType is MQIACF_Q_STATUS	330
Response data if StatusType is MQIACF_Q_HANDLE	332
Inquire Security	336
Optional parameters	337
Inquire Security (Response)	337
Response data	338
Inquire Service	339
Required parameters	339
Optional parameters	339
Inquire Service (Response)	340
Response data	341
Inquire Service Status	342
Required parameters	342
Optional parameters	343
Error codes	344
Inquire Service Status (Response)	344
Response data	344
Inquire Storage Class	346

Required parameters	346
Optional parameters	347
Inquire Storage Class (Response)	349
Response data	349
Inquire Storage Class Names	350
Required parameters	350
Optional parameters	350
Inquire Storage Class Names (Response)	351
Response data	352
Inquire System	352
Optional parameters	352
Inquire System (Response)	353
Response data	353
Inquire Usage	357
Optional parameters	357
Inquire Usage (Response)	358
Response data if UsageType is MQIACF_USAGE_PAGESET	358
Response data if UsageType is MQIACF_USAGE_BUFFER_POOL	359
Response data if UsageType is MQIACF_USAGE_DATA_SET	360
Move Queue	360
Required parameters	360
Optional parameters	361
Ping Channel	362
Required parameters	363
Optional parameters	363
Error codes	364
Ping Queue Manager	366
Recover CF Structure	366
Required parameters	366
Optional parameters	366
Refresh Cluster	367
Required parameters	367
Optional parameters	368
Refresh Queue Manager	369
Required parameters	369
Optional parameters	369
Refresh Security	371
Optional parameters	371
Reset Channel	373
Required parameters	373
Optional parameters	374
Error codes	375
Reset Cluster	375
Required parameters	376
Optional parameters	376
Error codes	377
Reset Queue Manager	377
Required parameters	377
Error codes	378
Reset Queue Statistics	378
Required parameters	378
Optional parameters	379
Error codes	379
Reset Queue Statistics (Response)	379
Response data	379
Resolve Channel	380
Required parameters	381
Optional parameters	381
Error codes	383
Resume Queue Manager	383
Required parameters	383
Optional parameters	383
Resume Queue Manager Cluster	384
Required parameters	384
Optional parameters	384
Error codes	385
Reverify Security	385
Required parameters	385
Optional parameters	385
Set Archive	386
Required parameters	386
Optional parameters	386
Set Authority Record	390
Required parameters	390
Optional parameters	391
Error codes	393
Set Log	394
Required parameters	394
Optional parameters	394
Set System	396
Required parameters	396
Optional parameters	396
Start Channel	397
Required parameters	398
Optional parameters	398
Error codes	400
Start Channel Initiator	401
Required parameters	401
Optional parameters	401
Error codes	402
Start Channel Listener	402
Optional parameters	403
Error codes	404
Start Service	404
Required parameters	405
Error codes	405
Stop Channel	405
Required parameters	406
Optional parameters	406
Error codes	408
Stop Channel Initiator	409
Optional parameters	409
Stop Channel Listener	410
Required parameters	410
Optional parameters	410
Error codes	411
Stop Connection	412
Required parameters	412
Stop Service	412
Required parameters	412
Error codes	413
Suspend Queue Manager	413
Required parameters	413
Optional parameters	414
Suspend Queue Manager Cluster	414
Required parameters	414
Optional parameters	415
Error codes	415

Chapter 4. Structures for commands and responses	417
How the structures are shown.	417
Data types	417
Initial values and default structures	417
Usage notes	418
MQCFH - PCF header	418
Fields	418
Language declarations	420
MQCFBF - PCF byte string filter parameter	422
Fields	422
Language declarations	423
MQCFBS - PCF byte string parameter	425
Fields	425
Language declarations	426
MQCFIF - PCF integer filter parameter	427
Fields	427
Language declarations	429
MQCFIL - PCF integer list parameter	430
Fields	430
Language declarations	431
MQCFIN - PCF integer parameter	433
Fields	433
Language declarations	433
MQCFSF - PCF string filter parameter	434
Fields	435
Language declarations	437
MQCFSL - PCF string list parameter	439
Fields	439
Language declarations	441
MQCFST - PCF string parameter	442
Fields	443
Language declarations	444
Chapter 5. PCF example	447
Inquire local queue attributes	447
Program listing.	447
Part 2. Message Queueing Administration Interface	459
Chapter 6. Introduction to the WebSphere MQ Administration Interface (MQAI)	463
MQAI concepts and terminology	463
Use of the MQAI	464
How do I use the MQAI?	465
Overview.	465
Building your MQAI application	466
Chapter 7. Using data bags	467
Types of data bag	467
Creating and deleting data bags	467
Deleting data bags.	468
Types of data item.	468
Adding data items to bags	469
Adding an inquiry command to a bag	469
Changing information within a bag	470
Counting data items	471
Deleting data items	472
Deleting data items from a bag using the mqDeleteItem call	472
Clearing a bag using the mqClearBag call	472
Truncating a bag using the mqTruncateBag call	473
Inquiring within data bags	473
System items	473
Chapter 8. Configuring WebSphere MQ using mqExecute	475
Sending administration commands to the command server	475
Example code	476
Hints and tips for configuring WebSphere MQ	477
Chapter 9. Exchanging data between applications	479
Converting bags and buffers	479
Putting and receiving data bags	480
Sending PCF messages to a specified queue	480
Receiving PCF messages from a specified queue	480
Chapter 10. MQAI reference	483
mqAddBag	484
Syntax.	484
Parameters	484
Usage notes	485
C language invocation	485
Visual Basic invocation	485
mqAddByteString	485
Syntax.	485
Parameters	486
Usage notes	487
C language invocation	487
Visual Basic invocation	487
mqAddByteStringFilter	487
Syntax.	487
Parameters	487
Usage notes	489
C language invocation	489
Visual Basic invocation	489
mqAddInquiry	489
Syntax.	489
Parameters	490
Usage notes	490
C language invocation	490
Visual Basic invocation	491
Supported INQUIRE command codes	491
mqAddInteger	491
Syntax.	491
Parameters	491
Usage notes	492
C language invocation	492
Visual Basic invocation	493
mqAddInteger64	493
Syntax.	493
Parameters	493
Usage notes	494
C language invocation	494
Visual Basic invocation	494

mqAddIntegerFilter	494	Syntax.	514
Syntax.	494	Parameters	514
Parameters	495	Usage notes	517
Usage notes	496	C language invocation	517
C language invocation	496	Visual Basic invocation	518
Visual Basic invocation	496	mqGetBag	518
mqAddString	496	Syntax.	518
Syntax.	496	Parameters	518
Parameters	496	Usage notes	520
Usage notes	498	C language invocation	520
C language invocation	498	Visual Basic invocation	520
Visual Basic invocation	498	mqInquireBag	520
mqAddStringFilter	498	Syntax.	520
Syntax.	498	Parameters	520
Parameters	498	C language invocation	522
Usage notes	500	Visual Basic invocation	522
C language invocation	500	mqInquireByteString	523
Visual Basic invocation	500	Syntax.	523
mqBagToBuffer	500	Parameters	523
Syntax.	500	C language invocation	525
Parameters	501	Visual Basic invocation	525
Usage notes	502	mqInquireByteStringFilter	525
C language invocation	502	Syntax.	525
Visual Basic invocation	502	Parameters	526
mqBufferToBag	503	C language invocation	528
Syntax.	503	Visual Basic invocation	528
Parameters	503	mqInquireInteger	528
Usage notes	504	Syntax.	528
C language invocation	504	Parameters	529
Visual Basic invocation	504	C language invocation	530
mqClearBag	504	Visual Basic invocation	530
Syntax.	504	mqInquireInteger64	531
Parameters	504	Syntax.	531
Usage notes	505	Parameters	531
C language invocation	505	C language invocation	532
Visual Basic invocation	505	Visual Basic invocation	533
mqCountItems	505	mqInquireIntegerFilter	533
Syntax.	505	Syntax.	533
Parameters	505	Parameters	533
Usage notes	506	C language invocation	535
C language invocation	506	Visual Basic invocation	535
Visual Basic invocation	506	mqInquireItemInfo	535
mqCreateBag	507	Syntax.	535
Syntax.	507	Parameters	535
Parameters	507	C language invocation	537
Usage notes	510	Visual Basic invocation	538
C language invocation	510	mqInquireString	538
Visual Basic invocation	510	Syntax.	538
mqDeleteBag	511	Parameters	538
Syntax.	511	C language invocation	540
Parameters	511	Visual Basic invocation	541
Usage notes	511	mqInquireStringFilter	541
C language invocation	511	Syntax.	541
Visual Basic invocation	511	Parameters	541
mqDeleteItem	512	C language invocation	543
Syntax.	512	Visual Basic invocation	544
Parameters	512	mqPad	544
Usage notes	513	Syntax.	544
C language invocation	514	Parameters	544
Visual Basic invocation	514	Usage notes	545
mqExecute	514	C language invocation	545

mqPutBag	545	C language invocation	565
Syntax	545	Visual Basic invocation	566
Parameters	545	mqTrim	566
C language invocation	547	Syntax	566
Visual Basic invocation	547	Parameters	566
mqSetByteString	547	Usage notes	567
Syntax	547	C language invocation	567
Parameters	547	mqTruncateBag	567
C language invocation	549	Syntax	567
Visual Basic invocation	549	Parameters	567
mqSetByteStringFilter	550	Usage notes	568
Syntax	550	C language invocation	568
Parameters	550	Visual Basic invocation	568
C language invocation	552	MQAI Selectors	568
Visual Basic invocation	552	User selectors	569
mqSetInteger	553	System selectors	569
Syntax	553		
Parameters	553		
C language invocation	555		
Visual Basic invocation	555		
mqSetInteger64	555		
Syntax	555	Chapter 11. Examples of using the MQAI	571
Parameters	555	Creating a local queue (amqsaicq.c)	571
C language invocation	557	Inquiring about queues and printing information (amqsailq.c)	575
Visual Basic invocation	557	Displaying events using an event monitor (amqsaiem.c)	580
mqSetIntegerFilter	557		
Syntax	557		
Parameters	558	Chapter 12. Advanced topics	589
C language invocation	559	Indexing	589
Visual Basic invocation	560	Data conversion	590
mqSetString	560	Use of the message descriptor	591
Syntax	560		
Parameters	560		
Usage notes	562	Part 3. Appendixes	593
C language invocation	562		
Visual Basic invocation	563	Notices	595
mqSetStringFilter	563	Trademarks	596
Syntax	563		
Parameters	563	Index	599
Usage notes	565		

Figures

1. Hierarchy of MQAI concepts	464	9. Nesting	475
2. How the MQAI administers WebSphere MQ	465	10. Using mqExecute to create a local queue	476
3. Adding data items	469	11. Using mqExecute to inquire about queue	
4. Modifying a single data item	470	attributes	477
5. Modifying all data items	471	12. Converting bags to PCF messages.	479
6. Deleting a single data item	472	13. Converting PCF messages to bag form	479
7. Deleting all data items	472	14. Indexing	589
8. Truncating a bag	473		

Tables

1. Windows,HP OpenVMS Alpha, NP NonStop Server, and UNIX systems - object authorities	20	11. ChannelDisposition and CommandScope for RESET CHANNEL.	374
2. MQIACF_COMMAND_INFO values	26	12. ChannelDisposition and CommandScope for RESOLVE CHANNEL	382
3. Change, Copy, Create Channel parameters	41	13. ChannelDisposition and CommandScope for START CHANNEL.	399
4. CipherSpecs that can be used with WebSphere MQ SSL support	63	14. ChannelDisposition and CommandScope for STOP CHANNEL	406
5. Change, Copy, Create Queue parameters	81	15. CCSID processing	590
6. ChannelDisposition and CommandScope for Inquire Channel Status, Current	207	16. PCF command type	591
7. ChannelDisposition and CommandScope for Inquire Channel Status, Short	208	17. Format and MsgType parameters of the MQMD	591
8. ChannelDisposition and CommandScope for Inquire Channel Status, Saved	208	18. Message descriptor values	592
9. Inquire Queue command, queue attributes	274		
10. ChannelDisposition and CommandScope for PING CHANNEL	364		

About this book

This book applies to the following:

- WebSphere MQ for AIX
- WebSphere MQ for HP-UX
- WebSphere MQ for iSeries
- WebSphere MQ for Linux
- WebSphere MQ for Solaris
- WebSphere MQ for Windows
- WebSphere MQ for z/OS

Unless otherwise stated, the information also applies to the following:

- MQSeries for Compaq NonStop Kernel, V5.1
- WebSphere MQ for HP OpenVMS, V5.3

The first section of this book describes the facilities available on WebSphere® MQ products for writing programs using the WebSphere MQ Programmable Command Formats (PCFs) to administer IBM® WebSphere MQ systems either locally or remotely.

The second section of this book describes the administration interface for WebSphere MQ. This part of the product is referred to as the *WebSphere MQ Administration Interface (MQAI)*.

The MQAI is a programming interface that simplifies the use of PCF messages to configure WebSphere MQ.

The term UNIX® systems is used to denote the following UNIX operating systems, unless otherwise stated:

- AIX®
- HP-UX
- Linux (POWER™, zSeries®, and X86 platforms)
- Solaris

The term Windows® is used throughout this book to denote the following Windows operating systems, unless stated otherwise:

- Windows 2000
- Windows XP
- Windows 2003

Who this book is for

Primarily, this book is for system programmers who write programs to monitor and administer WebSphere MQ products.

What you need to know to understand this book

For the first section of this book (PCFs) you need:

- Experience in writing systems management applications
- An understanding of the Message Queue Interface (MQI)
- Experience of WebSphere MQ programs in general, or familiarity with the content of the other books in the WebSphere MQ library.

For the second section of this book (MQAI) you need:

- Some knowledge of WebSphere MQ
- Knowledge of how to write programs in the C programming language or in Visual Basic for Windows.

How to use this book

The first part of this book describes PCFs.

PCFs are the formats of command and response messages that are sent between a WebSphere MQ systems management application, or other program, and a WebSphere MQ queue manager.

Chapter 1, “Introduction to Programmable Command Formats,” on page 7 and Chapter 2, “Using Programmable Command Formats,” on page 11 contain introduction and guidance material. You are advised to read both of these chapters.

The reference material starts in Chapter 3, “Definitions of the Programmable Command Formats,” on page 25. See Chapter 5, “PCF example,” on page 447 for an example of how PCFs can be used.

The second part of this book describes the MQAI.

The first four chapters introduce the Message Queuing Administration Interface and tell you how to use it.

Chapter 10, “MQAI reference,” on page 483 contains the reference information.

Chapter 11, “Examples of using the MQAI,” on page 571 provides some example programs.

Chapter 12, “Advanced topics,” on page 589 describes indexing, data conversion, and the message descriptor.

Appendices

The MQAI user and system selectors are given in “MQAI Selectors” on page 568

Summary of changes

This section describes changes in this edition of *WebSphere MQ Programmable Command Formats and Administration Interface*. Changes since the previous edition of the book are highlighted with a vertical bar to the left of the canged text, as shown on this paragraph.

Changes for this edition (SC34-6598-01)

The changes for this edition of the *WebSphere MQ Programmable Command Formats and Administration Interface* manual include:

- Various minor corrections and changes.

Changes for the previous edition (SC34-6598-00)

The changes for this edition of the *WebSphere MQ Programmable Command Formats and Administration Interface* manual include:

Changes to PCF commands and responses

- You can now use PCFs to administer WebSphere MQ objects on z/OS. Most PCF commands have been extended so that they are available on z/OS. A number of new commands that are specific to z/OS are now available in PCF. Information about the extended response types introduced as a result of PCF being made available on z/OS is also provided in <xref refid="useres">.
- A number of commands are introduced, on platforms other than z/OS, to control certain queue manager service tasks such as channel listeners.
- A number of commands are introduced, on platforms other than z/OS, to permit the administration of object authorities.
- On platforms other than z/OS, a command, Inquire Authority Service, and response, Inquire Authority Service (Response) are introduced to return information about the level of function supported by installed authority managers
- A command, Inquire Connection, and response, Inquire Connection (Response) are introduced to return connection information about the applications connected to the queue manager. This enables you to identify applications with long-running units of work.
- A command, Stop Connection, is introduced to break a connection between an application and the queue manager.
- On platforms other than z/OS, a command, Inquire Queue Manager Status, and response, Inquire Queue Manager (Response) are introduced to display status information about the queue manager
- On a number of Inquire commands, you can now specify a filter condition to return information only about those objects which satisfy the filter condition. The PCF structures MQCFBF, MQCFIF, MQCFSF are introduced to facilitate this.
- On the Inquire Queue Manager command, you can now choose to return different sets of queue manager parameters. The parameters are MQIACF_Q_MGR_SYSTEM, MQIACF_Q_MGR_EVENT, MQIACF_Q_MGR_DQM, and MQIACF_Q_MGR_CLUSTER. They permit the return of the following groups of parameters:

- MQIACF_Q_MGR_SYSTEM - displays queue manager system parameters
 - MQIACF_Q_MGR_EVENT - displays the set of parameters that are related to event control
 - MQIACF_Q_MGR_DQM - displays the set of parameters relating to distributed queuing
 - MQIACF_Q_MGR_CLUSTER - displays the set of parameters relating to clustering
- System queue, SYSTEM.ADMIN.TRACE.ROUTE.QUEUE is now available to which trace-route records are written. A queue manager parameter, **TraceRouteRecording**, is introduced to control whether such messages are generated, and where they are written. The Change Queue Manager, Inquire Queue Manager commands, and Inquire Queue Manager (Response) have been updated.
- System queue, SYSTEM.ADMIN.ACTIVITY.QUEUE is now available to which activity reports are written. A new queue manager parameter, **ActivityRecording**, is introduced to control whether such messages are generated, and where they are written. The Change Queue Manager, Inquire Queue Manager commands, and Inquire Queue Manager (Response) have been updated.
- Parameters are introduced to improve cluster workload management. These are:
 - **CLWLMRUCChannels** - the maximum number of outbound cluster channels
 - **CLWLUseQ-** specifies the behavior of an MQPUT operation when the target queue has a local instance and at least one remote cluster instance

on the queue manager definition,

 - **CLWLChannelPriority** - the priority of the channel
 - **CLWLChannelRank** - the rank of the channel
 - **CLWLChannelWeight** - the weighting to be applied to the channel

on the channel definition, and

 - **CLWLQueuePriority** - the priority of the queue
 - **CLWLQueueRank** - the rank of the queue

on the queue definition.

The Change, Copy and Create Channel, Change, Copy, and Create Queue, Change Queue Manager, Inquire Channel, Inquire Queue, Inquire Queue Manager commands and responses Inquire Channel (Response), Inquire Queue (Response), Inquire Queue Manager (Response) have been updated.
- Accounting data can now be collected for local and model queues. A queue manager parameter is introduced, **QueueAccounting**, to control the collection of accounting data for queues. The Change Queue Manager and Inquire Queue Manager commands, and Inquire Queue Manager (Response) have been updated.
- A queue parameter is introduced, **QueueAccounting**, to control the collection of accounting data for queues on an individual queue basis. The Change, Copy, Create Queue and Inquire Queue Manager commands, and Inquire Queue (Response) have been updated.
- Monitoring data can now be collected and displayed for channels and queues. The following queue manager parameters are introduced:
 - **ChannelMonitoring** - controls the collection of online monitoring data for channels
 - **ClusterSenderMonitoringDefault** - controls the collection of online monitoring data for auto-defined cluster-sender channels

- **QueueMonitoring** - controls the collection of online monitoring data for queues

The Change Queue Manager and Inquire Queue Manager commands, and Inquire Queue Manager (Response) have been updated.

The channel parameter, **ChannelMonitoring**, is introduced to specify the level of online monitoring data collection for channels on an individual channel basis.

The Change, Copy, and Create Channel and Inquire Channel commands, and Inquire Channel (Response) have been updated.

The queue parameter, **QueueMonitoring**, is introduced to specify the level of online monitoring data collection for queues on an individual queue basis. The Change, Copy, and Create Queue and Inquire Queue commands, and Inquire Queue (Response) have been updated.

A number of parameters are introduced on the Inquire Channel Status command and Inquire Channel Status (Response) to return information relating to message activity on the channel. These are:

- **BatchSizeIndicator** - the size of the batches transmitted over the channel
- **ChannelMonitoring** - the current level of monitoring data collection for the channel
- **ExitTime** - the time, displayed in microseconds, spent processing user exits per message
- **MessagesAvailable** - the number of messages queued on the transmission queue available to the channel for MQGETs
- **NetTime** - the time to send a request to the remote end of the channel and receive a response
- **Substate** - the action currently being performed by the channel
- **XQTime** - the time that messages remained on the transmission queue before being retrieved

You can specify any of these individually or you can select them all by using the MQIACF_MONITORING parameter on the command.

A number of parameters are introduced on the Inquire Queue Status command and Inquire Queue Status (Response) to return information relating to message activity on the queue. These are

- **LastGetDate** - the date on which the last message was retrieved from the queue since the queue manager started
- **LastgetTime** - the time at which the last message was retrieved from the queue since the queue manager started
- **LastPutDate** - the date on which the last message was put to the queue since the queue manager started
- **LastPutTime** - the time at which the last message was put to the queue since the queue manager started
- **QueueMonitoring** - the current level of monitoring data collection for the queue
- **OldestMsgAge** - the age, in seconds, of the oldest message on the queue
- **OnQTime** - the interval between messages being put on the queue and then being destructively read

You can specify any of these individually or you can select them all by using the MQIACF_MONITORING parameter on the command.

- On platforms other than z/OS, accounting statistics data can now be collected and displayed for channels and queues. The following queue manager parameters are introduced:

- **AccountingConnOverride** - whether applications can override the settings of the ACCTQ and ACCTMQI queue manager parameters
- **AccountingInterval** - the time interval at which intermediate accounting records are written
- **ChannelStatistics** - whether statistics data is to be collected for channels
- **ClusterSenderStatistics** - whether statistics data is to be collected for auto-defined cluster-sender channels
- **MQIAccounting** - whether accounting information for MQI data is to be collected
- **MQIStatistics** - whether MQI statistics monitoring data is to be collected for the queue manager
- **QueueStatistics** - whether statistics data is to be collected for queues
- **StatisticsInterval** - the time interval at which statistics monitoring data is written to the monitoring queue

The Change Queue Manager and Inquire Queue Manager commands, and Inquire Queue Manager (Response) have been updated.

The channel parameter, **ChannelStatistics**, is introduced to specify the level of online statistics data collection for channels on an individual channel basis. The Change, Copy, and Create Channel and Inquire Channel commands, and Inquire Channel (Response) have been updated.

The queue parameter, **QueueStatistics**, is introduced to specify the level of online statistics data collection for queues on an individual queue basis. The Change, Copy, and Create Queue and Inquire Queue commands, and Inquire Queue (Response) have been updated.

- Queue manager parameter, **ChannelEvent**, is introduced to control the generation of channel events. The Change Queue Manager and Inquire Queue Manager commands, and Inquire Queue Manager (Response) have been updated.
- Queue manager parameter, **IPaddressVersion**, is introduced to allow you to specify which IP protocol to use for a channel connection; IPv4 or IPv6. The Change Queue Manager and Inquire Queue Manager commands, and Inquire Queue Manager (Response) have been updated.
- On platforms other than z/OS, a queue manager parameter, **LoggerEvent**, is introduced to specify whether recovery log events are generated. The Change Queue Manager and Inquire Queue Manager commands, and Inquire Queue Manager (Response) have been updated. On the Inquire Queue Status command and Inquire Queue Status (Response), parameter **MediaRecoveryLogExtent** is introduced to display the log extent or the journal receiver needed for media recovery of the queue.
- Queue manager parameter, **SSLEvent**, is introduced to control the generation of SSL events. The Change Queue Manager and Inquire Queue Manager commands, and Inquire Queue Manager (Response) have been updated.
- On UNIX and Windows platforms, a queue manager parameter, **SSLFIPSRequired**, is introduced to specify whether only FIPS-certified algorithms are to be used if cryptography is carried out. The Change Queue Manager and Inquire Queue Manager commands, and Inquire Queue Manager (Response) have been updated. The 'CipherSpecs' table in the Change, Copy, and Create Channel command description has been updated to reflect this function.
- Queue manager parameter, **SSLKeyReset**, is introduced to specify the number of unencrypted bytes sent and received within an SSL conversation before the

secret key is renegotiated. The Change Queue Manager and Inquire Queue Manager commands, and Inquire Queue Manager (Response) have been updated.

- Compression of channel data is now possible. This reduces the amount of network traffic and can therefore improve the performance of channels. Parameters **HeaderCompression** and **MessageCompression** are introduced on the channel definition to enable this. The Change, Copy, and Create Channel, Inquire Channel, Inquire Cluster Queue Manager, Display Channel Status commands and Inquire Channel (Response), Inquire Cluster Queue Manager (Response) and Inquire Channel Status (Response) have been updated. Parameters **CompressionRate** and **CompressionTime** are introduced on the Inquire Channel Status command and Inquire Channel Status (Response) to display the compression rate achieved and the amount of time per message spent during compression.
- The message retry parameters **MessageRetryUserData**, **MessageRetryExit**, **MessageRetryCount**, and **MessageRetryInterval** have been extended to include z/OS. This means that transient PUT failures in the message channel agent can be retried on this platform. The Change, Copy, and Create Channel and Inquire Channel commands and Inquire Channel (Response) have been updated.
- The **HeartbeatInterval** parameter has been extended to include z/OS on server-connection and client-connection channels meaning that the server message channel agent can handle situations where the client connection fails during an MQGET with WAIT on this platform. The Change, Copy, and Create Channel and Inquire Channel commands and Inquire Channel (Response) have been updated.
- Parameters **SSLKeyResetDate**, **SSLKeyResetTime**, and **SSLKeyResets** are introduced to display the date and time of the previous successful key reset, and the number of successful key resets since channel start. The Inquire Channel Status command and Inquire Channel Status (Response) have been updated.
- The following parameters are also now available on the Inquire Channel Status command and Inquire Channel Status (Response):
 - **KeepAliveInterval** - the KeepAlive interval (on z/OS only)
 - **MCAUserIdentity** - the message channel agent user identifier
 - **RemoteApplTag** - the remote partner application
 - **SSLCertRemoteIssuerName** - the full Distinguished name of the issuer of a remote certificate
 - **SSLCertUserId** - the local user identifier associated with a remote certificate (on z/OS only)
- On platforms other than z/OS, you can now set the reliability to be assigned to non-persistent messages put to local and model queues by using the **NonPersistentMessageClass** parameter. The Change, Copy, and Create Queue and Inquire Queue commands, and Inquire Queue (Response) have been updated
- The Refresh Security command now has a parameter, **Securitytype**, to specify which type of refresh is to be performed.
- UDP support is withdrawn.

Changes to the MQAI

- The following calls are introduced:
 - mqAddBag to permit the nesting of a bag within another bag

- mqAddByteString to add a byte string identified by a user selector to the end of a specified bag.
- mqAddByteStringFilter to add a byte string filer identified by a user selector to the end of a specified bag.
- mqAddInteger64 to add a 64-bit integer item identified by a user selector to the end of a specified bag.
- mqAddIntegerFilter to add an integer filter identified by a user selector to the end of a specified bag.
- mqAddStringFilter to add a string filter identified by a user selector to the end of a specified bag.
- mqInquireByteString to request the value of a byte string data item present in the bag.
- mqInquireByteStringFilter to request the value and operator of a byte string filter item present in the bag.
- mqInquireInteger64 to request the value of a 64-bit integer data item present in the bag.
- mqInquireIntegerFilter to request the value and operator of an integer filter item present in the bag.
- mqInquireStringFilter to request the value and operator of a string filter item present in the bag.
- mqSetByteString to either modify a byte string data item that is already present in the bag, or delete all existing occurrences of the specified selector and add a new occurrence at the end of the bag.
- mqSetByteStringFilter to either modify a byte string filter item that is already present in the bag, or delete all existing occurrences of the specified selector and add a new occurrence at the end of the bag.
- mqSetInteger64 to either modify a 64-bit integer item that is already present in the bag, or delete all existing occurrences of the specified selector and add a new occurrence at the end of the bag.
- mqSetIntegerFilter to either modify an integer filter item that is already present in the bag, or delete all existing occurrences of the specified selector and add a new occurrence at the end of the bag.
- mqSetStringFilter to either modify a string filter item that is already present in the bag, or delete all existing occurrences of the specified selector and add a new occurrence at the end of the bag.

Changes to make this manual easier to use

- All parameters within commands and responses are described in alphabetical order.
- All commands and responses have a table at the start of their description showing for which WebSphere MQ products they are valid.
-
- The MQRCCF_* reason codes (they used to be listed in the Appendix) are now in the *WebSphere MQ Messages* and *WebSphere MQ for z/OS Messages and Codes* manuals.
- All constants (they used to be listed in the Appendix) are now in the *WebSphere MQ Constants* manual.
- MQAI selector reference information (it used to be listed in the Appendix) is now in “MQAI Selectors” on page 568.

Part 1. Programmable Command Formats

Chapter 1. Introduction to Programmable Command Formats	7
The problem PCF commands solve	7
What PCFs are	8
Other administration interfaces	8
WebSphere MQ for iSeries	8
i5/OS Control Language (CL)	8
WebSphere MQ Commands (MQSC)	8
WebSphere MQ for z/OS	9
MQSeries for Compaq NonStop Kernel, V5.1	9
WebSphere MQ for Windows, UNIX systems and HP OpenVMS	9
WebSphere MQ commands (MQSC)	9
Control commands	9
WebSphere MQ Explorer - WebSphere MQ for Windows and WebSphere MQ for Linux (x86 platform)	9
The WebSphere MQ Administration Interface (MQAI)	9

Chapter 2. Using Programmable Command Formats	11
PCF command messages	11
How to issue PCF command messages	11
Message descriptor for a PCF command	11
Sending user data	13
Responses	13
Message descriptor for a response	13
Standard responses	14
OK response	14
Error response	14
Data response	15
Extended responses	15
Extended responses to Inquire commands	16
Extended responses to commands other than Inquire	16
Extended responses to commands using CommandScope	17
Rules for naming WebSphere MQ objects	17
Name lengths	18
Reserved object names	18
Generic values	18
Authority checking for PCF commands	18
WebSphere MQ for iSeries	18
WebSphere MQ for Windows, and UNIX systems	19
WebSphere MQ for HP OpenVMS and Compaq NonStop Kernel	20
WebSphere MQ for z/OS	23

Chapter 3. Definitions of the Programmable Command Formats	25
How the definitions are shown	25
Commands	25
Responses	25
Parameters and response data	26
Constants	26
Informational messages	26

Error codes	27
Error codes applicable to all commands	27
PCF commands and responses in groups	30
Authentication Information commands	30
Authority Record commands	30
CF commands	30
Channel commands	31
Cluster commands	31
Connection commands	31
Escape command	31
Namelist commands	31
Process commands	32
Queue commands	32
Queue Manager commands	32
Security commands	32
Service commands	32
Storage class commands	32
System commands	33
Data responses to commands	33
Definitions of Programmable Command Formats	34
Backup CF Structure	34
Required parameters	34
Optional parameters	34
Change, Copy, and Create Authentication Information Object	35
Required parameters (Change authentication information)	35
Required parameters (Copy authentication information)	35
Required parameters (Create authentication information)	36
Optional parameters	36
Change, Copy, and Create CF Structure	38
Required parameters (Change and Create CF Structure)	39
Required parameters (Copy CF Structure)	39
Optional parameters	39
Change, Copy and Create Channel	41
Required parameters (Change, Create Channel)	43
Required parameters (Copy Channel)	43
Optional parameters	44
Error codes	67
Change, Copy, and Create Channel Listener	70
Required parameters (Change and Create Channel Listener)	70
Required parameters (Copy Channel Listener)	70
Optional parameters	71
Change, Copy, and Create Namelist	72
Required parameter (Change and Create Namelist)	73
Required parameters (Copy Namelist)	73
Optional parameters	73
Change, Copy, and Create Process	76
Required parameters (Change and Create Process)	76
Required parameters (Copy Process)	76

Optional parameters	77
Change, Copy, and Create Queue	80
Required parameters (Change and Create Queue) .	82
Required parameters (Copy Queue)	82
Required parameters (all commands)	83
Optional parameters	83
Error codes	99
Change Queue Manager	99
Optional parameters	100
Error codes	123
Change Security	124
Optional parameters	124
Change, Copy, and Create Service	125
Required parameter (Change and Create Service)	125
Required parameters (Copy Service)	125
Optional parameters	126
Change, Copy, and Create Storage Class	128
Required parameters (Change and Create Storage Class)	128
Required parameters (Copy Storage Class)	128
Optional parameters	129
Clear Queue.	131
Required parameters	131
Optional parameters	131
Error codes	132
Delete Authentication Information Object	132
Required parameters	132
Optional parameters	133
Delete Authority Record.	134
Required parameters	134
Optional parameters	135
Error codes	135
Delete CF Structure	135
Required parameters	136
Delete Channel	136
Required parameters	136
Optional parameters	136
Error codes	137
Delete Channel Listener	138
Required parameters	138
Delete Namelist	138
Required parameters	138
Optional parameters	139
Delete Process	140
Required parameters	140
Optional parameters	140
Delete Queue	141
Required parameters	141
Optional parameters	142
Error codes	143
Delete Service	144
Required parameters	144
Delete Storage Class	144
Required parameters	144
Optional parameters	144
Escape.	145
Required parameters	146
Error codes	146
Escape (Response).	146
Parameters	147
Inquire Archive.	147
Optional parameters	147
Inquire Archive (Response).	148
Response data - archive parameter information .	149
Response data - tape unit status information .	151
Inquire Authentication Information Object.	152
Required parameters	152
Optional parameters	152
Inquire Authentication Information Object (Response)	154
Response data	154
Inquire Authentication Information Object Names	156
Required parameters	156
Optional parameters	156
Inquire Authentication Information Object Names (Response)	157
Response data	158
Inquire Authority Records	158
Required parameters	158
Optional parameters	160
Error codes	161
Inquire Authority Records (Response)	161
Response data	162
Inquire Authority Service	164
Required parameters	164
Optional parameters	164
Error codes	165
Inquire Authority Service (Response)	165
Response data	165
Inquire CF Structure	166
Required parameters	166
Optional parameters	166
Inquire CF Structure (Response)	167
Response data	167
Inquire CF Structure Names	168
Required parameters	169
Inquire CF Structure Names (Response)	169
Response data	169
Inquire CF Structure Status.	169
Required parameters	170
Optional parameters	170
Inquire CF Structure Status (Response)	171
Response data	171
Inquire Channel	174
Required parameters	174
Optional parameters	175
Error codes	180
Inquire Channel (Response)	181
Response data	181
Inquire Channel Initiator	189
Optional parameters	189
Inquire Channel Initiator (Response)	190
Response data - channel initiator information .	190
Response data - listener information.	191
Inquire Channel Listener	192
Required parameters	193
Optional parameters	193
Inquire Channel Listener (Response)	195
Response data	195
Inquire Channel Listener Status	197
Required parameters	197

Optional parameters	197
Error codes	199
Inquire Channel Listener Status (Response)	199
Response data	200
Inquire Channel Names	202
Required parameters	202
Optional parameters	202
Error codes	204
Inquire Channel Names (Response)	204
Response data	204
Inquire Channel Status	205
Required parameters	207
Optional parameters	207
Error codes	215
Inquire Channel Status (Response)	216
Response data	217
Inquire Cluster Queue Manager	226
Required parameters	226
Optional parameters	226
Inquire Cluster Queue Manager (Response)	231
Response data	231
Inquire Connection	239
Required parameters	239
Optional parameters	240
Error codes	243
Inquire Connection (Response)	243
Response data	244
Inquire Entity Authority	248
Required parameters	249
Optional parameters	250
Error codes	250
Inquire Entity Authority (Response)	251
Response data	251
Inquire Group	253
Inquire Group (Response)	253
Response data relating to the queue manager	254
Response data relating to obsolete DB2 messages	255
Inquire Log	255
Optional parameters	256
Inquire Log (Response)	256
Response data - log parameter information	257
Response data - to log status information	258
Inquire Namelist	259
Required parameters	260
Optional parameters	260
Inquire Namelist (Response)	262
Response data	262
Inquire Namelist Names	264
Required parameters	264
Optional parameters	264
Inquire Namelist Names (Response)	265
Response data	265
Inquire Process	266
Required parameters	266
Optional parameters	266
Inquire Process (Response)	268
Response data	268
Inquire Process Names	270
Required parameters	270
Optional parameters	270
Inquire Process Names (Response)	271
Response data	272
Inquire Queue	272
Required parameters	272
Optional parameters	272
Error codes	280
Inquire Queue (Response)	281
Response data	281
Inquire Queue Manager	291
Optional parameters	291
Inquire Queue Manager (Response)	299
Response data	300
Inquire Queue Manager Status	319
Optional parameters	319
Inquire Queue Manager Status (Response)	320
Response data	320
Inquire Queue Names	322
Required parameters	322
Optional parameters	322
Inquire Queue Names (Response)	324
Response data	324
Inquire Queue Status	325
Required parameters	325
Optional parameters	325
Error codes	330
Inquire Queue Status (Response)	330
Response data if StatusType is MQIACF_Q_STATUS	330
Response data if StatusType is MQIACF_Q_HANDLE	332
Inquire Security	336
Optional parameters	337
Inquire Security (Response)	337
Response data	338
Inquire Service	339
Required parameters	339
Optional parameters	339
Inquire Service (Response)	340
Response data	341
Inquire Service Status	342
Required parameters	342
Optional parameters	343
Error codes	344
Inquire Service Status (Response)	344
Response data	344
Inquire Storage Class	346
Required parameters	346
Optional parameters	347
Inquire Storage Class (Response)	349
Response data	349
Inquire Storage Class Names	350
Required parameters	350
Optional parameters	350
Inquire Storage Class Names (Response)	351
Response data	352
Inquire System	352
Optional parameters	352
Inquire System (Response)	353
Response data	353
Inquire Usage	357
Optional parameters	357

Inquire Usage (Response)	358
Response data if UsageType is	
MQIACF_USAGE_PAGESET	358
Response data if UsageType is	
MQIACF_USAGE_BUFFER_POOL	359
Response data if UsageType is	
MQIACF_USAGE_DATA_SET	360
Move Queue	360
Required parameters	360
Optional parameters	361
Ping Channel	362
Required parameters	363
Optional parameters	363
Error codes	364
Ping Queue Manager.	366
Recover CF Structure.	366
Required parameters	366
Optional parameters	366
Refresh Cluster	367
Required parameters	367
Optional parameters	368
Refresh Queue Manager.	369
Required parameters	369
Optional parameters	369
Refresh Security	371
Optional parameters	371
Reset Channel	373
Required parameters	373
Optional parameters	374
Error codes	375
Reset Cluster	375
Required parameters	376
Optional parameters	376
Error codes	377
Reset Queue Manager	377
Required parameters	377
Error codes	378
Reset Queue Statistics	378
Required parameters	378
Optional parameters	379
Error codes	379
Reset Queue Statistics (Response).	379
Response data	379
Resolve Channel	380
Required parameters	381
Optional parameters	381
Error codes	383
Resume Queue Manager	383
Required parameters	383
Optional parameters	383
Resume Queue Manager Cluster	384
Required parameters	384
Optional parameters	384
Error codes	385
Reverify Security	385
Required parameters	385
Optional parameters	385
Set Archive	386
Required parameters	386
Optional parameters	386
Set Authority Record	390
Required parameters	390
Optional parameters	391
Error codes	393
Set Log	394
Required parameters	394
Optional parameters	394
Set System	396
Required parameters	396
Optional parameters	396
Start Channel	397
Required parameters	398
Optional parameters	398
Error codes	400
Start Channel Initiator	401
Required parameters	401
Optional parameters	401
Error codes	402
Start Channel Listener	402
Optional parameters	403
Error codes	404
Start Service.	404
Required parameters	405
Error codes	405
Stop Channel	405
Required parameters	406
Optional parameters	406
Error codes	408
Stop Channel Initiator	409
Optional parameters	409
Stop Channel Listener	410
Required parameters	410
Optional parameters	410
Error codes	411
Stop Connection	412
Required parameters	412
Stop Service	412
Required parameters	412
Error codes	413
Suspend Queue Manager	413
Required parameters	413
Optional parameters	414
Suspend Queue Manager Cluster.	414
Required parameters	414
Optional parameters	415
Error codes	415
Chapter 4. Structures for commands and responses	417
How the structures are shown.	417
Data types	417
Initial values and default structures	417
Usage notes	418
MQCFH - PCF header	418
Fields	418
Language declarations	420
C language declaration	420
COBOL language declaration	421
PL/I language declaration (z/OS only).	421
System/390 assembler-language declaration (z/OS only)	421

Visual Basic language declaration (Windows only)	421
RPG language declaration (iSeries only)	422
MQCFBF - PCF byte string filter parameter	422
Fields	422
Language declarations	423
C language declaration	424
COBOL language declaration	424
PL/I language declaration (z/OS only)	424
System/390 assembler-language declaration (z/OS only)	424
Visual Basic language declaration (Windows only)	424
RPG language declaration (iSeries only)	424
MQCFBS - PCF byte string parameter	425
Fields	425
Language declarations	426
C language declaration	426
COBOL language declaration	426
PL/I language declaration (z/OS only)	426
System/390 assembler-language declaration (z/OS only)	427
Visual Basic language declaration (Windows only)	427
RPG language declaration (iSeries only)	427
MQCFIF - PCF integer filter parameter	427
Fields	427
Language declarations	429
C language declaration	429
COBOL language declaration	429
PL/I language declaration (z/OS only)	429
System/390 assembler-language declaration (z/OS only)	430
Visual Basic language declaration (Windows only)	430
RPG language declaration (iSeries only)	430
MQCFIL - PCF integer list parameter	430
Fields	430
Language declarations	431
C language declaration	432
COBOL language declaration	432
PL/I language declaration (z/OS only)	432
System/390 assembler-language declaration (z/OS only)	432
Visual Basic language declaration (Windows only)	432
RPG language declaration (iSeries only)	432
MQCFIN - PCF integer parameter	433
Fields	433
Language declarations	433
C language declaration	433
COBOL language declaration	434
PL/I language declaration (z/OS only)	434
System/390 assembler-language declaration (z/OS only)	434
Visual Basic language declaration (Windows only)	434
RPG language declaration (iSeries only)	434
MQCFSF - PCF string filter parameter	434
Fields	435
Language declarations	437
C language declaration	437
COBOL language declaration	438
PL/I language declaration (z/OS only)	438
System/390 assembler-language declaration (z/OS only)	438
Visual Basic language declaration (Windows only)	438
RPG language declaration (iSeries only)	438
MQCFSL - PCF string list parameter	439
Fields	439
Language declarations	441
C language declaration	441
COBOL language declaration	441
PL/I language declaration (z/OS only)	442
System/390 assembler-language declaration (z/OS only)	442
Visual Basic language declaration (Windows only)	442
RPG language declaration (iSeries only)	442
MQCFST - PCF string parameter	442
Fields	443
Language declarations	444
C language declaration	445
COBOL language declaration	445
PL/I language declaration (z/OS only)	445
System/390 assembler-language declaration (z/OS only)	445
Visual Basic language declaration (Windows only)	445
RPG language declaration (iSeries only)	445
Chapter 5. PCF example	447
Inquire local queue attributes	447
Program listing	447

Chapter 1. Introduction to Programmable Command Formats

This topic introduces WebSphere MQ Programmable Command Formats (PCFs) and their relationship to other parts of the WebSphere MQ products. It includes:

- “The problem PCF commands solve”
- “What PCFs are” on page 8
- “Other administration interfaces” on page 8
- “The WebSphere MQ Administration Interface (MQAI)” on page 9

The Programmable Command Formats described in this book are supported by:

- IBM WebSphere MQ for AIX
- IBM WebSphere MQ for HP-UX
- IBM WebSphere MQ for iSeries
- IBM WebSphere MQ for Linux
- IBM WebSphere MQ for Solaris
- IBM WebSphere MQ for Windows
- IBM WebSphere MQ for z/OS
- IBM MQSeries for Compaq NonStop Kernel, V5.1
- IBM WebSphere MQ for HP OpenVMS, V5.3

The problem PCF commands solve

The administration of distributed networks can become very complex. The problems of administration will continue to grow as networks increase in size and complexity.

Examples of administration specific to messaging and queuing include:

- Resource management.
For example, queue creation and deletion.
- Performance monitoring.
For example, maximum queue depth or message rate.
- Control.
For example, tuning queue parameters such as maximum queue depth, maximum message length, and enabling and disabling queues.
- Message routing.
Definition of alternative routes through a network.

WebSphere MQ PCF commands can be used to simplify queue manager administration and other network administration. PCF commands allow you to use a single application to perform network administration from a single queue manager within the network.

What PCFs are

PCFs define command and reply messages that can be exchanged between a program and any queue manager (that supports PCFs) in a network. You can use PCF commands in a systems management application program for administration of WebSphere MQ objects: authentication information objects, channels, channel listeners, namelists, process definitions, queue managers, queues, services, and storage classes. The application can operate from a single point in the network to communicate command and reply information with any queue manager, local or remote, via the local queue manager.

Each queue manager has an administration queue with a standard queue name and your application can send PCF command messages to that queue. Each queue manager also has a command server to service the command messages from the administration queue. PCF command messages can therefore be processed by any queue manager in the network and the reply data can be returned to your application, using your specified reply queue. PCF commands and reply messages are sent and received using the normal Message Queue Interface (MQI).

Other administration interfaces

Administration of WebSphere MQ objects can be carried out in other ways.

WebSphere MQ for iSeries

In addition to PCFs, there are two further administration interfaces:

i5/OS Control Language (CL)

This can be used to issue administration commands to WebSphere MQ for iSeries. They can be issued either at the command line or by writing a CL program. These commands perform similar functions to PCF commands, but the format is completely different. CL commands are designed exclusively for servers and CL responses are designed to be human-readable, whereas PCF commands are platform independent and both command and response formats are intended for program use.

WebSphere MQ Commands (MQSC)

These provide a uniform method of issuing commands across WebSphere MQ platforms. The general format of the commands is shown in the *WebSphere MQ Script (MQSC) Command Reference* manual.

To issue the commands on an iSeries server, create a list of commands in a Script file, and then run the file using the STRMQMMQSC command.

MQSC responses are designed to be human readable, whereas PCF command and response formats are intended for program use.

Note: MQSC responses to commands issued from a script file are returned in a spool file.

WebSphere MQ for z/OS

In addition to PCFs, WebSphere MQ for z/OS supports the WebSphere MQ commands (MQSC). With z/OS these commands can be entered from the z/OS console, or sent to the system command input queue. More information about issuing the commands is given in the *WebSphere MQ Script (MQSC) Command Reference* manual, and in the *WebSphere MQ for z/OS System Administration Guide*.

MQSeries for Compaq NonStop Kernel, V5.1

In addition to PCFs, there are three further administrative interfaces:

- WebSphere MQ commands (MQSC)
 - Control commands
 - Message Queue Management (MQM) facility
- MQSeries® for Compaq NonStop Kernel, V5.1 provides a panel interface for some of the functions.

WebSphere MQ for Windows, UNIX systems and HP OpenVMS

In addition to PCFs, there are three further administrative interfaces:

WebSphere MQ commands (MQSC)

You can use the MQSC as single commands issued at the Windows, or UNIX system command line. To issue more complicated, or multiple commands, the MQSC can be built into a file that you run from the Windows, or UNIX system command line. MQSC can be sent to a remote queue manager. For full details see the *WebSphere MQ Script (MQSC) Command Reference* manual.

Control commands

WebSphere MQ for Windows, and UNIX systems provides another type of command for some of the functions. These are the *control commands* that you issue at the system command line. Reference material for these commands is contained in the *WebSphere MQ System Administration* manual.

WebSphere MQ Explorer - WebSphere MQ for Windows and WebSphere MQ for Linux (x86 platform)

The WebSphere MQ Explorer is an Eclipse-based application that provides a graphical user interface for controlling resources in a network. For full details see the *WebSphere MQ System Administration Guide* manual.

The WebSphere MQ Administration Interface (MQAI)

In addition to the methods described in “Other administration interfaces” on page 8, WebSphere MQ for Windows, AIX, iSeries, Linux, HP-UX, and Solaris support the WebSphere MQ Administration Interface (MQAI).

The MQAI is a programming interface to WebSphere MQ that gives you an alternative to the MQI, for sending and receiving PCFs. The MQAI uses *data bags* which allow you to handle properties (or parameters) of objects more easily than using PCFs directly via the MQI.

The MQAI provides easier programming access to PCF messages by passing parameters into the data bag, so that only one statement is required for each structure. This removes the need for the programmer to handle arrays and allocate storage, and provides some isolation from the details of PCF.

The MQAI administers WebSphere MQ by sending PCF messages to the command server and waiting for a response.

The MQAI is described in the second section of this manual. See the *WebSphere MQ Using Java* book for a description of a component object model interface to the MQAI.

Chapter 2. Using Programmable Command Formats

This topic describes how to use the PCFs in a systems management application program for WebSphere MQ remote administration. The topic includes:

- “PCF command messages”
- “Responses” on page 13
- “Extended responses” on page 15
- “Rules for naming WebSphere MQ objects” on page 17
- “Authority checking for PCF commands” on page 18

PCF command messages

Each command and its parameters are sent as a separate command message containing a PCF header followed by a number of parameter structures (see “MQCFH - PCF header” on page 418). The PCF header identifies the command and the number of parameter structures that follow in the same message. Each parameter structure provides a parameter to the command.

Replies to the commands, generated by the command server, have a similar structure. There is a PCF header, followed by a number of parameter structures. Replies can consist of more than one message but commands always consist of one message only.

On platforms other than z/OS, the queue to which the PCF commands are sent is always called the SYSTEM.ADMIN.COMMAND.QUEUE. On z/OS, commands are sent to SYSTEM.COMMAND.INPUT, although SYSTEM.ADMIN.COMMAND.QUEUE can be an alias for it. The command server servicing this queue sends the replies to the queue defined by the *ReplyToQ* and *ReplyToQMgr* fields in the message descriptor of the command message.

How to issue PCF command messages

Use the normal Message Queue Interface (MQI) calls, MQPUT, MQGET and so on, to put and retrieve PCF command and response messages to and from their respective queues.

Note to users

Ensure that the command server is running on the target queue manager for the PCF command to process on that queue manager.

For a list of supplied header files, see the *WebSphere MQ Constants* manual.

Message descriptor for a PCF command

The WebSphere MQ message descriptor is fully documented in the *WebSphere MQ Application Programming Reference* manual.

A PCF command message contains the following fields in the message descriptor:

Report

Any valid value, as required.

MsgType

This must be MQMT_REQUEST to indicate a message requiring a response.

Expiry

Any valid value, as required.

Feedback

Set to MQFB_NONE

Encoding

If you are sending to iSeries, Windows or UNIX systems, set this field to the encoding used for the message data; conversion will be performed if necessary.

CodedCharSetId

If you are sending to iSeries, Windows, or UNIX systems, set this field to the coded character-set identifier used for the message data; conversion will be performed if necessary.

Format

Set to MQFMT_ADMIN.

Priority

Any valid value, as required.

Persistence

Any valid value, as required.

MsgId

The sending application may specify any value, or MQMI_NONE can be specified to request the queue manager to generate a unique message identifier.

CorrelId

The sending application may specify any value, or MQCI_NONE can be specified to indicate no correlation identifier.

ReplyToQ

The name of the queue to receive the response.

ReplyToQMgr

The name of the queue manager for the response (or blank).

Message context fields

These can be set to any valid values, as required. Normally the Put message option MQPMO_DEFAULT_CONTEXT is used to set the message context fields to the default values.

If you are using a version-2 MQMD structure, you must set the following additional fields:

GroupId

Set to MQGI_NONE

MsgSeqNumber

Set to 1

Offset

Set to 0

MsgFlags

Set to MQMF_NONE

OriginalLength
Set to MQOL_UNDEFINED

Sending user data

The PCF structures can also be used to send user-defined message data. In this case the message descriptor *Format* field should be set to MQFMT_PCF.

Responses

In response to each command, the command server generates one or more response messages. A response message has a similar format to a command message; the PCF header has the same command identifier value as the command to which it is a response (see “MQCFH - PCF header” on page 418 for details). The message identifier and correlation identifier are set according to the report options of the request.

If the PCF header type of the command message is MQCFT_COMMAND, standard responses only are generated. Such commands are supported on all platforms except z/OS®. Older applications will not support PCF on z/OS; the WebSphere MQ Windows Explorer is one such application (however, the Version 6.0 WebSphere MQ Explorer does support PCF on z/OS).

If the PCF header type of the command message is MQCFT_COMMAND_XR, either extended or standard responses are generated. Such commands are supported on z/OS and some other platforms. Commands issued on z/OS generate only extended responses. On other platforms, either type of response may be generated.

If a single command specifies a generic object name, a separate response is returned in its own message for each matching object. For the purpose of response generation, a single command with a generic name is treated as multiple individual commands (except for the control field MQCFC_LAST or MQCFC_NOT_LAST). Otherwise, one command message generates one response message.

Certain PCF responses might return a structure even when it is not requested. This is shown in the definition of the response (Chapter 3, “Definitions of the Programmable Command Formats,” on page 25) as *always returned*. The reason that, for these responses, it is necessary to name the objects in the response to identify which object the data applies.

Message descriptor for a response

A response message has the following fields in the message descriptor:

MsgType
This is MQMT_REPLY.

MsgId
This is generated by the queue manager.

CorrelId
This is generated according to the report options of the command message.

Format
This is MQFMT_ADMIN.

Encoding

Set to MQENC_NATIVE.

CodedCharSetId

Set to MQCCSI_Q_MGR.

Persistence

The same as in the command message.

Priority

The same as in the command message.

The response is generated with MQPMO_PASS_IDENTITY_CONTEXT.

Standard responses

If the PCF header type of the command message is MQCFT_COMMAND, standard responses only are generated. Such commands are supported on all platforms except z/OS.

There are three types of standard response:

- OK response
- Error response
- Data response

OK response

This consists of a message starting with a command format header, with a *CompCode* field of MQCC_OK or MQCC_WARNING.

For MQCC_OK, the *Reason* is MQRC_NONE.

For MQCC_WARNING, the *Reason* identifies the nature of the warning. In this case the command format header may be followed by one or more warning parameter structures appropriate to this reason code.

In either case, for an inquire command further parameter structures might follow as described below.

Error response

If the command has an error, one or more error response messages are sent (more than one might be sent even for a command that would normally have only a single response message). These error response messages have MQCFC_LAST or MQCFC_NOT_LAST set as appropriate.

Each such message starts with a response format header, with a *CompCode* value of MQCC_FAILED and a *Reason* field that identifies the particular error. In general each message describes a different error. In addition, each message has either zero or one (never more than one) error parameter structures following the header. This parameter structure, if there is one, is an MQCFIN structure, with a *Parameter* field containing one of the following:

- MQIACF_PARAMETER_ID

The *Value* field in the structure is the parameter identifier of the parameter that was in error (for example, MQCA_Q_NAME).

- MQIACF_ERROR_ID

This is used with a *Reason* value (in the command format header) of MQRC_UNEXPECTED_ERROR. The *Value* field in the MQCFIN structure is the unexpected reason code received by the command server.

- MQIACF_SELECTOR

This occurs if a list structure (MQCFIL) sent with the command contains a duplicate selector or one that is not valid. The *Reason* field in the command format header identifies the error, and the *Value* field in the MQCFIN structure is the parameter value in the MQCFIL structure of the command that was in error.

- MQIACF_ERROR_OFFSET

This occurs when there is a data compare error on the Ping Channel command. The *Value* field in the structure is the offset of the Ping Channel compare error.

- MQIA_CODED_CHAR_SET_ID

This occurs when the coded character-set identifier in the message descriptor of the incoming PCF command message does not match that of the target queue manager. The *Value* field in the structure is the coded character-set identifier of the queue manager.

The last (or only) error response message is a summary response, with a *CompCode* field of MQCC FAILED, and a *Reason* field of MQRCCF_COMMAND_FAILED. This message has no parameter structure following the header.

Data response

This consists of an OK response (as described above) to an inquire command. The OK response is followed by additional structures containing the requested data as described in Chapter 3, “Definitions of the Programmable Command Formats,” on page 25.

Applications should not depend upon these additional parameter structures being returned in any particular order.

Extended responses

Commands issued on z/OS generate extended responses only. There are three types of extended response:

- Message response, with type MQCFT_XR_MSG
- Item response, with type MQCFT_XR_ITEM
- Summary response, with type MQCFT_XR_SUMMARY

Each command may generate one, or more, sets of responses. Each set of responses comprises one or more messages, numbered sequentially from 1 in the *MsgSeqNumber* field of the PCF header. The *Control* field of the last (or only) response in each set has the value MQCFC_LAST. For all other responses in the set, this value is MQCFC_NOT_LAST.

Any response may include one, or more, optional MQCFBS structures in which the *Parameter* field is set to MQBACF_RESPONSE_SET, the value being a response set identifier. Identifiers are unique and identify the set of responses which contain the response. For every set of responses, there is an MQCFBS structure that identifies it.

Extended responses have at least two parameter structures:

- An MQCFBS structure with the *Parameter* field set to MQBACF_RESPONSE_ID. The value in this field is the identifier of the set of responses to which the response belongs. The identifier in the first set is arbitrary. In subsequent sets, the identifier is one previously notified in an MQBACF_RESPONSE_SET structure.
- An MQCFST structure with the *Parameter* field set to MQCACF_RESPONSE_Q_MGR_NAME, the value being the name of the queue manager from which the set of responses come.

Many responses have additional parameter structures, and these are described in “Extended responses to Inquire commands,” “Extended responses to commands other than Inquire,” and “Extended responses to commands using CommandScope” on page 17.

You cannot determine in advance how many responses there will be in a set other than by getting responses until one with MQCFC_LAST is found. Neither can you determine in advance how many sets of responses there will be as any set may include MQBACF_RESPONSE_SET structures to indicate that additional sets will be generated.

Extended responses to Inquire commands

Inquire commands normally generate an item response (type MQCFT_XR_ITEM) for each item found that matches the specified search criteria. The item response has a *CompCode* field in the header with a value of MQCC_OK, and a *Reason* field with a value of MQRC_NONE. It also includes other parameter structures describing the item and its requested attributes, as described in “Definitions of Programmable Command Formats” on page 34.

If an item is in error, the *CompCode* field in the header has a value of MQCC_FAILED and the *Reason* field identifies the particular error. Additional parameter structures are included to identify the item.

Certain Inquire commands may return general (not name-specific) message responses in addition to the item responses. These are informational, or error, responses of the type MQCFT_XR_MSG.

If the Inquire command succeeds, there may, optionally, be a summary response (type MQCFT_XR_SUMMARY), with a *CompCode* value of MQCC_OK, and a *Reason* field value of MQRC_NONE.

If the Inquire command fails, item responses may be returned, and there may optionally be a summary response (type MQCFT_XR_SUMMARY), with a *CompCode* value of MQCC_FAILED, and a *Reason* field value of MQRCCF_COMMAND_FAILED.

Extended responses to commands other than Inquire

Successful commands generate message responses in which the *CompCode* field in the header has a value of MQCC_OK, and the *Reason* field has a value of MQRC_NONE. There will always be at least one message; it may be informational (MQCFT_XR_MSG) or a summary (MQCFT_XR_SUMMARY). There may optionally be additional informational (type MQCFT_XR_MSG) messages. Each

informational message may include a number of additional parameter structures with information about the command; see the individual command descriptions for the structures that may occur.

Commands that fail generate error message responses (type MQCFT_XR_MSG), in which the *CompCode* field in the header has a value of MQCC_FAILED and the *Reason* field identifies the particular error. Each message may include a number of additional parameter structures with information about the error: see the individual error descriptions for the structures that may occur. Informational message responses may be generated. There may, optionally, be a summary response (MQCFT_XR_SUMMARY), with a *CompCode* value of MQCC_FAILED, and a *Reason* field value of MQRCCF_COMMAND_FAILED.

Extended responses to commands using CommandScope

If a command uses the *CommandScope* parameter, or causes a command using the *CommandScope* parameter to be generated, there is an initial response set from the queue manager where the command was received. Then a separate set, or sets, of responses is generated for each queue manager to which the command is directed (as if multiple individual commands were issued). Finally, there is a response set from the receiving queue manager which includes an overall summary response (type MQCFT_XR_SUMMARY). The MQCACF_RESPONSE_Q_MGR_NAME parameter structure identifies the queue manager that generates each set.

The initial response set has the following additional parameter structures:

- MQIACF_COMMAND_INFO (MQCFIN). Possible values in this structure are MQCMDI_CMDSCOPE_ACCEPTED or MQCMDI_CMDSCOPE_GENERATED.
- MQIACF_CMDSCOPE_Q_MGR_COUNT (MQCFIN). This indicates the number of queue managers to which the command is sent.

Rules for naming WebSphere MQ objects

WebSphere MQ authentication information, channel, client channel, listener, namelist, process, queue, service and storage class objects exist in separate object *name spaces*, and so objects from each type can all have the same name. However, an object cannot have the same name as any other object in the same name space. (For example, a local queue cannot have the same name as a model queue.) Names in WebSphere MQ are case sensitive.

The character set that can be used for naming all WebSphere MQ objects is as follows:

- Uppercase A–Z
- Lowercase a–z (however, on systems using EBCDIC Katakana you cannot use lowercase characters, and there are also restrictions on the use of lowercase letters for z/OS console support)
- Numerics 0–9
- Period (.)
- Forward slash (/)
- Underscore (_)
- Percent sign (%). The percent sign (%) is a special character to RACF®. If you are using RACF as the external security manager for WebSphere MQ for z/OS, you should not use % in object names. If you do, these names are not included in any security checks when RACF generic profiles are used.

Note:

1. Leading or embedded blanks are not allowed.
2. Avoid using names with leading or trailing underscores, because they cannot be handled by the WebSphere MQ for z/OS operations and control panels.
3. Any name that is less than the full field length can be padded to the right with blanks. All short names that are returned by the queue manager are always padded to the right with blanks.
4. Any structure to the names (for example, the use of the period or underscore) is not significant to the queue manager.

Name lengths

Queues can have names up to 48 characters long. Processes, namelists, clusters, and authentication information objects can have names up to 48 characters long. Channels can have names up to 20 characters long. Storage classes can have names up to 8 characters long. CF structures can have names up to 12 characters long.

Reserved object names

Names that start with “SYSTEM.” are reserved for objects defined by the queue manager. You can use the Change commands to change these object definitions to suit your installation. The names that are defined for WebSphere MQ are listed in full in the *WebSphere MQ Script (MQSC) Command Reference* manual.

Generic values

Wherever a parameter can have a generic value, it is entered ending with an asterisk (*), for example ABC*. A generic value means ‘all values beginning with’; so ABC* means ‘all values beginning with ABC’.

The question mark (?) and colon (:) are not allowed in generic values.

Authority checking for PCF commands

When a PCF command is processed, the *UserIdentifier* from the message descriptor in the command message is used for the required WebSphere MQ object authority checks. The checks are performed on the system on which the command is being processed; therefore this user ID must exist on the target system and have the required authorities to process the command. If the message has come from a remote system, one way of achieving this is to have a matching user ID on both the local and remote systems.

Authority checking is implemented differently on each platform.

WebSphere MQ for iSeries

In order to process any PCF command, the user ID must have *dsp* authority for the WebSphere MQ object on the target system.

In addition, WebSphere MQ object authority checks are performed for certain PCF commands, as shown in Table 1 on page 20.

In most cases these are the same checks as those performed by the equivalent WebSphere MQ CL commands issued on a local system. See the *WebSphere MQ for iSeries System Administration* book for more information on the mapping from WebSphere MQ authorities to i5/OS system authorities, and the authority requirements for the WebSphere MQ CL commands. Details of security concerning exits are given in the *WebSphere MQ Intercommunications* manual.

To process any of the following commands the user ID must be a member of the group profile QMQADM:

- Ping Channel
- Change Channel
- Copy Channel
- Create Channel
- Delete Channel
- Reset Channel
- Resolve Channel
- Start Channel
- Stop Channel
- Start Channel Initiator
- Start Channel Listener

WebSphere MQ for Windows, and UNIX systems

In order to process any PCF command, the user ID must have *dsp* authority for the queue manager object on the target system. In addition, WebSphere MQ object authority checks are performed for certain PCF commands, as shown in Table 1 on page 20.

To process any of the following commands the user ID must belong to group *mqm*.

Note: For Windows **only**, the user ID can belong to group *Administrators* or group *mqm*.

- Change Channel
- Copy Channel
- Create Channel
- Delete Channel
- Ping Channel
- Reset Channel
- Start Channel
- Stop Channel
- Start Channel Initiator
- Start Channel Listener
- Resolve Channel
- Reset Cluster
- Refresh Cluster
- Suspend Queue Manager
- Resume Queue Manager

WebSphere MQ for HP OpenVMS and Compaq NonStop Kernel

In order to process any PCF command, the user ID must have *dsp* authority for the queue manager object on the target system. In addition, WebSphere MQ object authority checks are performed for certain PCF commands, as shown in Table 1.

To process any of the following commands the user ID must belong to group *mqm*:

- Change Channel
- Copy Channel
- Create Channel
- Delete Channel
- Ping Channel
- Reset Channel
- Start Channel
- Stop Channel
- Start Channel Initiator
- Start Channel Listener
- Resolve Channel
- Reset Cluster
- Refresh Cluster
- Suspend Queue Manager
- Resume Queue Manager

Table 1. Windows,HP OpenVMS Alpha, NP NonStop Server, and UNIX systems - object authorities

Command	WebSphere MQ object authority	Class authority (for object type)
Change Authentication Information	<i>dsp</i> and <i>chg</i>	n/a
Change Channel	<i>dsp</i> and <i>chg</i>	n/a
Change Channel Listener	<i>dsp</i> and <i>chg</i>	n/a
Change Client Connection Channel	<i>dsp</i> and <i>chg</i>	n/a
Change Namelist	<i>dsp</i> and <i>chg</i>	n/a
Change Process	<i>dsp</i> and <i>chg</i>	n/a
Change Queue	<i>dsp</i> and <i>chg</i>	n/a
Change Queue Manager	<i>chg</i> <i>see Note 3</i>	n/a
Change Service	<i>dsp</i> and <i>chg</i>	n/a
Clear Queue	<i>clr</i>	n/a
Copy Authentication Information	<i>dsp</i>	<i>crt</i>
Copy Authentication Information (Replace) <i>see Note 1</i>	<i>from: dsp to: chg</i>	<i>crt</i>
Copy Channel	<i>dsp</i>	<i>crt</i>

Table 1. Windows,HP OpenVMS Alpha, NP NonStop Server, and UNIX systems - object authorities (continued)

Command	WebSphere MQ object authority	Class authority (for object type)
Copy Channel (Replace) <i>see Note 1</i>	<i>from: dsp to: chg</i>	crt
Copy Channel Listener	dsp	crt
Copy Channel Listener (Replace) <i>see Note 1</i>	<i>from: dsp to: chg</i>	crt
Copy Client Connection Channel	dsp	crt
Copy Client Connection Channel (Replace) <i>see Note 1</i>	<i>from: dsp to: chg</i>	crt
Copy Namelist	dsp	crt
Copy Namelist (Replace) <i>see Note 1</i>	<i>from: dsp to: dsp and chg</i>	crt
Copy Process	dsp	crt
Copy Process (Replace) <i>see Note 1</i>	<i>from: dsp to: chg</i>	crt
Copy Queue	dsp	crt
Copy Queue (Replace) <i>see Note 1</i>	<i>from: dsp to: dsp and chg</i>	crt
Create Authentication Information	<i>(system default authentication information) dsp</i>	crt
Create Authentication Information (Replace) <i>see Note 1</i>	<i>(system default authentication information) dsp to: chg</i>	crt
Create Channel	<i>(system default channel) dsp</i>	crt
Create Channel (Replace) <i>see Note 1</i>	<i>(system default channel) dsp to: chg</i>	crt
Create Channel Listener	<i>(system default listener) dsp</i>	crt
Create Channel Listener (Replace) <i>see Note 1</i>	<i>(system default listener) dsp to: chg</i>	crt
Create Client Connection Channel	<i>(system default channel) dsp</i>	crt
Create Client Connection Channel (Replace) <i>see Note 1</i>	<i>(system default channel) dsp to: chg</i>	crt
Create Namelist	<i>(system default namelist) dsp</i>	crt
Create Namelist (Replace) <i>see Note 1</i>	<i>(system default namelist) dsp to: dsp and chg</i>	crt
Create Process	<i>(system default process) dsp</i>	crt
Create Process (Replace) <i>see Note 1</i>	<i>(system default process) dsp to: chg</i>	crt
Create Queue	<i>(system default queue) dsp</i>	crt
Create Queue (Replace) <i>see Note 1</i>	<i>(system default queue) dsp to: dsp and chg</i>	crt
Create Service	<i>(system default queue) dsp</i>	crt

Table 1. Windows,HP OpenVMS Alpha, NP NonStop Server, and UNIX systems - object authorities (continued)

Command	WebSphere MQ object authority	Class authority (for object type)
Create Service (Replace) <i>see Note 1</i>	(<i>system default queue</i>) dsp to: chg	crt
Delete Authentication Information	dsp and dlt	n/a
Delete Authority Record	(<i>queue manager object</i>) chg <i>see Note 4</i>	<i>see Note 4</i>
Delete Channel	dsp and dlt	n/a
Delete Channel Listener	dsp and dlt	n/a
Delete Client Connection Channel	dsp and dlt	n/a
Delete Namelist	dsp and dlt	n/a
Delete Process	dsp and dlt	n/a
Delete Queue	dsp and dlt	n/a
Delete Service	dsp and dlt	n/a
Inquire Authentication Information	dsp	n/a
Inquire Authority Records	<i>see Note 4</i>	<i>see Note 4</i>
Inquire Channel	dsp	n/a
Inquire Channel Listener	dsp	n/a
Inquire Client Connection Channel	dsp	n/a
Inquire Namelist	dsp	n/a
Inquire Process	dsp	n/a
Inquire Queue	dsp	n/a
Inquire Queue Manager	<i>see note 3</i>	n/a
Inquire Service	dsp	n/a
Ping Channel	ctrl	n/a
Ping Queue Manager	<i>see note 3</i>	n/a
Reset Channel	ctrlx	n/a
Reset Queue Statistics	dsp and chg	n/a
Resolve Channel	ctrlx	n/a
Set Authority Record	(<i>queue manager object</i>) chg <i>see Note 4</i>	<i>see Note 4</i>
Start Channel	ctrl	n/a
Stop Channel	ctrl	n/a
Escape	<i>see Note 2</i>	<i>see Note 2</i>

Table 1. Windows, HP OpenVMS Alpha, NP NonStop Server, and UNIX systems - object authorities (continued)

Command	WebSphere MQ object authority	Class authority (for object type)
Note:		
	<ol style="list-style-type: none">1. This applies if the object to be replaced does already exist, otherwise the authority check is as for Create or Copy without Replace.2. The required authority is determined by the MQSC command defined by the escape text, and it will be equivalent to one of the above.3. In order to process any PCF command, the user ID must have dsp authority for the queue manager object on the target system.4. This PCF command will be authorized unless the command server has been started with the -a parameter. By default the command server starts when the Queue Manager is started, and without the -a parameter. See the System Administration Guide for further information.	

WebSphere MQ also supplies some channel security exit points so that you can supply your own user exit programs for security checking. Details are given in the *WebSphere MQ Intercommunications* manual.

WebSphere MQ for z/OS

See the *WebSphere MQ for z/OS System Setup Guide* for information about authority checking on z/OS.

Chapter 3. Definitions of the Programmable Command Formats

The topic discusses:

- “How the definitions are shown”
- “PCF commands and responses in groups” on page 30

Following is the reference material for all Programmable Command Formats (PCFs) of commands and responses.

How the definitions are shown

For each PCF command or response there is a description of what the command or response does, giving the command identifier in parentheses. See the *WebSphere MQ Constants* manual for all values of the command identifier. Each command description starts with a table that identifies the platforms on which the command is valid. For additional, more detailed, usage notes for each command, see the corresponding command description in the *WebSphere MQ Script (MQSC) Command Reference* manual.

WebSphere MQ products, other than WebSphere MQ for z/OS, can use the WebSphere MQ Administration Interface (MQAI), which provides a simplified way for applications written in the C and Visual Basic programming language to build and send PCF commands. For information on the MQAI see the second section of this manual.

On Windows, you can use the Microsoft® Active Directory Services Interface (ADSI), as well as PCFs, to inquire about and set parameters. For information on using Microsoft ADSI see the *WebSphere MQ for Windows Using the Component Object Model Interface* book.

Commands

The *required parameters* and the *optional parameters* are listed. On platforms other than z/OS, the parameters *must* occur in the order:

1. All required parameters, in the order stated, followed by
2. Optional parameters as required, in any order, unless specifically noted in the PCF definition.

On z/OS, the parameters can be in any order.

Responses

The response data attribute is *always returned* whether it is requested or not. This parameter is required to identify, uniquely, the object when there is a possibility of multiple reply messages being returned.

The other attributes shown are *returned if requested* as optional parameters on the command. The response data attributes are not returned in a defined order.

Parameters and response data

Each parameter name is followed by its structure name in parentheses (details are given in Chapter 4, “Structures for commands and responses,” on page 417). The parameter identifier is given at the beginning of the description.

Constants

The values of constants used by PCF commands and responses are in the *WebSphere MQ Constants* manual.

Informational messages

On z/OS, a number of command responses return a structure, `MQIACF_COMMAND_INFO`, with values that provide information about the command.

Table 2. MQIACF_COMMAND_INFO values

MQIACF_COMMAND_INFO value	Meaning
<code>MQCMDI_CMDSCOPE_ACCEPTED</code>	A command that specified <i>CommandScope</i> was entered. It has been passed to the requested queue manager(s) for processing
<code>MQCMDI_CMDSCOPE_GENERATED</code>	A command that specified <i>CommandScope</i> was generated in response to the command originally entered
<code>MQCMDI_CMDSCOPE_COMPLETED</code>	Processing for the command that specified <i>CommandScope</i> - either entered or generated by another command - has completed successfully on all requested queue managers
<code>MQCMDI_QSG_DISP_COMPLETED</code>	Processing for the command that refers to an object with the indicated disposition has completed successfully
<code>MQCMDI_COMMAND_ACCEPTED</code>	Initial processing for the command has completed successfully. The command requires further action by the channel initiator, for which a request has been queued. Messages reporting the success or otherwise of the action will be sent to the command issuer subsequently
<code>MQCMDI_CLUSTER_REQUEST_QUEUED</code>	Initial processing for the command has completed successfully. The command requires further action by the cluster repository manager, for which a request has been queued
<code>MQCMDI_CHANNEL_INIT_STARTED</code>	A Start Channel Initiator command has been issued and the channel initiator address space has been started successfully
<code>MQCMDI_RECOVER_STARTED</code>	The queue manager has successfully started a task to process the Recover CF Structure command for the named structure
<code>MQCMDI_BACKUP_STARTED</code>	The queue manager has successfully started a task to process the Backup CF Structure command for the named structure

Table 2. MQIACF_COMMAND_INFO values (continued)

MQIACF_COMMAND_INFO value	Meaning
MQCMDI_RECOVER_COMPLETED	The named CF structure has been recovered successfully. The structure is available for use again
MQCMDI_SEC_TIMER_ZERO	The Change Security command was entered with the <i>SecurityInterval</i> attribute set to 0. This means that no user timeouts will occur
MQCMDI_REFRESH_CONFIGURATION	A Change Queue Manager command has been issued that enables configuration events. Event messages need to be generated to ensure that the configuration information is complete and up-to-date
MQCMDI_IMS_BRIDGE_SUSPENDED	The MQ-IMS Bridge facility is suspended.
MQCMDI_DB2_SUSPENDED	The connection to DB2 is suspended
MQCMDI_DB2_OBSOLETE_MSGS	Obsolete DB2 messages exist in the queue-sharing group

Error codes

At the end of most command format definitions there is a list of error codes that might be returned by that command.

Error codes applicable to all commands

In addition to those listed under each command format, any command might return the following in the response format header (descriptions of the MQRC_* error codes are given in the *WebSphere MQ Messages* and *WebSphere MQ for z/OS Messages and Codes* manuals):

Reason (MQLONG)

The value can be:

MQRC_NONE

(0, X'000') No reason to report.

MQRC_MSG_TOO_BIG_FOR_Q

(2030, X'7EE') Message length greater than maximum for queue.

MQRC_CONNECTION_BROKEN

(2009, X'7D9') Connection to queue manager lost.

MQRC_NOTAUTHORIZED

(2035, X'7F3') Not authorized for access.

MQRC_UNKNOWN_OBJECT_NAME

(2067, X'813') Attribute selector not valid.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') Insufficient storage available.

MQRC_UNKNOWN_OBJECT_NAME

(2085, X'825') Unknown object name.

MQRCCF_ATTR_VALUE_ERROR

Attribute value not valid.

MQRCCF_CFBF_FILTER_VAL_LEN_ERROR
Filter value length not valid.

MQRCCF_CFBF_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFBF_OPERATOR_ERROR
Operator error.

MQRCCF_CFBF_PARM_ID_ERROR
Parameter identifier not valid.

MQRCCF_CFBS_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFBS_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFBS_PARM_ID_ERROR
Parameter identifier not valid.

MQRCCF_CFBS_STRING_LENGTH_ERROR
String length not valid.

MQRCCF_CFGR_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFGR_PARM_COUNT_ERROR
Parameter count not valid.

MQRCCF_CFGR_PARM_ID_ERROR
Parameter identifier not valid.

MQRCCF_CFH_COMMAND_ERROR
Command identifier not valid.

MQRCCF_CFH_CONTROL_ERROR
Control option not valid.

MQRCCF_CFH_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFH_MSG_SEQ_NUMBER_ERR
Message sequence number not valid.

MQRCCF_CFH_PARM_COUNT_ERROR
Parameter count not valid.

MQRCCF_CFH_TYPE_ERROR
Type not valid.

MQRCCF_CFH_VERSION_ERROR
Structure version number is not valid.

MQRCCF_CFIF_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFIF_OPERATOR_ERROR
Operator error.

MQRCCF_CFIF_PARM_ID_ERROR
Parameter identifier not valid.

MQRCCF_CFIL_COUNT_ERROR
Count of parameter values not valid.

MQRCCF_CFIL_DUPLICATE_VALUE
Duplicate parameter.

MQRCCF_CFIL_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFIN_PARM_ID_ERROR
Parameter identifier not valid.

MQRCCF_CFIN_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFIN_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFIN_PARM_ID_ERROR
Parameter identifier not valid.

MQRCCF_CFSF_FILTER_VAL_LEN_ERROR
Filter value length not valid.

MQRCCF_CFSF_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFSF_OPERATOR_ERROR
Operator error.

MQRCCF_CFSF_PARM_ID_ERROR
Parameter identifier not valid.

MQRCCF_CFSL_COUNT_ERROR
Count of parameter values not valid.

MQRCCF_CFSL_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFSL_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFSL_PARM_ID_ERROR
Parameter identifier not valid.

MQRCCF_CFSL_STRING_LENGTH_ERROR
String length value not valid.

MQRCCF_CFSL_TOTAL_LENGTH_ERROR
Total string length error.

MQRCCF_CFST_CONFLICTING_PARM
Conflicting parameters.

MQRCCF_CFST_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFST_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFST_PARM_ID_ERROR
Parameter identifier not valid.

MQRCCF_CFST_STRING_LENGTH_ERROR
String length value not valid.

MQRCCF_COMMAND_FAILED
Command failed.

MQRCCF_ENCODING_ERROR
Encoding error.

MQRCCF_MD_FORMAT_ERROR
Format not valid.

MQRCCF_MSG_SEQ_NUMBER_ERROR
Message sequence number not valid.

MQRCCF_MSG_TRUNCATED
Message truncated.

MQRCCF_MSG_LENGTH_ERROR
Message length not valid.

MQRCCF_OBJECT_NAME_ERROR
Object name not valid.

MQRCCF_OBJECT_OPEN
Object is open.

MQRCCF_PARM_COUNT_TOO_BIG
Parameter count too big.

MQRCCF_PARM_COUNT_TOO_SMALL
Parameter count too small.

MQRCCF_PARM_SEQUENCE_ERROR
Parameter sequence not valid.

MQRCCF_PARM_SYNTAX_ERROR
Syntax error found in parameter.

MQRCCF_STRUCTURE_TYPE_ERROR
Structure type not valid.

PCF commands and responses in groups

The commands and data responses are given in alphabetic order in this book.

They can be usefully grouped as follows:

Authentication Information commands

- “Change, Copy, and Create Authentication Information Object” on page 35
- “Delete Authentication Information Object” on page 132
- “Inquire Authentication Information Object” on page 152
- “Inquire Authentication Information Object Names” on page 156

Authority Record commands

- “Delete Authority Record” on page 134
- “Inquire Authority Records” on page 158
- “Inquire Entity Authority” on page 248
- “Set Authority Record” on page 390

CF commands

- “Backup CF Structure” on page 34
- “Change, Copy, and Create CF Structure” on page 38

- “Delete CF Structure” on page 135
- “Inquire CF Structure” on page 166
- “Inquire CF Structure Names” on page 168
- “Inquire CF Structure Status” on page 169
- “Recover CF Structure” on page 366

Channel commands

- “Change, Copy and Create Channel” on page 41
- “Change, Copy, and Create Channel Listener” on page 70
- “Delete Channel” on page 136
- “Delete Channel Listener” on page 138
- “Inquire Channel” on page 174
- “Inquire Channel Initiator” on page 189
- “Inquire Channel Listener” on page 192
- “Inquire Channel Listener Status” on page 197
- “Inquire Channel Names” on page 202
- “Inquire Channel Status” on page 205
- “Ping Channel” on page 362
- “Reset Channel” on page 373
- “Resolve Channel” on page 380
- “Start Channel” on page 397
- “Start Channel Initiator” on page 401
- “Start Channel Listener” on page 402
- “Stop Channel” on page 405
- “Stop Channel Initiator” on page 409
- “Stop Channel Listener” on page 410

Cluster commands

- “Inquire Cluster Queue Manager” on page 226
- “Refresh Cluster” on page 367
- “Reset Cluster” on page 375
- “Resume Queue Manager Cluster” on page 384
- “Suspend Queue Manager Cluster” on page 414

Connection commands

- “Inquire Connection” on page 239
- “Stop Connection” on page 412

Escape command

- “Escape” on page 145

Namelist commands

- “Change, Copy, and Create Namelist” on page 72
- “Delete Namelist” on page 138
- “Inquire Namelist” on page 259
- “Inquire Namelist Names” on page 264

Process commands

- “Change, Copy, and Create Process” on page 76
- “Delete Process” on page 140
- “Inquire Process” on page 266
- “Inquire Process Names” on page 270

Queue commands

- “Change, Copy, and Create Queue” on page 80
- “Clear Queue” on page 131
- “Delete Queue” on page 141
- “Inquire Queue” on page 272
- “Inquire Queue Names” on page 322
- “Move Queue” on page 360
- “Reset Queue Statistics” on page 378

Queue Manager commands

- “Change Queue Manager” on page 99
- “Inquire Queue Manager” on page 291
- “Inquire Queue Manager Status” on page 319
- “Ping Queue Manager” on page 366
- “Refresh Queue Manager” on page 369
- “Reset Queue Manager” on page 377

Security commands

- “Change Security” on page 124
- “Inquire Security” on page 336
- “Refresh Security” on page 371
- “Reverify Security” on page 385

Service commands

- “Change, Copy, and Create Service” on page 125
- “Delete Service” on page 144
- “Inquire Service” on page 339
- “Inquire Service Status” on page 342
- “Start Service” on page 404
- “Stop Service” on page 412

Storage class commands

- “Change, Copy, and Create Storage Class” on page 128
- “Delete Storage Class” on page 144
- “Inquire Storage Class” on page 346
- “Inquire Storage Class Names” on page 350

System commands

- “Inquire Archive” on page 147
- “Set Archive” on page 386
- “Inquire Group” on page 253
- “Inquire Log” on page 255
- “Set Log” on page 394
- “Inquire System” on page 352
- “Set System” on page 396
- “Inquire Usage” on page 357

Data responses to commands

- “Escape (Response)” on page 146
- “Inquire Archive (Response)” on page 148
- “Inquire Authentication Information Object (Response)” on page 154
- “Inquire Authentication Information Object Names (Response)” on page 157
- “Inquire Authority Records (Response)” on page 161
- “Inquire CF Structure (Response)” on page 167
- “Inquire CF Structure Names (Response)” on page 169
- “Inquire CF Structure Status (Response)” on page 171
- “Inquire Channel (Response)” on page 181
- “Inquire Channel Initiator (Response)” on page 190
- “Inquire Channel Listener (Response)” on page 195
- “Inquire Channel Listener Status (Response)” on page 199
- “Inquire Channel Names (Response)” on page 204
- “Inquire Channel Status (Response)” on page 216
- “Inquire Cluster Queue Manager (Response)” on page 231
- “Inquire Connection (Response)” on page 243
- “Inquire Entity Authority (Response)” on page 251
- “Inquire Group (Response)” on page 253
- “Inquire Log (Response)” on page 256
- “Inquire Namelist (Response)” on page 262
- “Inquire Namelist Names (Response)” on page 265
- “Inquire Process (Response)” on page 268
- “Inquire Process Names (Response)” on page 271
- “Inquire Queue (Response)” on page 281
- “Inquire Queue Manager (Response)” on page 299
- “Inquire Queue Manager Status (Response)” on page 320
- “Inquire Queue Names (Response)” on page 324
- “Reset Queue Statistics (Response)” on page 379
- “Inquire Security (Response)” on page 337
- “Inquire Service (Response)” on page 340
- “Inquire Service Status (Response)” on page 344
- “Inquire Storage Class (Response)” on page 349
- “Inquire Storage Class Names (Response)” on page 351
- “Inquire System (Response)” on page 353

- “Inquire Usage (Response)” on page 358

Definitions of Programmable Command Formats

Reference information for the Programmable Command Formats (PCFs) of commands and responses sent between a WebSphere MQ systems management application program and a WebSphere MQ queue manager now follows.

Backup CF Structure

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The Backup CF Structure (MQCMD_BACKUP_CF_STRUC) command initiates a CF application structure backup.

Note: This command is supported only on z/OS when the queue manager is a member of a queue-sharing group.

Required parameters

CFStrucName

Optional parameters:

CommandScope, *ExcludeInterval*

Required parameters

CFStrucName (MQCFST)

The name of the CF application structure to be backed up (parameter identifier: MQCA_CF_STRUC_NAME).

The maximum length is MQ_CF_STRUC_NAME_LENGTH.

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.

The maximum length is MQ_QSG_NAME_LENGTH.

ExcludeInterval (MQCFIN)

Exclude interval (parameter identifier: MQIACF_EXCLUDE_INTERVAL).

Specifies a value in seconds that defines the length of time immediately before the current time where the backup starts. The backup excludes backing-up the

last *n* seconds activity. For example, if 30 seconds is specified, the backup does not include the last 30 seconds worth of activity for this application-structure.

The value must be in the range 30 through 600. The default value is 30.

Change, Copy, and Create Authentication Information Object

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Change authentication information (MQCMD_CHANGE_AUTH_INFO) command changes the specified attributes in an authentication information object. For any optional parameters that are omitted, the value does not change.

The Copy authentication information (MQCMD_COPY_AUTH_INFO) command creates a new authentication information object using, for attributes not specified in the command, the attribute values of an existing authentication information object.

The Create authentication information (MQCMD_CREATE_AUTH_INFO) command creates an authentication information object. Any attributes that are not defined explicitly are set to the default values on the destination queue manager. A system default authentication information object exists and default values are taken from it.

Required parameters (Change authentication information):
AuthInfoName

Required parameters (Copy authentication information):
FromAuthInfoName, *ToAuthInfoName*, *AuthInfoType*

Required parameters (Create authentication information):
AuthInfoName, *AuthInfoType*, *AuthInfoConnName*

Optional parameters:

AuthInfoConnName, *AuthInfoDesc*, *CommandScope*, *LDAPPassword*,
LDAPUserName, *QSGDisposition*

Required parameters (Change authentication information)

AuthInfoName (MQCFST)

The authentication information object name (parameter identifier: MQCA_AUTH_INFO_NAME).

The maximum length of the string is MQ_AUTH_INFO_NAME_LENGTH.

Required parameters (Copy authentication information)

FromAuthInfoName (MQCFST)

The name of the authentication information object definition to be copied from (parameter identifier: MQCACF_FROM_AUTH_INFO_NAME).

On z/OS, the queue manager searches for an object with the name you specify and a disposition of MQQSGD_Q_MGR or MQQSGD_COPY to copy from. This parameter is ignored if a value of MQQSGD_COPY is specified for

QSGDisposition. In this case, an object with the name specified by *ToAuthInfoName* and the disposition of MQQSGD_GROUP is searched for to copy from.

The maximum length of the string is MQ_AUTH_INFO_NAME_LENGTH.

AuthInfoName (MQCFST)

The name of the authentication information object to copy to (parameter identifier: MQCACF_TO_AUTH_INFO_NAME).

The maximum length of the string is MQ_AUTH_INFO_NAME_LENGTH.

AuthInfoType (MQCFIN)

The type of authentication information object (parameter identifier: MQIA_AUTH_INFO_TYPE).

The value can be:

MQAIT_CRL_LDAP

This defines this authentication information object as specifying Certificate Revocation Lists that are held on the LDAP. Please see the *WebSphere MQ Security* book for more information.

Required parameters (Create authentication information)

AuthInfoName (MQCFST)

Authentication information object name (parameter identifier: MQCA_AUTH_INFO_NAME).

The maximum length of the string is MQ_AUTH_INFO_NAME_LENGTH.

AuthInfoType (MQCFIN)

The type of authentication information object (parameter identifier: MQIA_AUTH_INFO_TYPE).

The value can be:

MQAIT_CRL_LDAP

This defines this authentication information object as specifying Certificate Revocation Lists that are held on the LDAP. Please see the *WebSphere MQ Security* book for more information.

AuthInfoConnName (MQCFST)

The connection name of the authentication information object (parameter identifier: MQCA_AUTH_INFO_CONN_NAME).

On platforms other than z/OS, the maximum length is MQ_AUTH_INFO_CONN_NAME_LENGTH. On z/OS, it is MQ_LOCAL_ADDRESS_LENGTH.

Optional parameters

AuthInfoConnName (MQCFST)

The connection name of the authentication information object (parameter identifier: MQCA_AUTH_INFO_CONN_NAME).

On platforms other than z/OS, the maximum length is MQ_AUTH_INFO_CONN_NAME_LENGTH. On z/OS, it is MQ_LOCAL_ADDRESS_LENGTH.

AuthInfoDesc (MQCFST)

The description of the authentication information object (parameter identifier: MQCA_AUTH_INFO_DESC).

The maximum length is MQ_AUTH_INFO_DESC_LENGTH.

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

LDAPPassword (MQCFST)

The LDAP password (parameter identifier: MQCA_LDAP_PASSWORD).

The maximum length is MQ_LDAP_PASSWORD_LENGTH.

LDAPUserName (MQCFST)

The LDAP user name (parameter identifier: MQCA_LDAP_USER_NAME).

On platforms other than z/OS, the maximum length is MQ_DISTINGUISHED_NAME_LENGTH. On z/OS, it is MQ_SHORT_DNAME_LENGTH.

QSGDisposition (MQCFIN)

Disposition of the object within the group (parameter identifier: MQIA_QSG_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves). The value can be:

QSGDisposition	Change	Copy, Create
MQQSGD_COPY	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameter MQQSGD_COPY. Any object residing in the shared repository, or any object defined using a command that had the parameter MQQSGD_Q_MGR, is not affected by this command.	The object is defined on the page set of the queue manager that executes the command using the MQQSGD_GROUP object of the same name as the <i>ToAuthInfoName</i> object (for Copy) or the <i>AuthInfoName</i> object (for Create).

QSGDisposition	Change	Copy, Create
MQQSGD_GROUP	<p>The object definition resides in the shared repository. The object was defined using a command that had the parameter MQQSGD_GROUP. Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command.</p> <p>If the command is successful, the following MQSC command is generated and sent to all active queue managers in the queue-sharing group so that they make or refresh local copies on page set zero:</p> <pre>DEFINE AUTHINFO(name) REPLACE QSGDISP(COPY)</pre> <p>The Change for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.</p>	<p>The object definition resides in the shared repository. This is allowed only if the queue manager is in a queue-sharing group.</p> <p>If the definition is successful, the following MQSC command is generated and sent to all active queue managers in the queue-sharing group so that they make or refresh local copies on page set zero:</p> <pre>DEFINE AUTHINFO(name) REPLACE QSGDISP(COPY)</pre> <p>The Copy or Create for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.</p>
MQQSGD_PRIVATE	The object resides on the page set of the queue manager that executes the command, and was defined with MQQSGD_Q_MGR, or MQQSGD_COPY. Any object residing in the shared repository is unaffected.	Not permitted.
MQQSGD_Q_MGR	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameter MQQSGD_Q_MGR. Any object residing in the shared repository, or any local copy of such an object, is not affected by this command. This is the default value.	The object is defined on the page set of the queue manager that executes the command. This is the default value.

Replace (MQCFIN)

Replace attributes (parameter identifier: MQIACF_REPLACE).

If an Authentication Information object with the same name as AuthInfoName or ToAuthInfoName already exists, this specifies whether it is to be replaced. The value can be:

MQRP_YES

Replace existing definition

MQRP_NO

Do not replace existing definition

Change, Copy, and Create CF Structure

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

Note: These commands are supported only on z/OS when the queue manager is a member of a queue-sharing group.

The Change CF Structure (MQCMD_CHANGE_CF_STRUC) command changes the specified attributes in a CF application structure. For any optional parameters that are omitted, the value does not change.

The Copy CF Structure (MQCMD_COPY_CF_STRUC) command creates a new CF application structure using, for attributes not specified in the command, the attribute values of an existing CF application structure.

The Create CF Structure (MQCMD_CREATE_CF_STRUC) command creates a CF application structure. Any attributes that are not defined explicitly are set to the default values on the destination queue manager.

Required parameters (Change and Create CF Structure):

CFStrucName

Required parameters (Copy CF Structure):

FromCFStrucName, ToCFStrucName

Optional parameters:

CFLevel, CFStrucDesc, Recovery, Replace

Required parameters (Change and Create CF Structure)

***CFStrucName* (MQCFST)**

The name of the CF application structure whose backup and recovery parameters you want to define (parameter identifier: MQCA_CF_STRUC_NAME).

The maximum length of the string is MQ_CF_STRUC_NAME_LENGTH.

Required parameters (Copy CF Structure)

***FromCFStrucName* (MQCFST)**

The name of the CF application structure to be copied from (parameter identifier: MQCACF_FROM_CF_STRUC_NAME).

The maximum length of the string is MQ_CF_STRUC_NAME_LENGTH.

***ToCFStrucName* (MQCFST)**

The name of the CF application structure to copy to (parameter identifier: MQCACF_TO_CF_STRUC_NAME).

The maximum length of the string is MQ_CF_STRUC_NAME_LENGTH.

Optional parameters

***CFLevel* (MQCFIN)**

The functional capability level for this CF application structure (parameter identifier: MQIA_CF_LEVEL).

Specifies the functional capability level for the CF application structure. The value can be:

- 1 A CF structure that can be "auto-created" by a queue manager at command level 520.
- 2 A CF structure at command level 520 that can only be created or deleted by a queue manager at command level 530 or greater.
- 3

A CF structure at command level 530. This *CFLevel* is required if you want to use persistent messages on shared queues, or for message grouping, or both. This is the default *CFLevel* for queue managers at command level 600.

You can only increase the value of *CFLevel* to 3 if all the queue managers in the queue-sharing group are at command level 530 or greater - this is to ensure that there are no latent command level 520 connections to queues referencing the CF structure.

You can only decrease the value of *CFLevel* from 3 if all the queues that reference the CF structure are both empty (have no messages or uncommitted activity) and closed.

4

This *CFLevel* supports all the *CFLevel* (3) functions. *CFLevel* (4) allows queues defined with CF structures at this level to have messages with a length greater than 63 KB.

Only a queue manager with a command level of 600 can connect to a CF structure at *CFLevel* (4).

You can only increase the value of *CFLevel* to 4 if all the queue managers in the queue-sharing group are at command level 600 or greater.

You can only decrease the value of *CFLevel* from 4 if all the queues that reference the CF structure are both empty (have no messages or uncommitted activity) and closed.

CFStrucDesc (MQCFST)

The description of the CF structure (parameter identifier: MQCA_CF_STRUC_DESC).

The maximum length is MQ_CF_STRUC_DESC_LENGTH.

Recovery (MQCFIN)

Recovery (parameter identifier: MQIA_CF_RECOVER).

Specifies whether CF recovery is supported for the application structure. The value can be:

MQCFR_YES

Recovery is supported.

MQCFR_NO

Recovery is not supported.

Replace (MQCFIN)

Replace attributes (parameter identifier: MQIACF_REPLACE).

If a CF structure definition with the same name as *ToCFStrucName* already exists, this specifies whether it is to be replaced. The value can be:

MQRP_YES

Replace existing definition.

MQRP_NO

Do not replace existing definition.

Change, Copy and Create Channel

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Change Channel (MQCMD_CHANGE_CHANNEL) command changes the specified attributes in a channel definition. For any optional parameters that are omitted, the value does not change.

The Copy Channel (MQCMD_COPY_CHANNEL) command creates a new channel definition using, for attributes not specified in the command, the attribute values of an existing channel definition.

The Create Channel (MQCMD_CREATE_CHANNEL) command creates a WebSphere MQ channel definition. Any attributes that are not defined explicitly are set to the default values on the destination queue manager. If a system default channel exists for the type of channel being created, the default values are taken from there.

Table 3 shows the parameters that are applicable to each type of channel.

Table 3. Change, Copy, Create Channel parameters

Parameter	Sender	Server	Receiver	Requester	Client conn	Server conn	Cluster sender	Cluster receiver
<i>BatchHeartBeat</i>	X	X					X	X
<i>BatchInterval</i>	X	X					X	X
<i>BatchSize</i>	X	X	X	X			X	X
<i>ChannelDesc</i>	X	X	X	X	X	X	X	X
<i>ChannelMonitoring</i>	X	X	X	X		X	X	X
<i>ChannelStatistics</i>	X	X	X	X			X	X
<i>ChannelName</i> ¹	X	X	X	X	X	X	X	X
<i>ChannelType</i> ³	X	X	X	X	X	X	X	X
<i>ClusterName</i>							X	X
<i>ClusterNameist</i>							X	X
<i>CLWLChannelPriority</i>							X	X
<i>CLWLChannelRank</i>							X	X
<i>CLWLChannelWeight</i>							X	X
<i>CommandScope</i>	X	X	X	X	X	X	X	X
<i>ConnectionName</i>	X	X		X	X		X	X
<i>DataConversion</i>	X	X		X	X		X	X
<i>DiscInterval</i>	X	X				X	X	X
<i>FromChannelName</i> ²	X	X	X	X	X	X	X	X
<i>HeaderCompression</i>	X	X	X	X	X	X	X	X
<i>HeartBeatInterval</i>	X	X	X	X	X	X	X	X
<i>KeepAliveInterval</i>	X	X	X	X	X	X	X	X

Table 3. Change, Copy, Create Channel parameters (continued)

Parameter	Sender	Server	Receiver	Requester	Client conn	Server conn	Cluster sender	Cluster receiver
<i>LocalAddress</i>	X	X		X	X		X	X
<i>LongRetryCount</i>	X	X					X	X
<i>LongRetryInterval</i>	X	X					X	X
<i>MaxMsgLength</i>	X	X	X	X	X	X	X	X
<i>MCAName</i>	X	X		X			X	
<i>MCAType</i>	X	X		X			X	X
<i>MCAUserIdentifer</i>	X	X	X	X		X	X	X
<i>MessageCompression</i>	X	X	X	X	X	X	X	X
<i>ModeName</i>	X	X		X	X		X	X
<i>MsgExit</i>	X	X	X	X			X	X
<i>MsgRetryCount</i>			X	X				X
<i>MsgRetryExit</i>			X	X				X
<i>MsgRetryInterval</i>			X	X				X
<i>MsgRetryUserData</i>			X	X				X
<i>MsgUserData</i>	X	X	X	X			X	X
<i>NetworkPriority</i>								X
<i>NonPersistentMsgSpeed</i>	X	X	X	X			X	X
<i>Password</i>	X	X		X	X		X	
<i>PutAuthority</i>			X	X		X		X
<i>QMgrName</i>					X			
<i>QSGDisposition</i>	X	X	X	X	X	X	X	X
<i>ReceiveExit</i>	X	X	X	X	X	X	X	X
<i>ReceiveUserData</i>	X	X	X	X	X	X	X	X
<i>Replace</i>	X	X	X	X	X	X	X	X
<i>SecurityExit</i>	X	X	X	X	X	X	X	X
<i>SecurityUserData</i>	X	X	X	X	X	X	X	X
<i>SendExit</i>	X	X	X	X	X	X	X	X
<i>SendUserData</i>	X	X	X	X	X	X	X	X
<i>SeqNumberWrap</i>	X	X	X	X			X	X
<i>ShortRetryCount</i>	X	X					X	X
<i>ShortRetryInterval</i>	X	X					X	X
<i>SSLCipherSpec</i>	X	X	X	X	X	X	X	X
<i>SSLClientAuth</i>			X	X		X		X
<i>SSLPeerName</i>	X	X	X	X	X	X	X	X
<i>ToChannelName</i> ²	X	X	X	X	X	X	X	X
<i>TpName</i>	X	X		X	X	X	X	X
<i>TransportType</i>	X	X	X	X	X	X	X	X
<i>UserIdentifier</i>	X	X		X	X		X	
<i>XmitQName</i>	X	X						

Table 3. Change, Copy, Create Channel parameters (continued)

Parameter	Sender	Server	Receiver	Requester	Client conn	Server conn	Cluster sender	Cluster receiver
Note:								
1.	Required parameter on Change and Create Channel commands							
2.	Required parameter on Copy Channel command							
3.	Required parameter on Change, Create, and Copy Channel commands							

Required parameters (Change, Create Channel)

ChannelName (MQCFST)

Channel name (parameter identifier: MQCACH_CHANNEL_NAME).

Specifies the name of the channel definition to be changed, or created

The maximum length of the string is MQ_CHANNEL_NAME_LENGTH.

This parameter is required on all types of channel; on a CLUSSDR it can be different from on the other channel types. If your convention for naming channels includes the name of the queue manager, you can make a CLUSSDR definition using the +QMNAME+ construction, and WebSphere MQ substitutes the correct repository queue manager name in place of +QMNAME+. This facility applies to AIX, HP-UX, Linux, i5/OS, Solaris, and Windows only. See *WebSphere MQ Queue Manager Clusters* for more details.

ChannelType (MQCFIN)

Channel type (parameter identifier: MQIACH_CHANNEL_TYPE).

Specifies the type of the channel being changed, copied, or created. The value can be:

MQCHT_SENDER

Sender.

MQCHT_SERVER

Server.

MQCHT_RECEIVER

Receiver.

MQCHT_REQUESTER

Requester.

MQCHT_SVRCONN

Server-connection (for use by clients).

MQCHT_CLNTCONN

Client connection.

MQCHT_CLUSRCVR

Cluster-receiver.

MQCHTCLUSSDR

Cluster-sender.

Required parameters (Copy Channel)

FromChannelName (MQCFST)

From channel name (parameter identifier: MQCACF_FROM_CHANNEL_NAME).

The name of the existing channel definition that contains values for the attributes that are not specified in this command.

On z/OS, the queue manager searches for an object with the name you specify and a disposition of MQQSGD_Q_MGR or MQQSGD_COPY to copy from. This parameter is ignored if a value of MQQSGD_COPY is specified for *QSGDisposition*. In this case, an object with the name specified by *ToChannelName* and the disposition MQQSGD_GROUP is searched for to copy from.

The maximum length of the string is MQ_CHANNEL_NAME_LENGTH.

ChannelType (MQCFIN)

Channel type (parameter identifier: MQIACH_CHANNEL_TYPE).

Specifies the type of the channel being changed, copied, or created. The value can be:

MQCHT_SENDER
Sender.

MQCHT_SERVER
Server.

MQCHT_RECEIVER
Receiver.

MQCHT_REQUESTER
Requester.

MQCHT_SVRCONN
Server-connection (for use by clients).

MQCHT_CLNTCONN
Client connection.

MQCHT_CLUSRCVR
Cluster-receiver.

MQCHTCLUSSDR
Cluster-sender.

ToChannelName (MQCFST)

To channel name (parameter identifier: MQCACF_TO_CHANNEL_NAME).

The name of the new channel definition.

The maximum length of the string is MQ_CHANNEL_NAME_LENGTH.

Channel names must be unique; if a channel definition with this name already exists, the value of *Replace* must be MQRP_YES. The channel type of the existing channel definition must be the same as the channel type of the new channel definition otherwise it cannot be replaced.

Optional parameters

BatchHeartbeat (MQCFIN)

The batch heartbeat interval (parameter identifier: MQIACH_BATCH_HB).

Batch heartbeating allows sender-type channels to determine whether the remote channel instance is still active, before going in-doubt. The value can be between 0 and 999999. A value of 0 indicates that batch heartbeating is not to be used. Batch heartbeat is measured in milliseconds.

This parameter is valid only for *ChannelType* values of MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR, or MQCHT_CLUSRCVR.

BatchInterval (MQCFIN)

Batch interval (parameter identifier: MQIACH_BATCH_INTERVAL).

This is the approximate time in milliseconds that a channel will keep a batch open, if fewer than *BatchSize* messages have been transmitted in the current batch.

This parameter is supported in the following environments: AIX, HP-UX, i5/OS, Solaris, Linux and z/OS.

If *BatchInterval* is greater than zero, the batch is terminated by whichever of the following occurs first:

- *BatchSize* messages have been sent, or
- *BatchInterval* milliseconds have elapsed since the start of the batch.

If *BatchInterval* is zero, the batch is terminated by whichever of the following occurs first:

- *BatchSize* messages have been sent, or
- the transmission queue becomes empty.

BatchInterval must be in the range zero through 999 999 999.

This parameter applies only to channels with a *ChannelType* of: MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR, or MQCHT_CLUSRCVR.

BatchSize (MQCFIN)

Batch size (parameter identifier: MQIACH_BATCH_SIZE).

The maximum number of messages that should be sent down a channel before a checkpoint is taken.

The batch size which is actually used is the lowest of the following:

- The *BatchSize* of the sending channel
- The *BatchSize* of the receiving channel
- The maximum number of uncommitted messages at the sending queue manager
- The maximum number of uncommitted messages at the receiving queue manager

The maximum number of uncommitted messages is specified by the *MaxUncommittedMsgs* parameter of the Change Queue Manager command.

Specify a value in the range 1-9999.

This parameter is not valid for channels with a *ChannelType* of MQCHT_SVRCONN or MQCHT_CLNTCONN.

ChannelDesc (MQCFST)

Channel description (parameter identifier: MQCACH_DESC).

The maximum length of the string is MQ_CHANNEL_DESC_LENGTH.

Use characters from the character set, identified by the coded character set identifier (CCSID) for the message queue manager on which the command is executing, to ensure that the text is translated correctly.

ChannelMonitoring (MQCFIN)

Online monitoring data collection (parameter identifier: MQIA_MONITORING_CHANNEL).

Specifies whether online monitoring data is to be collected and, if so, the rate at which the data is collected. The value can be:

MQMON_OFF

Online monitoring data collection is turned off for this channel.

MQMON_Q_MGR

The value of the queue manager's *ChannelMonitoring* parameter is inherited by the channel.

MQMON_LOW

If the value of the queue manager's *ChannelMonitoring* parameter is not MQMON_NONE, online monitoring data collection is turned on, with a low rate of data collection, for this channel.

MQMON_MEDIUM

If the value of the queue manager's *ChannelMonitoring* parameter is not MQMON_NONE, online monitoring data collection is turned on, with a moderate rate of data collection, for this channel.

MQMON_HIGH

If the value of the queue manager's *ChannelMonitoring* parameter is not MQMON_NONE, online monitoring data collection is turned on, with a high rate of data collection, for this channel.

ChannelStatistics (MQCFIN)

Statistics data collection (parameter identifier: MQIA_STATISTICS_CHANNEL).

Specifies whether statistics data is to be collected and, if so, the rate at which the data is collected. The value can be:

MQMON_OFF

Statistics data collection is turned off for this channel.

MQMON_Q_MGR

The value of the queue manager's *ChannelStatistics* parameter is inherited by the channel.

MQMON_LOW

If the value of the queue manager's *ChannelStatistics* parameter is not MQMON_NONE, online monitoring data collection is turned on, with a low rate of data collection, for this channel.

MQMON_MEDIUM

If the value of the queue manager's *ChannelStatistics* parameter is not MQMON_NONE, online monitoring data collection is turned on, with a moderate rate of data collection, for this channel.

MQMON_HIGH

If the value of the queue manager's *ChannelStatistics* parameter is not MQMON_NONE, online monitoring data collection is turned on, with a high rate of data collection, for this channel.

This parameter is valid only on AIX, HP-UX, Linux, i5/OS, Solaris, and Windows.

ClusterName (MQCFST)

Cluster name (parameter identifier: MQCA_CLUSTER_NAME).

The name of the cluster to which the channel belongs.

This parameter applies only to channels with a *ChannelType* of:

- MQCHT_CLUSSDR
- MQCHT_CLUSRCVR

Only one of the values of *ClusterName* and *ClusterNamelist* can be nonblank; the other must be blank.

The maximum length of the string is MQ_CLUSTER_NAME_LENGTH.

ClusterNamelist (MQCFST)

Cluster namelist (parameter identifier: MQCA_CLUSTER_NAMELIST).

The name, of the namelist, that specifies a list of clusters to which the channel belongs.

This parameter applies only to channels with a *ChannelType* of:

- MQCHT_CLUSSDR
- MQCHT_CLUSRCVR

Only one of the values of *ClusterName* and *ClusterNamelist* can be nonblank; the other must be blank.

CLWLChannelPriority (MQCFIN)

Channel priority for the purposes of cluster workload distribution (parameter identifier: MQIACH_CLWL_CHANNEL_PRIORITY).

Specify a value in the range zero through 9 where zero is the lowest priority and 9 is the highest.

This parameter applies only to channels with a *ChannelType* of:

- MQCHT_CLUSSDR
- MQCHT_CLUSRCVR

For more information about this parameter, see *WebSphere MQ Queue Manager Clusters*.

CLWLChannelRank (MQCFIN)

Channel rank for the purposes of cluster workload distribution (parameter identifier: MQIACH_CLWL_CHANNEL_RANK).

Specify a value in the range zero through 9 where zero is the lowest priority and 9 is the highest.

This parameter applies only to channels with a *ChannelType* of:

- MQCHT_CLUSSDR
- MQCHT_CLUSRCVR

For more information about this parameter, see *WebSphere MQ Queue Manager Clusters*.

CLWLChannelWeight (MQCFIN)

Channel weighting for the purposes of cluster workload distribution (parameter identifier: MQIACH_CLWL_CHANNEL_WEIGHT).

Specify a weighting for the channel for use in workload management. Specify a value in the range 1 through 99 where 1 is the lowest priority and 99 is the highest.

This parameter applies only to channels with a *ChannelType* of:

- MQCHT_CLUSSDR
- MQCHT_CLUSRCVR

For more information about this parameter, see *WebSphere MQ Queue Manager Clusters*.

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

ConnectionName (MQCFST)

Connection name (parameter identifier: MQCACH_CONNECTION_NAME).

On platforms other than z/OS, the maximum length of the string is MQ_CONN_NAME_LENGTH. On z/OS, it is MQ_LOCAL_ADDRESS_LENGTH.

Specify the name of the machine as required for the stated *TransportType*:

- For MQXPT_LU62 on i5/OS, and UNIX systems, specify the name of the CPI-C communications side object. On Windows specify the CPI-C symbolic destination name.

On z/OS, there are two forms in which to specify the value:

Logical unit name

The logical unit information for the queue manager, comprising the logical unit name, TP name, and optional mode name. This can be specified in one of 3 forms:

Form	Example
luname	IGY12355
luname/TPname	IGY12345/APING
luname/TPname/modename	IGY12345/APINGD/#INTER

For the first form, the TP name and mode name must be specified for the *TpName* and *ModeName* parameters; otherwise these parameters must be blank.

Note: For client-connection channels, only the first form is allowed.

Symbolic name

The symbolic destination name for the logical unit information for

the queue manager, as defined in the side information data set. The *TpName* and *ModeName* parameters must be blank.

Note: For cluster-receiver channels, the side information is on the other queue managers in the cluster. Alternatively, in this case it can be a name that a channel auto-definition exit can resolve into the appropriate logical unit information for the local queue manager.

The specified or implied LU name can be that of a VTAM® generic resources group.

- For MQXPT_TCP you can specify the host name or the network address of the remote machine.

On z/OS, the connection name can include the IP_name of a z/OS dynamic DNS group or a network dispatcher input port. Do **not** include this for channels with a *ChannelType* value of MQCHT_CLUSSDR.

On a MQCHT_CLUSRCVR channel, the *ConnectionName* parameter is optional. On AIX, HP-UX, Linux, i5/OS, Solaris, or Windows MQCHT_CLUSRCVR channel, if you leave *ConnectionName* blank, WebSphere MQ generates a *ConnectionName* for you, assuming the default port and using the current IP address of the system.

- For MQXPT_NETBIOS specify the NetBIOS station name.
- For MQXPT_SPX specify the 4 byte network address, the 6 byte node address, and the 2 byte socket number. These should be entered in hexadecimal, with a period separating the network and node addresses. The socket number should be enclosed in brackets, for example:

0a0b0c0d.804abcde23a1(5e86)

If the socket number is omitted, the WebSphere MQ default value (5e86 hex) is assumed.

This parameter is valid only for *ChannelType* values of MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER, MQCHT_CLNTCONN, MQCHT_CLUSSDR, or MQCHT_CLUSRCVR.

Note: If you are using clustering between IPv6-only and IPv4-only queue managers, do not specify an IPv6 network address as the *ConnectionName* for cluster-receiver channels. A queue manager that is capable only of IPv4 communication is unable to start a cluster sender channel definition that specifies the *ConnectionName* in IPv6 hexadecimal form. Consider, instead, using hostnames in a heterogeneous IP environment.

DataConversion (MQCFIN)

Whether sender should convert application data (parameter identifier: MQIACH_DATA_CONVERSION).

This parameter is valid only for *ChannelType* values of MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR, or MQCHT_CLUSRCVR.

The value can be:

MQCDC_NO_SENDER_CONVERSION

No conversion by sender.

MQCDC_SENDER_CONVERSION

Conversion by sender.

DiscInterval (MQCFIN)

Disconnection interval (parameter identifier: MQIACH_DISC_INTERVAL).

This defines the maximum number of seconds that the channel waits for messages to be put on a transmission queue before terminating the channel. A value of zero causes the message channel agent to wait indefinitely.

Specify a value in the range 0 through 999 999.

This parameter is valid only for *ChannelType* values of MQCHT_SENDER, MQCHT_SERVER, MQCHT_SVRCONN (on z/OS only), MQCHT_CLUSSDR, or MQCHT_CLUSRCVR.

For server-connection channels on z/OS using the TCP protocol, this is the minimum time in seconds for which the server-connection channel instance remains active without any communication from its partner client. A value of zero disables this disconnect processing. The server-connection inactivity interval only applies between MQ API calls from a client, so no client is disconnected during an extended MQGET with wait call. This attribute is ignored for server-connection channels using protocols other than TCP.

HeaderCompression (MQCFIL)

Header data compression techniques supported by the channel (parameter identifier: MQIACH_HDR_COMPRESSION).

The list of header data compression techniques supported by the channel. For sender, server, cluster-sender, cluster-receiver, and client-connection channels, the values specified are in order of preference with the first compression technique supported by the remote end of the channel being used.

The channel's mutually supported compression techniques are passed to the sending channel's message exit where the compression technique used can be altered on a per message basis. Compression alters the data passed to send and receive exits.

Specify one or more of:

MQCOMPRESS_NONE

No header data compression is performed. This is the default value.

MQCOMPRESS_SYSTEM

Header data compression is performed.

HeartbeatInterval (MQCFIN)

Heartbeat interval (parameter identifier: MQIACH_HB_INTERVAL).

The interpretation of this parameter depends on the channel type, as follows:

- For a channel type of MQCHT_SENDER, MQCHT_SERVER, MQCHT_RECEIVER, MQCHT_REQUESTER, MQCHT_CLUSSDR, or MQCHT_CLUSRCVR, this is the time in seconds between heartbeat flows passed from the sending MCA when there are no messages on the transmission queue. This gives the receiving MCA the opportunity to quiesce the channel. To be useful, *HeartbeatInterval* should be significantly less than *DiscInterval*. However, the only check is that the value is within the permitted range.

This type of heartbeat is supported in the following environments: AIX, HP-UX, i5/OS, Solaris, Windows, and z/OS.

- For a channel type of MQCHT_CLNTCONN or MQCHT_SVRCONN, this is the time in seconds between heartbeat flows passed from the server MCA when that MCA has issued an MQGET call with the MQGMO_WAIT option

on behalf of a client application. This allows the server MCA to handle situations where the client connection fails during an MQGET with MQGMO_WAIT.

This type of heartbeat is supported in the following environments: AIX, HP-UX, i5/OS, Solaris, Windows, Linux and z/OS.

The value must be in the range 0 through 999 999. A value of 0 means that no heartbeat exchange occurs. The value that is actually used is the larger of the values specified at the sending side and receiving side.

KeepAliveInterval (MQCFIN)

KeepAlive interval (parameter identifier: MQIACH_KEEP_ALIVE_INTERVAL).

Specifies the value passed to the communications stack for KeepAlive timing for the channel.

For this attribute to have any effect, TCP/IP keepalive must be enabled. On z/OS, you do this by issuing the Change Queue Manager with a value of MQTCPKEEP in the *TCPKeepAlive* parameter; if the *TCPKeepAlive* queue manager parameter has a value of MQTCPKEEP_NO, the value is ignored and the KeepAlive facility is not used.. On other platforms, TCP/IP keepalive is enabled when the KEEPALIVE=YES parameter is specified in the TCP stanza in the distributed queuing configuration file, qm.ini, or through the WebSphere MQ Explorer. Keepalive must also be switched on within TCP/IP itself, using the TCP profile configuration data set.

Although this parameter is available on all platforms, its setting is implemented only on z/OS. On platforms other than z/OS, you can access and modify the parameter, but it is only stored and forwarded; there is no functional implementation of the parameter. This is useful in a clustered environment where a value set in a cluster-receiver channel definition on Solaris, for example, flows to (and is implemented by) z/OS queue managers that are in, or join, the cluster.

Specify either:

integer

The KeepAlive interval to be used, in seconds, in the range 0 through 99 999. If you specify a value of 0, the value used is that specified by the INTERVAL statement in the TCP profile configuration data set.

MQKAI_AUTO

The KeepAlive interval is calculated based upon the negotiated heartbeat value as follows:

- If the negotiated *HeartbeatInterval* is greater than zero, KeepAlive interval is set to that value plus 60 seconds.
- If the negotiated *HeartbeatInterval* is zero, the value used is that specified by the INTERVAL statement in the TCP profile configuration data set.

On platforms other than z/OS, if you need the functionality provided by the *KeepAliveInterval* parameter, use the *HeartBeatInterval* parameter.

LocalAddress (MQCFST)

Local communications address for the channel (parameter identifier: MQCACH_LOCAL_ADDRESS).

The maximum length of the string is MQ_LOCAL_ADDRESS_LENGTH.

The value that you specify depends on the transport type (*TransportType*) to be used:

TCP/IP

The value is the optional IP address and optional port or port range to be used for outbound TCP/IP communications. The format for this information is as follows:

[ip-addr][(low-port[,high-port])]

where ip-addr is specified in IPv4 dotted decimal, IPv6 hexadecimal notation, or alphanumeric form, and low-port and high-port are port numbers enclosed in parentheses. All are optional.

All Others

The value is ignored; no error is diagnosed.

Use this parameter if you want a channel to use a particular IP address, port, or port range for outbound communications. This is useful when a machine is connected to multiple networks with different IP addresses.

Examples of use

Value	Meaning
9.20.4.98	Channel binds to this address locally
9.20.4.98 (1000)	Channel binds to this address and port 1000 locally
9.20.4.98 (1000,2000)	Channel binds to this address and uses a port in the range 1000 to 2000 locally
(1000)	Channel binds to port 1000 locally
(1000,2000)	Channel binds to a port in the range 1000 to 2000 locally

This parameter is valid for the following channel types:

- MQCHT_SENDER
- MQCHT_SERVER
- MQCHT_REQUESTER
- MQCHT_CLNTCONN
- MQCHT_CLUSRCVR
- MQCHTCLUSSDR

Note:

- Do not confuse this parameter with *ConnectionName*. The *LocalAddress* parameter specifies the characteristics of the local communications; the *ConnectionName* parameter specifies how to reach a remote queue manager.

LongRetryCount (MQCFIN)

Long retry count (parameter identifier: MQIACH_LONG_RETRY).

When a sender or server channel is attempting to connect to the remote machine, and the count specified by *ShortRetryCount* has been exhausted, this specifies the maximum number of further attempts that are made to connect to the remote machine, at intervals specified by *LongRetryInterval*.

If this count is also exhausted without success, an error is logged to the operator, and the channel is stopped. The channel must subsequently be restarted with a command (it is not started automatically by the channel

initiator), and it then makes only one attempt to connect, as it is assumed that the problem has now been cleared by the administrator. The retry sequence is not carried out again until after the channel has successfully connected.

Specify a value in the range 0 through 999 999 999.

This parameter is valid only for *ChannelType* values of MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR, or MQCHT_CLUSRCVR.

LongRetryInterval (MQCFIN)

Long timer (parameter identifier: MQIACH_LONG_TIMER).

Specifies the long retry wait interval for a sender or server channel that is started automatically by the channel initiator. It defines the interval in seconds between attempts to establish a connection to the remote machine, after the count specified by *ShortRetryCount* has been exhausted.

The time is approximate; zero means that another connection attempt is made as soon as possible.

Specify a value in the range 0 through 999 999. Values exceeding this are treated as 999 999.

This parameter is valid only for *ChannelType* values of MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR, or MQCHT_CLUSRCVR.

MaxMsgLength (MQCFIN)

Maximum message length (parameter identifier: MQIACH_MAX_MSG_LENGTH).

Specifies the maximum message length that can be transmitted on the channel. This is compared with the value for the remote channel and the actual maximum is the lower of the two values.

The value zero means the maximum message length for the queue manager.

The lower limit for this parameter is 0. The upper limit depends on the environment:

- On AIX, HP OpenVMS, Compaq NonStop Kernel, Linux, HP-UX, i5/OS, Solaris, Windows, and z/OS the maximum message length is 100 MB (104 857 600 bytes).
- On UNIX systems not listed above, the maximum message length is 4 MB (4 194 304 bytes).

MCAName (MQCFST)

Message channel agent name (parameter identifier: MQCACH_MCA_NAME).

This is reserved, and if specified can be set only to blanks.

The maximum length of the string is MQ_MCA_NAME_LENGTH.

This parameter is valid only for *ChannelType* values of MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER, MQCHT_CLUSSDR, or MQCHT_CLUSRCVR.

MCAType (MQCFIN)

Message channel agent type (parameter identifier: MQIACH_MCA_TYPE).

Specifies the type of the message channel agent program.

On AIX, HP-UX, i5/OS, Solaris, Windows and Linux, this parameter is valid only for *ChannelType* values of MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER, or MQCHT_CLUSSDR.

On z/OS, this parameter is valid only for a *ChannelType* value of MQCHT_CLURCVR.

The value can be:

MQMCAT_PROCESS
Process.

MQMCAT_THREAD
Thread.

MCAUserIdentifier (MQCFST)

Message channel agent user identifier (parameter identifier: MQCACH_MCA_USER_ID).

If this is nonblank, it is the user identifier which is to be used by the message channel agent for authorization to access WebSphere MQ resources, including (if *PutAuthority* is MQPA_DEFAULT) authorization to put the message to the destination queue for receiver or requester channels.

If it is blank, the message channel agent uses its default user identifier.

This user identifier can be overridden by one supplied by a channel security exit.

This parameter is not valid for channels with a *ChannelType* of MQCHT_CLNTCONN.

The maximum length of the MCA user identifier depends on the environment in which the MCA is running. MQ_MCA_USER_ID_LENGTH gives the maximum length for the environment for which your application is running. MQ_MAX_MCA_USER_ID_LENGTH gives the maximum for all supported environments.

On Windows, you can optionally qualify a user identifier with the domain name in the following format:

user@domain

MessageCompression (MQCFIL)

Header data compression techniques supported by the channel (parameter identifier: MQIACH_MSG_COMPRESSION). The list of message data compression techniques supported by the channel. For sender, server, cluster-sender, cluster-receiver, and client-connection channels, the values specified are in order of preference with the first compression technique supported by the remote end of the channel being used.

The channel's mutually supported compression techniques are passed to the sending channel's message exit where the compression technique used can be altered on a per message basis. Compression will alter the data passed to send and receive exits.

Specify one or more of:

MQCOMPRESS_NONE

No message data compression is performed. This is the default value.

MQCOMPRESS_RLE

Message data compression is performed using run-length encoding.

MQCOMPRESS_ZLIBFAST

Message data compression is performed using ZLIB encoding with speed prioritized.

MQCOMPRESS_ZLIBHIGH

Message data compression is performed using ZLIB encoding with compression prioritized.

MQCOMPRESS_ANY

Any compression technique supported by the queue manager can be used. This is only valid for receiver, requester, and server-connection channels.

ModeName (MQCFST)

Mode name (parameter identifier: MQCACH_MODE_NAME).

This is the LU 6.2 mode name.

The maximum length of the string is MQ_MODE_NAME_LENGTH.

- On HP OpenVMS, i5/OS, Compaq NonStop Kernel, UNIX systems, and Windows, this can be set only to blanks. The actual name is taken instead from the CPI-C Communications Side Object or (on Windows) from the CPI-C symbolic destination name properties.

This parameter is valid only for channels with a *TransportType* of MQXPT_LU62. It is not valid for receiver or server-connection channels.

MsgExit (MQCFSL)

Message exit name (parameter identifier: MQCACH_MSG_EXIT_NAME).

If a nonblank name is defined, the exit is invoked immediately after a message has been retrieved from the transmission queue. The exit is given the entire application message and message descriptor for modification.

For channels with a channel type (*ChannelType*) of MQCHT_SVRCONN or MQCHT_CLNTCONN, this parameter is accepted but ignored, since message exits are not invoked for such channels.

The format of the string is the same as for *SecurityExit*.

The maximum length of the exit name depends on the environment in which the exit is running. MQ_EXIT_NAME_LENGTH gives the maximum length for the environment in which your application is running.

MQ_MAX_EXIT_NAME_LENGTH gives the maximum for all supported environments.

In the following environments, a list of exit names can be specified by using an MQCFSL structure instead of an MQCFST structure: AIX, HP-UX, i5/OS, Solaris, Windows, Linux, and z/OS.

- The exits are invoked in the order specified in the list.
- A list with only one name is equivalent to specifying a single name in an MQCFST structure.
- You cannot specify both a list (MQCFSL) and a single entry (MQCFST) structure for the same channel attribute.
- The total length of all of the exit names in the list (excluding trailing blanks in each name) must not exceed MQ_TOTAL_EXIT_NAME_LENGTH. An individual string must not exceed MQ_EXIT_NAME_LENGTH.
- On z/OS, you can specify the names of up to 8 exit programs.

MsgRetryCount (MQCFIN)

Message retry count (parameter identifier: MQIACH_MR_COUNT).

Specifies the number of times that a failing message should be retried.

Specify a value in the range 0 through 999 999 999.

This parameter is valid only for *ChannelType* values of MQCHT_RECEIVER, MQCHT_REQUESTER, or MQCHT_CLUSRCVR.

MsgRetryExit (**MQCFST**)

Message retry exit name (parameter identifier: MQCACH_MR_EXIT_NAME).

If a nonblank name is defined, the exit is invoked prior to performing a wait before retrying a failing message.

The format of the string is the same as for *SecurityExit*.

The maximum length of the exit name depends on the environment in which the exit is running. MQ_EXIT_NAME_LENGTH gives the maximum length for the environment in which your application is running.

MQ_MAX_EXIT_NAME_LENGTH gives the maximum for all supported environments.

This parameter is valid only for *ChannelType* values of MQCHT_RECEIVER, MQCHT_REQUESTER, or MQCHT_CLUSRCVR.

MsgRetryInterval (**MQCFIN**)

Message retry interval (parameter identifier: MQIACH_MR_INTERVAL).

Specifies the minimum time interval in milliseconds between retries of failing messages.

Specify a value in the range 0 through 999 999 999.

This parameter is valid only for *ChannelType* values of MQCHT_RECEIVER, MQCHT_REQUESTER, or MQCHT_CLUSRCVR.

MsgRetryUserData (**MQCFST**)

Message retry exit user data (parameter identifier: MQCACH_MR_EXIT_USER_DATA).

Specifies user data that is passed to the message retry exit.

The maximum length of the string is MQ_EXIT_DATA_LENGTH.

This parameter is valid only for *ChannelType* values of MQCHT_RECEIVER, MQCHT_REQUESTER, or MQCHT_CLUSRCVR.

MsgUserData (**MQCFSL**)

Message exit user data (parameter identifier: MQCACH_MSG_EXIT_USER_DATA).

Specifies user data that is passed to the message exit.

The maximum length of the string is MQ_EXIT_DATA_LENGTH.

For channels with a channel type (*ChannelType*) of MQCHT_SVRCONN or MQCHT_CLNTCONN, this parameter is accepted but ignored, since message exits are not invoked for such channels.

In the following environments, a list of exit user data strings can be specified by using an MQCFSL structure instead of an MQCFST structure: AIX, HP-UX, i5/OS, Solaris, Windows, Linux, and z/OS.

- Each exit user data string is passed to the exit at the same ordinal position in the *MsgExit* list.
- A list with only one name is equivalent to specifying a single name in an MQCFST structure.
- You cannot specify both a list (MQCFSL) and a single entry (MQCFST) structure for the same channel attribute.

- The total length of all of the exit user data in the list (excluding trailing blanks in each string) must not exceed MQ_TOTAL_EXIT_DATA_LENGTH. An individual string must not exceed MQ_EXIT_DATA_LENGTH.
- On z/OS, you can specify up to 8 strings.

NetworkPriority (MQCFIN)

Network priority (parameter identifier: MQIACH_NETWORK_PRIORITY).

The priority for the network connection. If there are multiple paths available, distributed queuing selects the path with the highest priority.

The value must be in the range 0 (lowest) through 9 (highest).

This parameter applies only to channels with a *ChannelType* of MQCHT_CLUSRCVR

NonPersistentMsgSpeed (MQCFIN)

Speed at which nonpersistent messages are to be sent (parameter identifier: MQIACH_NPM_SPEED).

This parameter is supported in the following environments: AIX, HP-UX, i5/OS, Solaris, Windows and Linux.

Specifying MQNPMS_FAST means that nonpersistent messages on a channel need not wait for a syncpoint before being made available for retrieval. The advantage of this is that nonpersistent messages become available for retrieval far more quickly. The disadvantage is that because they do not wait for a syncpoint, they might be lost if there is a transmission failure.

This parameter is valid only for *ChannelType* values of MQCHT_SENDER, MQCHT_SERVER, MQCHT_RECEIVER, MQCHT_REQUESTER, MQCHT_CLUSSDR, or MQCHT_CLUSRCVR. The value can be:

MQNPMS_NORMAL
Normal speed.

MQNPMS_FAST
Fast speed.

Password (MQCFST)

Password (parameter identifier: MQCACH_PASSWORD).

This is used by the message channel agent when attempting to initiate a secure SNA session with a remote message channel agent. On HP OpenVMS, i5/OS, Compaq NonStop Kernel, and UNIX systems, it is valid only for *ChannelType* values of MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER, MQCHT_CLNTCONN, or MQCHT_CLUSSDR. On z/OS, it is valid only for a *ChannelType* value of MQCHT_CLNTCONN.

The maximum length of the string is MQ_PASSWORD_LENGTH. However, only the first 10 characters are used.

PutAuthority (MQCFIN)

Put authority (parameter identifier: MQIACH_PUT_AUTHORITY).

Specifies whether the user identifier in the context information associated with a message should be used to establish authority to put the message on the destination queue.

This parameter is valid only for channels with a *ChannelType* value of MQCHT_RECEIVER, MQCHT_REQUESTER, MQCHT_CLUSRCVR, or, on z/OS only, MQCHT_SVRCONN.

The value can be:

MQPA_DEFAULT

Default user identifier is used.

MQPA_CONTEXT

Context user identifier is used. This value is not valid for channels of type MQCHT_SVRCONN.

MQPA_ALTERNATE_OR_MCA

The user ID from the *UserIdentifier* field of the message descriptor is used. Any user ID received from the network is not used. This value is supported only on z/OS and is not valid for channels of type MQCHT_SVRCONN.

MQPA_ONLY_MCA

The default user ID is used. Any user ID received from the network is not used. This value is supported only on z/OS.

***QMgrName* (MQCFST)**

Queue-manager name (parameter identifier: MQCA_Q_MGR_NAME).

For channels with a *ChannelType* of MQCHT_CLNTCONN, this is the name of a queue manager to which a client application can request connection.

For channels of other types, this parameter is not valid. The maximum length of the string is MQ_Q_MGR_NAME_LENGTH.

***QSGDisposition* (MQCFIN)**

Disposition of the object within the group (parameter identifier: MQIA_QSG_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves). The value can be:

QSGDisposition	Change	Copy, Create
MQQSGD_COPY	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameter MQQSGD_COPY. Any object residing in the shared repository, or any object defined using a command that had the parameters MQQSGD_Q_MGR, is not affected by this command.	The object is defined on the page set of the queue manager that executes the command using the MQQSGD_GROUP object of the same name as the <i>ToChannelName</i> object (for Copy) or <i>ChannelName</i> object (for Create).

QSGDisposition	Change	Copy, Create
MQQSGD_GROUP	<p>The object definition resides in the shared repository. The object was defined using a command that had the parameter MQQSGD_GROUP. Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command.</p> <p>If the command is successful, the following MQSC command is generated and sent to all active queue managers in the queue-sharing group to attempt to refresh local copies on page set zero:</p> <pre>DEFINE CHANNEL(channel-name) CHLTYPE(type) REPLACE QSGDISP(COPY)</pre> <p>The Change for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.</p>	<p>The object definition resides in the shared repository. This is allowed only if the queue manager is in a queue-sharing group.</p> <p>If the definition is successful, the following MQSC command is generated and sent to all active queue managers in the queue-sharing group to attempt to make or refresh local copies on page set zero:</p> <pre>DEFINE CHANNEL(channel-name) CHLTYPE(type) REPLACE QSGDISP(COPY)</pre> <p>The Copy or Create for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.</p>
MQQSGD_PRIVATE	The object resides on the page set of the queue manager that executes the command, and was defined with MQQSGD_Q_MGR or MQQSGD_COPY. Any object residing in the shared repository is unaffected.	Not permitted.
MQQSGD_Q_MGR	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameter MQQSGD_Q_MGR. Any object residing in the shared repository, or any local copy of such an object, is not affected by this command. This is the default value.	The object is defined on the page set of the queue manager that executes the command. This is the default value.

ReceiveExit (MQCFSL)

Receive exit name (parameter identifier: MQCACH_RCV_EXIT_NAME).

If a nonblank name is defined, the exit is invoked before data received from the network is processed. The complete transmission buffer is passed to the exit and the contents of the buffer can be modified as required.

The format of the string is the same as for *SecurityExit*.

The maximum length of the exit name depends on the environment in which the exit is running. MQ_EXIT_NAME_LENGTH gives the maximum length for the environment in which your application is running.

MQ_MAX_EXIT_NAME_LENGTH gives the maximum for all supported environments.

In the following environments, a list of exit names can be specified by using an MQCFSL structure instead of an MQCFST structure: AIX, HP-UX, i5/OS, Solaris, Windows, Linux, and z/OS.

- The exits are invoked in the order specified in the list.
- A list with only one name is equivalent to specifying a single name in an MQCFST structure.
- You cannot specify both a list (MQCFSL) and a single entry (MQCFST) structure for the same channel attribute.

- The total length of all of the exit names in the list (excluding trailing blanks in each name) must not exceed MQ_TOTAL_EXIT_NAME_LENGTH. An individual string must not exceed MQ_EXIT_NAME_LENGTH.
- On z/OS, you can specify the names of up to 8 exit programs.

ReceiveUserData (MQCFSL)

Receive exit user data (parameter identifier: MQCACH_RCV_EXIT_USER_DATA).

Specifies user data that is passed to the receive exit.

The maximum length of the string is MQ_EXIT_DATA_LENGTH.

In the following environments, a list of exit user data strings can be specified by using an MQCFSL structure instead of an MQCFST structure: AIX, HP-UX, i5/OS, Solaris, Windows, Linux, and z/OS.

- Each exit user data string is passed to the exit at the same ordinal position in the *ReceiveExit* list.
- A list with only one name is equivalent to specifying a single name in an MQCFST structure.
- You cannot specify both a list (MQCFSL) and a single entry (MQCFST) structure for the same channel attribute.
- The total length of all of the exit user data in the list (excluding trailing blanks in each string) must not exceed MQ_TOTAL_EXIT_DATA_LENGTH. An individual string must not exceed MQ_EXIT_DATA_LENGTH.
- On z/OS, you can specify up to 8 strings.

Replace (MQCFIN)

Replace channel definition (parameter identifier: MQIACF_REPLACE).

The value can be:

MQRP_YES

Replace existing definition.

If *ChannelType* is MQCHT_CLUSSDR, MQRP_YES can be specified only if the channel was created manually.

MQRP_NO

Do not replace existing definition.

SecurityExit (MQCFST)

Security exit name (parameter identifier: MQCACH_SEC_EXIT_NAME).

If a nonblank name is defined, the security exit is invoked at the following times:

- Immediately after establishing a channel.
Before any messages are transferred, the exit is given the opportunity to instigate security flows to validate connection authorization.
- Upon receipt of a response to a security message flow.
Any security message flows received from the remote processor on the remote machine are passed to the exit.

The exit is given the entire application message and message descriptor for modification.

The format of the string depends on the platform, as follows:

- On iSeries and UNIX systems, it is of the form
`libraryname(functionname)`

Note: On iSeries systems, the following form is also supported for compatibility with older releases:

```
progname libname
```

where *progname* occupies the first 10 characters, and *libname* the second 10 characters (both blank-padded to the right if necessary).

- On Windows, it is of the form
`dllname(functionname)`
where *dllname* is specified without the suffix “.DLL”.
- On HP OpenVMS, it is of the form
`imagename(functionname)`
- On z/OS, it is a load module name, maximum length 8 characters (128 characters are allowed for exit names for client-connection channels, subject to a maximum total length of 999).

The maximum length of the exit name depends on the environment in which the exit is running. `MQ_EXIT_NAME_LENGTH` gives the maximum length for the environment in which your application is running.

`MQ_MAX_EXIT_NAME_LENGTH` gives the maximum for all supported environments.

SecurityUserData (`MQCFST`)

Security exit user data (parameter identifier:
`MQCACH_SEC_EXIT_USER_DATA`).

Specifies user data that is passed to the security exit.

The maximum length of the string is `MQ_EXIT_DATA_LENGTH`.

SendExit (`MQCFSL`)

Send exit name (parameter identifier: `MQCACH_SEND_EXIT_NAME`).

If a nonblank name is defined, the exit is invoked immediately before data is sent out on the network. The exit is given the complete transmission buffer before it is transmitted; the contents of the buffer can be modified as required.

The format of the string is the same as for *SecurityExit*.

The maximum length of the exit name depends on the environment in which the exit is running. `MQ_EXIT_NAME_LENGTH` gives the maximum length for the environment in which your application is running.

`MQ_MAX_EXIT_NAME_LENGTH` gives the maximum for all supported environments.

In the following environments, a list of exit names can be specified by using an `MQCFSL` structure instead of an `MQCFST` structure: AIX, HP-UX, i5/OS, Solaris, Windows, Linux, and z/OS.

- The exits are invoked in the order specified in the list.
- A list with only one name is equivalent to specifying a single name in an `MQCFST` structure.
- You cannot specify both a list (`MQCFSL`) and a single entry (`MQCFST`) structure for the same channel attribute.
- The total length of all of the exit names in the list (excluding trailing blanks in each name) must not exceed `MQ_TOTAL_EXIT_NAME_LENGTH`. An individual string must not exceed `MQ_EXIT_NAME_LENGTH`.
- On z/OS, you can specify the names of up to 8 exit programs.

SendUserData (MQCFSL)

Send exit user data (parameter identifier: MQCACH_SEND_EXIT_USER_DATA).

Specifies user data that is passed to the send exit.

The maximum length of the string is MQ_EXIT_DATA_LENGTH.

In the following environments, a list of exit user data strings can be specified by using an MQCFSL structure instead of an MQCFST structure: AIX, HP-UX, i5/OS, Solaris, Windows, Linux, and z/OS.

- Each exit user data string is passed to the exit at the same ordinal position in the *SendExit* list.
- A list with only one name is equivalent to specifying a single name in an MQCFST structure.
- You cannot specify both a list (MQCFSL) and a single entry (MQCFST) structure for the same channel attribute.
- The total length of all of the exit user data in the list (excluding trailing blanks in each string) must not exceed MQ_TOTAL_EXIT_DATA_LENGTH. An individual string must not exceed MQ_EXIT_DATA_LENGTH.
- On z/OS, you can specify up to 8 strings.

SeqNumberWrap (MQCFIN)

Sequence wrap number (parameter identifier: MQIACH_SEQUENCE_NUMBER_WRAP).

Specifies the maximum message sequence number. When the maximum is reached, sequence numbers wrap to start again at 1.

The maximum message sequence number is not negotiable; the local and remote channels must wrap at the same number.

Specify a value in the range 100 through 999 999 999.

This parameter is not valid for channels with a *ChannelType* of MQCHT_SVRCONN or MQCHT_CLNTCONN.

ShortRetryCount (MQCFIN)

Short retry count (parameter identifier: MQIACH_SHORT_RETRY).

The maximum number of attempts that are made by a sender or server channel to establish a connection to the remote machine, at intervals specified by *ShortRetryInterval* before the (normally longer) *LongRetryCount* and *LongRetryInterval* are used.

Retry attempts are made if the channel fails to connect initially (whether it is started automatically by the channel initiator or by an explicit command), and also if the connection fails after the channel has successfully connected.

However, if the cause of the failure is such that retry is unlikely to be successful, retries are not attempted.

Specify a value in the range 0 through 999 999 999.

This parameter is valid only for *ChannelType* values of MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR, or MQCHT_CLUSRCVR.

ShortRetryInterval (MQCFIN)

Short timer (parameter identifier: MQIACH_SHORT_TIMER).

Specifies the short retry wait interval for a sender or server channel that is started automatically by the channel initiator. It defines the interval in seconds between attempts to establish a connection to the remote machine.

The time is approximate; zero means that another connection attempt is made as soon as possible.

Specify a value in the range 0 through 999 999. Values exceeding this are treated as 999 999.

This parameter is valid only for *ChannelType* values of MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR, or MQCHT_CLUSRCVR.

*SSL**CipherSpec* (MQCFST)

CipherSpec (parameter identifier: MQCACH_SSL_CIPHER_SPEC).

The length of the string is MQ_SSL_CIPHER_SPEC_LENGTH.

This parameter is valid only for channels with a transport type (TRPTYPE) of TCP. If the TRPTYPE is not TCP, the data is ignored and no error message is issued.

The SSLCIPH values must specify the same CipherSpec on both ends of the channel. For more information about working with CipherSpecs, see the *WebSphere MQ Security* book.

Specify the name of the CipherSpec that you are using. Alternatively, on i5/OS, and z/OS, you can specify the two-digit hexadecimal code.

The following table shows the CipherSpecs that can be used with WebSphere MQ SSL.

On i5/OS, installation of AC3 is a prerequisite of the use of SSL.

Table 4. CipherSpecs that can be used with WebSphere MQ SSL support

CipherSpec name	Hash algorithm	Encryption algorithm	Encryption bits	FIPS on Windows and UNIX platforms ¹
NULL_MD5 Note: Available on all platforms.	MD5	None	0	No
NULL_SHA Note: Available on all platforms	SHA-1	None	0	No
RC4_MD5_EXPORT Note: Available on all platforms	MD5	RC4	40	No
RC4_MD5_US Note: Available on all platforms	MD5	RC4	128	No
RC4_SHA_US Note: Available on all platforms	SHA-1	RC4	128	No
RC2_MD5_EXPORT Note: Available on all platforms	MD5	RC2	40	No
DES_SHA_EXPORT Note: Available on all platforms	SHA-1	DES	56	No
RC4_56_SHA_EXPORT1024 Note: <ol style="list-style-type: none">1. Not available for z/OS or i5/OS2. Specifies a 1024-bit handshake key size	SHA-1	RC4	56	No
DES_SHA_EXPORT1024 Note: <ol style="list-style-type: none">1. Not available for z/OS or i5/OS2. Specifies a 1024-bit handshake key size	SHA-1	DES	56	No

Table 4. CipherSpecs that can be used with WebSphere MQ SSL support (continued)

CipherSpec name	Hash algorithm	Encryption algorithm	Encryption bits	FIPS on Windows and UNIX platforms ¹
TRIPLE_DES_SHA_US Note: Not available for i5/OS	SHA-1	3DES	168	No
TLS_RSA_WITH_AES_128_CBC_SHA Note: 1. Not available for i5/OS 2. The protocol used is TLS rather than SSL	SHA-1	AES	128	Yes
TLS_RSA_WITH_AES_256_CBC_SHA Note: 1. Not available for i5/OS 2. The protocol used is TLS rather than SSL	SHA-1	AES	256	Yes
AES_SHA_US Note: Available on i5/OS TM only	SHA-1	AES	128	No
TLS_RSA_WITH DES_CBC_SHA Note: 1. Not available for z/OS or i5/OS 2. The protocol used is TLS rather than SSL	SHA-1	DES	56	Yes
TLS_RSA_WITH_3DES_EDE_CBC_SHA Note: 1. Not available for z/OS or i5/OS 2. The protocol used is TLS rather than SSL	SHA-1	3DES	168	Yes
FIPS_WITH DES_CBC_SHA Note: Available only on Windows and UNIX platforms	SHA-1	DES	56	Yes
FIPS_WITH_3DES_EDE_CBC_SHA Note: Available only on Windows and UNIX platforms	SHA-1	3DES	168	Yes
Note: 1. Is the CipherSpec FIPS-certified on a FIPS-certified platform? See "CipherSuites and CipherSpecs" in the <i>WebSphere MQ Security</i> / <i>WebSphere MQ: Security</i> manual for an explanation of FIPS.				

When you request a personal certificate, you specify a key size for the public and private key pair. The key size that is used during the SSL handshake can depend on the size stored in the certificate and on the CipherSpec:

- On UNIX systems, Windows systems, and z/OS, when a CipherSpec name includes _EXPORT, the maximum handshake key size is 512 bits. If either of the certificates exchanged during the SSL handshake has a key size greater than 512 bits, a temporary 512-bit key is generated for use during the handshake.
- On UNIX and Windows systems, when a CipherSpec name includes _EXPORT1024, the handshake key size is 1024 bits.
- Otherwise the handshake key size is the size stored in the certificate.

If the SSLCIPH parameter is blank, no attempt is made to use SSL on the channel.

SSLClientAuth (MQCFIN)

Client authentication (parameter identifier: MQIACH_SSL_CLIENT_AUTH).

The value can be:

MQSCA_REQUIRED

Client authentication required

MQSCA_OPTIONAL

Client authentication optional.

Defines whether WebSphere MQ requires a certificate from the SSL client.

The initiating end of the channel acts as the SSL client, so this applies to the end of the channel that receives the initiation flow, which acts as the SSL server.

The parameter is used only for channels with SSLCIPH specified. If SSLCIPH is blank, the data is ignored and no error message is issued.

SSLPeerName (MQCFST)

Peer name (parameter identifier: MQCACH_SSL_PEER_NAME).

On platforms other than z/OS, the length of the string is MQ_SSL_PEER_NAME_LENGTH. On z/OS, it is MQ_SSL_SHORT_PEER_NAME_LENGTH.

Specifies the filter to use to compare with the Distinguished Name of the certificate from the peer queue manager or client at the other end of the channel. (A Distinguished Name is the identifier of the SSL certificate.) If the Distinguished Name in the certificate received from the peer does not match the SSLPEER filter, the channel does not start.

This parameter is optional; if it is not specified, the Distinguished Name of the peer is not checked at channel start up. (The Distinguished Name from the certificate is still written into the SSLPEER definition held in memory, and passed to the security exit). If SSLCIPH is blank, the data is ignored and no error message is issued.

This parameter is valid for all channel types.

The SSLPEER value is specified in the standard form used to specify a Distinguished Name. For example: SSLPEER('CN="xxx yyy zzz",0=xxx,C=xxx')

You can use a semi-colon as a separator instead of a comma.

The possible attribute types supported are:

CN	common name
T	title
OU	organizational unit name
O	organization name
L	locality name
ST, SP TM or S	state or province name
C	country

WebSphere MQ only accepts upper case letters for the attribute types.

If any of the unsupported attribute types are specified in the SSLPEER string, an error is output either when the attribute is defined or at run time (depending on which platform you are running on), and the string is deemed not to have matched the flowed certificate's Distinguished Name.

If the flowed certificate's Distinguished Name contains multiple OU (organisational unit) attributes, and SSLPEER specifies these attributes to be compared, they must be defined in descending hierarchical order. For example, if the flowed certificate's Distinguished Name contains the OUs OU=Large Unit,OU=Medium Unit,OU=Small Unit, specifying the following SSLPEER values will work:

```
('OU=Large Unit,OU=Medium Unit')
('OU=*,OU=Medium Unit,OU=Small Unit')
('OU=*,OU=Medium Unit')
```

but specifying the following SSLPEER values will fail:

```
('OU=Medium Unit,OU=Small Unit')
('OU=Large Unit,OU=Small Unit')
('OU=Medium Unit')
```

Any or all of the attribute values can be generic, either an asterisk (*) on its own, or a stem with initiating or trailing asterisks. This allows the SSLPEER to match any Distinguished Name value, or any value starting with the stem for that attribute.

If an asterisk is specified at the beginning or end of any attribute value in the Distinguished Name on the certificate, you can specify * to check for an exact match in SSLPEER. For example, if you have an attribute of CN=Test* in the Distinguished Name of the certificate, you can use the following command:

```
SSLPEER('CN=Test\*')
```

TpName (MQCFST)

Transaction program name (parameter identifier: MQCACH_TP_NAME).

This is the LU 6.2 transaction program name.

The maximum length of the string is MQ_TP_NAME_LENGTH.

- On HP OpenVMS, i5/OS, Compaq NonStop Kernel, UNIX systems, and Windows, this can be set only to blanks. The actual name is taken instead from the CPI-C Communications Side Object or (on Windows) from the CPI-C symbolic destination name properties.

This parameter is valid only for channels with a *TransportType* of MQXPT_LU62. It is not valid for receiver channels.

TransportType (MQCFIN)

Transmission protocol type (parameter identifier: MQIACH_XMIT_PROTOCOL_TYPE).

No check is made that the correct transport type has been specified if the channel is initiated from the other end. The value can be:

MQXPT_LU62
LU 6.2.

MQXPT_TCP
TCP.

MQXPT_NETBIOS
NetBIOS.

This value is supported in Windows. It also applies to z/OS for defining client-connection channels that connect to servers on the platforms supporting NetBIOS.

MQXPT_SPX
SPX.

This value is supported in Windows. It also applies to z/OS for defining client-connection channels that connect to servers on the platforms supporting SPX.

MQXPT_DECNET
DECnet.

This value is supported in the following environment: HP OpenVMS.

UserIdentifier (**MQCFST**)

Task user identifier (parameter identifier: MQCACH_USER_ID).

This is used by the message channel agent when attempting to initiate a secure SNA session with a remote message channel agent. On HP OpenVMS, i5/OS, Compaq NonStop Kernel, UNIX systems, it is valid only for *ChannelType* values of MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER, MQCHT_CLNTCONN, MQCHT_CLUSSDR, or MQCHT_CLUSRCVR. On z/OS, it is valid only for a *ChannelType* value of MQCHT_CLNTCONN.

The maximum length of the string is MQ_USER_ID_LENGTH. However, only the first 10 characters are used.

XmitQName (**MQCFST**)

Transmission queue name (parameter identifier: MQCACH_XMIT_Q_NAME).

The maximum length of the string is MQ_Q_NAME_LENGTH.

A transmission queue name is required (either previously defined or specified here) if *ChannelType* is MQCHT_SENDER or MQCHT_SERVER. It is not valid for other channel types.

Error codes

This command might return the following in the response format header, in addition to the values shown on page "Error codes applicable to all commands" on page 27.

Reason (**MQLONG**)

The value can be:

MQRCCF_BATCH_INT_ERROR

Batch interval not valid.

MQRCCF_BATCH_INT_WRONG_TYPE

Batch interval parameter not allowed for this channel type.

MQRCCF_BATCH_SIZE_ERROR

Batch size not valid.

MQRCCF_CHANNEL_NAME_ERROR

Channel name error.

MQRCCF_CHANNEL_NOT_FOUND

Channel not found.

MQRCCF_CHANNEL_TYPE_ERROR

Channel type not valid.

MQRCCF_CLUSTER_NAME_CONFLICT

Cluster name conflict.

MQRCCF_DISC_INT_ERROR
Disconnection interval not valid.

MQRCCF_DISC_INT_WRONG_TYPE
Disconnection interval not allowed for this channel type.

MQRCCF_HB_INTERVAL_ERROR
Heartbeat interval not valid.

MQRCCF_HB_INTERVAL_WRONG_TYPE
Heartbeat interval parameter not allowed for this channel type.

MQRCCF_LONG_RETRY_ERROR
Long retry count not valid.

MQRCCF_LONG_RETRY_WRONG_TYPE
Long retry parameter not allowed for this channel type.

MQRCCF_LONG_TIMER_ERROR
Long timer not valid.

MQRCCF_LONG_TIMER_WRONG_TYPE
Long timer parameter not allowed for this channel type.

MQRCCF_MAX_MSG_LENGTH_ERROR
Maximum message length not valid.

MQRCCF_MCA_NAME_ERROR
Message channel agent name error.

MQRCCF_MCA_NAME_WRONG_TYPE
Message channel agent name not allowed for this channel type.

MQRCCF_MCA_TYPE_ERROR
Message channel agent type not valid.

MQRCCF_MISSING_CONN_NAME
Connection name parameter required but missing.

MQRCCF_MR_COUNT_ERROR
Message retry count not valid.

MQRCCF_MR_COUNT_WRONG_TYPE
Message-retry count parameter not allowed for this channel type.

MQRCCF_MR_EXIT_NAME_ERROR
Channel message-retry exit name error.

MQRCCF_MR_EXIT_NAME_WRONG_TYPE
Message-retry exit parameter not allowed for this channel type.

MQRCCF_MR_INTERVAL_ERROR
Message retry interval not valid.

MQRCCF_MR_INTERVAL_WRONG_TYPE
Message-retry interval parameter not allowed for this channel type.

MQRCCF_MSG_EXIT_NAME_ERROR
Channel message exit name error.

MQRCCF_NET_PRIORITY_ERROR
Network priority value error.

MQRCCF_NET_PRIORITY_WRONG_TYPE
Network priority attribute not allowed for this channel type.

MQRCCF_NPM_SPEED_ERROR
Nonpersistent message speed not valid.

MQRCCF_NPM_SPEED_WRONG_TYPE
Nonpersistent message speed parameter not allowed for this channel type.

MQRCCF_PARM_SEQUENCE_ERROR
Parameter sequence not valid.

MQRCCF_PUT_AUTH_ERROR
Put authority value not valid.

MQRCCF_PUT_AUTH_WRONG_TYPE
Put authority parameter not allowed for this channel type.

MQRCCF_RCV_EXIT_NAME_ERROR
Channel receive exit name error.

MQRCCF_SEC_EXIT_NAME_ERROR
Channel security exit name error.

MQRCCF_SEND_EXIT_NAME_ERROR
Channel send exit name error.

MQRCCF_SEQ_NUMBER_WRAP_ERROR
Sequence wrap number not valid.

MQRCCF_SHORT_RETRY_ERROR
Short retry count not valid.

MQRCCF_SHORT_RETRY_WRONG_TYPE
Short retry parameter not allowed for this channel type.

MQRCCF_SHORT_TIMER_ERROR
Short timer value not valid.

MQRCCF_SHORT_TIMER_WRONG_TYPE
Short timer parameter not allowed for this channel type.

MQRCCF_SSL_CIPHER_SPEC_ERROR
SSL CipherSpec not valid.

MQRCCF_SSL_CLIENT_AUTH_ERROR
SSL client authentication not valid.

MQRCCF_SSL_PEER_NAME_ERROR
SSL peer name not valid.

MQRCCF_WRONG_CHANNEL_TYPE
Parameter not allowed for this channel type.

MQRCCF_XMIT_PROTOCOL_TYPE_ERR
Transmission protocol type not valid.

MQRCCF_XMIT_Q_NAME_ERROR
Transmission queue name error.

MQRCCF_XMIT_Q_NAME_WRONG_TYPE
Transmission queue name not allowed for this channel type.

Change, Copy, and Create Channel Listener

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
		X	X	X	

The Change Channel Listener (MQCMD_CHANGE_LISTENER) command changes the specified attributes of an existing WebSphere MQ listener definition. For any optional parameters that are omitted, the value does not change.

The Copy Channel Listener (MQCMD_COPY_LISTENER) command creates a new WebSphere MQ listener definition, using, for attributes not specified in the command, the attribute values of an existing listener definition.

The Create Channel Listener (MQCMD_CREATE_LISTENER) command creates a new WebSphere MQ listener definition. Any attributes that are not defined explicitly are set to the default values on the destination queue manager.

Required parameters (Change and Create Channel Listener):

ListenerName, TransportType

Required parameters (Copy Channel Listener):

FromListenerName, ToListenerName

Optional parameters:

*Adapter, Backlog, Commands, IPAddress, ListenerDesc, LocalName,
NetbiosNames, Port, Replace, Sessions, Socket, StartMode, TPname*

Required parameters (Change and Create Channel Listener)

ListenerName (MQCFST)

The name of the listener definition to be changed or created (parameter identifier: MQCACH_LISTENER_NAME).

The maximum length of the string is MQ_LISTENER_NAME_LENGTH.

TransportType (MQCFIN)

Transmission protocol (parameter identifier: MQIACH_XMIT_PROTOCOL_TYPE).

The value can be:

MQXPT_TCP
TCP.

MQXPT_LU62

LU 6.2. This is valid only on Windows.

MQXPT_NETBIOS

NetBIOS. This is valid only on Windows.

MQXPT_SPX

SPX. This is valid only on Windows.

Required parameters (Copy Channel Listener)

FromListenerName (MQCFST)

The name of the listener definition to be copied from (parameter identifier: MQCACF_FROM_LISTENER_NAME).

This specifies the name of the existing listener definition that contains values for the attributes not specified in this command.

The maximum length of the string is MQ_LISTENER_NAME_LENGTH.

ToListenerName (MQCFST)

To listener name (parameter identifier: MQCACF_TO_LISTENER_NAME).

This specifies the name of the new listener definition. If a listener definition with this name already exists, *Replace* must be specified as MQRP_YES.

The maximum length of the string is MQ_LISTENER_NAME_LENGTH.

Optional parameters

Adapter (MQCFIN)

Adapter number (parameter identifier: MQIACH_ADAPTER).

The adapter number on which NetBIOS listens. This is valid only on Windows.

Backlog (MQCFIN)

Backlog (parameter identifier: MQIACH_BACKLOG).

The number of concurrent connection requests that the listener supports.

Commands (MQCFIN)

Adapter number (parameter identifier: MQIACH_COMMAND_COUNT).

The number of commands that the listener can use. This is valid only on Windows.

IPAddress (MQCFST)

IP address (parameter identifier: MQCACH_IP_ADDRESS).

IP address for the listener specified in IPv4 dotted decimal, IPv6 hexadecimal notation, or alphanumeric hostname form. If you do not specify a value for this parameter, the listener listens on all configured IPv6 and IPv6 stacks.

The maximum length of the string is MQ_LOCAL_ADDRESS_LENGTH

ListenerDesc (MQCFST)

Description of listener definition (parameter identifier: MQCACH_LISTENER_DESC).

This is a plain-text comment that provides descriptive information about the listener definition. It should contain only displayable characters.

If characters are used that are not in the coded character set identifier (CCSID) for the queue manager on which the command is executing, they might be translated incorrectly.

The maximum length of the string is MQ_LISTENER_DESC_LENGTH.

LocalName (MQCFST)

NetBIOS local name (parameter identifier: MQCACH_LOCAL_NAME).

The NetBIOS local name that the listener uses. This is valid only on Windows.

The maximum length of the string is MQ_CONN_NAME_LENGTH

NetbiosNames (MQCFIN)

NetBIOS names (parameter identifier: MQIACH_NAME_COUNT).

The number of names that the listener supports. This is valid only on Windows.

Port (**MQCFIN**)

Port number (parameter identifier: MQIACH_PORT).

The port number for TCP/IP. This is valid only if the value of *TransportType* is MQXPT_TCP.

Replace (**MQCFIN**)

Replace attributes (parameter identifier: MQIACF_REPLACE).

If a namelist definition with the same name as *ToListenerName* already exists, this specifies whether it is to be replaced. The value can be:

MQRP_YES

Replace existing definition.

MQRP_NO

Do not replace existing definition.

Sessions (**MQCFIN**)

NetBIOS sessions (parameter identifier: MQIACH_SESSION_COUNT).

The number of sessions that the listener can use. This is valid only on Windows.

Socket (**MQCFIN**)

SPX socket number (parameter identifier: MQIACH_SOCKET).

The SPX socket on which to listen. This is valid only if the value of *TransportType* is MQXPT_SPX.

StartMode (**MQCFIN**)

Service mode (parameter identifier: MQIACH_LISTENER_CONTROL).

Specifies how the listener is to be started and stopped. The value can be:

MQSVC_CONTROL_MANUAL

The listener is not to be started automatically or stopped automatically. It is to be controlled by user command. This is the default value.

MQSVC_CONTROL_Q_MGR

The listener being defined is to be started and stopped at the same time as the queue manager is started and stopped.

MQSVC_CONTROL_Q_MGR_START

The listener is to be started at the same time as the queue manager is started, but is not requested to stop when the queue manager is stopped.

TPName (**MQCFST**)

Transaction program name (parameter identifier: MQCACH_TP_NAME).

The LU 6.2 transaction program name. This is valid only on Windows.

The maximum length of the string is MQ_TP_NAME_LENGTH

Change, Copy, and Create Namelist

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Change Namelist (MQCMD_CHANGE_NAMELIST) command changes the specified attributes of an existing WebSphere MQ namelist definition. For any optional parameters that are omitted, the value does not change.

The Copy Namelist (MQCMD_COPY_NAMELIST) command creates a new WebSphere MQ namelist definition, using, for attributes not specified in the command, the attribute values of an existing namelist definition.

The Create Namelist (MQCMD_CREATE_NAMELIST) command creates a new WebSphere MQ namelist definition. Any attributes that are not defined explicitly are set to the default values on the destination queue manager.

Required parameter (Change and Create Namelist):

NamelistName

Required parameters (Copy Namelist):

FromNamelistName, ToNamelistName

Optional parameters:

CommandScope, NamelistDesc, NamelistType, Names, QSGDisposition, Replace

Required parameter (Change and Create Namelist)

NamelistName (MQCFST)

The name of the namelist definition to be changed (parameter identifier: MQCA_NAMELIST_NAME).

The maximum length of the string is MQ_NAMELIST_NAME_LENGTH.

Required parameters (Copy Namelist)

FromNamelistName (MQCFST)

The name of the namelist definition to be copied from (parameter identifier: MQCACF_FROM_NAMELIST_NAME).

This specifies the name of the existing namelist definition that contains values for the attributes not specified in this command.

On z/OS, the queue manager searches for an object with the name you specify and a disposition of MQQSGD_Q_MGR or MQQSGD_COPY to copy from.

This parameter is ignored if a value of MQQSGD_COPY is specified for *QSGDisposition*. In this case, an object with the name specified by *ToNamelistName* and the disposition MQQSGD_GROUP is searched for to copy from.

The maximum length of the string is MQ_NAMELIST_NAME_LENGTH.

ToNamelistName (MQCFST)

To namelist name (parameter identifier: MQCACF_TO_NAMELIST_NAME).

This specifies the name of the new namelist definition. If a namelist definition with this name already exists, *Replace* must be specified as MQRP_YES.

The maximum length of the string is MQ_NAMELIST_NAME_LENGTH.

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

NamelistDesc (MQCFST)

Description of namelist definition (parameter identifier: MQCA_NAMELIST_DESC).

This is a plain-text comment that provides descriptive information about the namelist definition. It should contain only displayable characters.

If characters are used that are not in the coded character set identifier (CCSID) for the queue manager on which the command is executing, they might be translated incorrectly.

The maximum length of the string is MQ_NAMELIST_DESC_LENGTH.

NamelistType (MQCFIN)

Type of names in the namelist (parameter identifier: MQIA_NAMELIST_TYPE). This parameter applies to z/OS only.

Specifies the type of names in the namelist . The value can be:

MQNT_NONE

The names are of no particular type.

MQNT_Q

A namelist that holds a list of queue names.

MQNT_CLUSTER

A namelist that is associated with clustering, containing a list of the cluster names.

MQNT_AUTH_INFO

The namelist is associated with SSL, and contains a list of authentication information object names.

Names (MQCFSL)

The names to be placed in the namelist (parameter identifier: MQCA_NAMES).

The number of names in the list is given by the *Count* field in the MQCFSL structure. The length of each name is given by the *StringLength* field in that structure. The maximum length of a name is MQ_OBJECT_NAME_LENGTH.

QSGDisposition (MQCFIN)

Disposition of the object within the group (parameter identifier: MQIA_QSG_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves). The value can be:

QSGDisposition	Change	Copy, Create
MQQSGD_COPY	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameter MQQSGD_COPY. Any object residing in the shared repository, or any object defined using a command that had the parameters MQQSGD_Q_MGR, is not affected by this command.	The object is defined on the page set of the queue manager that executes the command using the MQQSGD_GROUP object of the same name as the <i>ToNameListName</i> object (for Copy) or <i>NameListName</i> object (for Create).
MQQSGD_GROUP	<p>The object definition resides in the shared repository. The object was defined using a command that had the parameter MQQSGD_GROUP. Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command.</p> <p>If the command is successful, the following MQSC command is generated and sent to all active queue managers in the queue-sharing group so that they make or refresh local copies on page set zero:</p> <pre>DEFINE NAMELIST(name) REPLACE QSGDISP(COPY)</pre> <p>The Change for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.</p>	<p>The object definition resides in the shared repository. This is allowed only if the queue manager is in a queue-sharing group.</p> <p>If the definition is successful, the following MQSC command is generated and sent to all active queue managers in the queue-sharing group so that they make or refresh local copies on page set zero:</p> <pre>DEFINE NAMELIST(name) REPLACE QSGDISP(COPY)</pre> <p>The Copy or Create for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.</p>
MQQSGD_PRIVATE	The object resides on the page set of the queue manager that executes the command, and was defined with MQQSGD_Q_MGR or MQQSGD_COPY. Any object residing in the shared repository is unaffected.	Not permitted.
MQQSGD_Q_MGR	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameter MQQSGD_Q_MGR. Any object residing in the shared repository, or any local copy of such an object, is not affected by this command. This is the default value.	The object is defined on the page set of the queue manager that executes the command. This is the default value.

Replace (MQCFIN)

Replace attributes (parameter identifier: MQIACF_REPLACE).

If a namelist definition with the same name as *ToNameListName* already exists, this specifies whether it is to be replaced. The value can be:

MQRP_YES

Replace existing definition.

MQRP_NO

Do not replace existing definition.

Change, Copy, and Create Process

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Change Process (MQCMD_CHANGE_PROCESS) command changes the specified attributes of an existing WebSphere MQ process definition. For any optional parameters that are omitted, the value does not change.

The Copy Process (MQCMD_COPY_PROCESS) command creates a new WebSphere MQ process definition, using, for attributes not specified in the command, the attribute values of an existing process definition.

The Create Process (MQCMD_CREATE_PROCESS) command creates a new WebSphere MQ process definition. Any attributes that are not defined explicitly are set to the default values on the destination queue manager.

Required parameter (Change and Create Process):

ProcessName

Required parameters (Copy Process):

FromProcessName, ToProcessName

Optional parameters:

*ApplId, ApplType, CommandScope, EnvData, ProcessDesc, QSGDisposition,
Replace, UserData*

Required parameters (Change and Create Process)

ProcessName (MQCFST)

The name of the process definition to be changed or created (parameter identifier: MQCA_PROCESS_NAME).

The maximum length of the string is MQ_PROCESS_NAME_LENGTH.

Required parameters (Copy Process)

FromProcessName (MQCFST)

The name of the process definition to be copied from (parameter identifier: MQCACF_FROM_PROCESS_NAME).

Specifies the name of the existing process definition that contains values for the attributes not specified in this command.

On z/OS, the queue manager searches for an object with the name you specify and a disposition of MQQSGD_Q_MGR or MQQSGD_COPY to copy from. This parameter is ignored if a value of MQQSGD_COPY is specified for *QSGDisposition*. In this case, an object with the name specified by *ToProcessName* and the disposition MQQSGD_GROUP is searched for to copy from.

The maximum length of the string is MQ_PROCESS_NAME_LENGTH.

ToProcessName (MQCFST)

To process name (parameter identifier: MQCACF_TO_PROCESS_NAME).

The name of the new process definition. If a process definition with this name already exists, *Replace* must be specified as MQRP_YES.

The maximum length of the string is MQ_PROCESS_NAME_LENGTH.

Optional parameters

ApplId (MQCFST)

Application identifier (parameter identifier: MQCA_APPL_ID).

This is the name of the application to be started, on the platform for which the command is executing, and might typically be a program name and library name.

The maximum length of the string is MQ_PROCESS_APPL_ID_LENGTH.

ApplType (MQCFIN)

Application type (parameter identifier: MQIA_APPL_TYPE).

Valid application types are:

MQAT_OS400

i5/OS application.

MQAT_WINDOWS_NT

Windows or Windows 95, Windows 98 application.

MQAT_DOS

DOS client application.

MQAT_WINDOWS

Windows client application.

MQAT_UNIX

UNIX application.

MQAT_AIX

AIX application (same value as MQAT_UNIX).

MQAT_CICS

CICS® transaction.

MQAT_VMS

HP OpenVMS application.

MQAT NSK

Compaq NonStop Kernel application.

MQAT_ZOS

z/OS application.

MQAT_DEFAULT

Default application type.

integer: System-defined application type in the range zero through 65 535 or a user-defined application type in the range 65 536 through 999 999 999 (not checked).

Only application types (other than user-defined types) that are supported on the platform at which the command is executed should be used:

- On HP OpenVMS:

MQAT_VMS,
MQAT_DOS,
MQAT_WINDOWS, and
MQAT_DEFAULT are supported.

- On i5/OS:

MQAT_OS400,
MQAT_CICS, and
MQAT_DEFAULT are supported.

- On Compaq NonStop Kernel:

MQAT NSK,
MQAT_DOS,
MQAT_WINDOWS, and
MQAT_DEFAULT are supported.

- On UNIX systems:

MQAT_UNIX,
MQAT_OS2,
MQAT_DOS,
MQAT_WINDOWS,
MQAT_CICS, and
MQAT_DEFAULT are supported.

- On Windows:

MQAT_WINDOWS_NT,
MQAT_OS2,
MQAT_DOS,
MQAT_WINDOWS,
MQAT_CICS, and
MQAT_DEFAULT are supported.

- On z/OS:

MQAT_DOS,
MQAT_IMS
MQAT_MVS,
MQAT_UNIX,
MQAT_CICS, and
MQAT_DEFAULT are supported.

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

EnvData (MQCFST)

Environment data (parameter identifier: MQCA_ENV_DATA).

A character string that contains environment information pertaining to the application to be started.

The maximum length of the string is MQ_PROCESS_ENV_DATA_LENGTH.

ProcessDesc (MQCFST)

Description of process definition (parameter identifier: MQCA_PROCESS_DESC).

A plain-text comment that provides descriptive information about the process definition. It must contain only displayable characters.

The maximum length of the string is MQ_PROCESS_DESC_LENGTH.

If characters are used that are not in the coded character set identifier (CCSID) for the queue manager on which the command is executing, they might be translated incorrectly.

QSGDisposition (MQCFIN)

Disposition of the object within the group (parameter identifier: MQIA_QSG_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves). The value can be:

QSGDisposition	Change	Copy, Create
MQQSGD_COPY	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameter MQQSGD_COPY. Any object residing in the shared repository, or any object defined using a command that had the parameters MQQSGD_Q_MGR, is not affected by this command.	The object is defined on the page set of the queue manager that executes the command using the MQQSGD_GROUP object of the same name as the <i>ToProcessName</i> object (for Copy) or <i>ProcessName</i> object (for Create).
MQQSGD_GROUP	The object definition resides in the shared repository. The object was defined using a command that had the parameter MQQSGD_GROUP. Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command. If the command is successful, the following MQSC command is generated and sent to all active queue managers in the queue-sharing group to attempt to refresh local copies on page set zero: DEFINE PROCESS(process-name) REPLACE QSGDISP(COPY) The Change for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.	The object definition resides in the shared repository. This is allowed only if the queue manager is in a queue-sharing group. If the definition is successful, the following MQSC command is generated and sent to all active queue managers in the queue-sharing group to attempt to make or refresh local copies on page set zero: DEFINE PROCESS(process-name) REPLACE QSGDISP(COPY) The Copy or Create for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.

QSGDisposition	Change	Copy, Create
MQQSGD_PRIVATE	The object resides on the page set of the queue manager that executes the command, and was defined with MQQSGD_Q_MGR or MQQSGD_COPY. Any object residing in the shared repository is unaffected.	Not permitted.
MQQSGD_Q_MGR	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameter MQQSGD_Q_MGR. Any object residing in the shared repository, or any local copy of such an object, is not affected by this command. This is the default value.	The object is defined on the page set of the queue manager that executes the command. This is the default value.

Replace (MQCFIN)

Replace attributes (parameter identifier: MQIACF_REPLACE).

If a process definition with the same name as *ToProcessName* already exists, this specifies whether it is to be replaced.

The value can be:

MQRP_YES

Replace existing definition.

MQRP_NO

Do not replace existing definition.

UserData (MQCFST)

User data (parameter identifier: MQCA_USER_DATA).

A character string that contains user information pertaining to the application (defined by *ApplId*) that is to be started.

The maximum length of the string is MQ_PROCESS_USER_DATA_LENGTH.

Change, Copy, and Create Queue

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Change Queue (MQCMD_CHANGE_Q) command changes the specified attributes of an existing WebSphere MQ queue. For any optional parameters that are omitted, the value does not change.

The Copy Queue (MQCMD_COPY_Q) command creates a new queue definition, of the same type, using, for attributes not specified in the command, the attribute values of an existing queue definition.

The Create Queue (MQCMD_CREATE_Q) command creates a queue definition with the specified attributes. All attributes that are not specified are set to the default value for the type of queue that is created.

Table 5 shows the parameters that are applicable to each type of queue.

Table 5. Change, Copy, Create Queue parameters

	Local queue	Model queue	Alias queue	Remote queue
<i>BackoutRequeueName</i>	X	X		
<i>BackoutThreshold</i>	X	X		
<i>BaseQName</i>			X	
<i>CFStructure</i>	X	X		
<i>ClusterName</i>	X		X	X
<i>ClusterNamelist</i>	X		X	X
<i>CLWLQueuePriority</i>	X		X	X
<i>CLWLQueueRank</i>	X		X	X
<i>CLWLUseQ</i>	X			
<i>CommandScope</i>	X	X	X	X
<i>DefBind</i>	X		X	X
<i>DefInputOpenOption</i>	X	X		
<i>DefPersistence</i>	X	X	X	X
<i>DefPriority</i>	X	X	X	X
<i>DistLists</i>	X	X		
<i>Force</i>	X		X	X
<i>FromQName</i> ²	X	X	X	X
<i>HardenGetBackout</i>	X	X		
<i>IndexType</i>	X	X		
<i>InhibitGet</i>	X	X	X	
<i>InhibitPut</i>	X	X	X	X
<i>InitiationQName</i>	X	X		
<i>MaxMsgLength</i>	X	X		
<i>MaxQDepth</i>	X	X		
<i>MsgDeliverySequence</i>	X	X		
<i>NonPersistentMessage Class</i>	X	X		
<i>ProcessName</i>	X	X		
<i>QDepthHighEvent</i>	X	X		
<i>QDepthHighLimit</i>	X	X		
<i>QDepthLowEvent</i>	X	X		
<i>QDepthLowLimit</i>	X	X		
<i>QDepthMaxEvent</i>	X	X		
<i>QDesc</i>	X	X	X	X
<i>QName</i> ¹	X	X	X	X
<i>QServiceInterval</i>	X	X		
<i>QServiceIntervalEvent</i>	X	X		
<i>QSGDisposition</i>	X	X	X	X
<i>QType</i> ³	X	X		

Table 5. Change, Copy, Create Queue parameters (continued)

	Local queue	Model queue	Alias queue	Remote queue
<i>QueueAccounting</i>	X	X		
<i>QueueMonitoring</i>	X	X		
<i>QueueStatistics</i>	X	X		
<i>RemoteQMgrName</i>				X
<i>RemoteQName</i>				X
<i>Replace</i> (not valid on Change Queue command)	X	X	X	X
<i>RetentionInterval</i>	X	X		
<i>Scope</i>	X		X	X
<i>Shareability</i>	X	X		
<i>StorageClass</i>	X	X		
<i>ToQName</i> ²	X	X	X	X
<i>TriggerControl</i>	X	X		
<i>TriggerData</i>	X	X		
<i>TriggerDepth</i>	X	X		
<i>TriggerMsgPriority</i>	X	X		
<i>TriggerType</i>	X	X		
<i>Usage</i>	X	X		
<i>XmitQName</i>				X

Note:

- Required parameter on Change and Create Queue commands
- Required parameter on Copy Queue command
- Required parameter on Change, Create, and Copy Queue commands

Required parameters (Change and Create Queue)

QName (MQCFST)

Queue name (parameter identifier: MQCA_Q_NAME).

The name of the queue to be changed. The maximum length of the string is MQ_Q_NAME_LENGTH.

Required parameters (Copy Queue)

FromQName (MQCFST)

From queue name (parameter identifier: MQCACF_FROM_Q_NAME).

Specifies the name of the existing queue definition.

On z/OS, the queue manager searches for an object with the name you specify and a disposition of MQQSGD_Q_MGR, MQQSGD_COPY, or MQQSGD_SHARED to copy from. This parameter is ignored if a value of MQQSGD_COPY is specified for *QSGDisposition*. In this case, an object with the name specified by *ToQName* and the disposition MQQSGD_GROUP is searched for to copy from.

The maximum length of the string is MQ_Q_NAME_LENGTH.

ToQName (MQCFST)

To queue name (parameter identifier: MQCACF_TO_Q_NAME).

Specifies the name of the new queue definition.

The maximum length of the string is MQ_Q_NAME_LENGTH.

Queue names must be unique; if a queue definition already exists with the name and type of the new queue, *Replace* must be specified as MQRP_YES. If a queue definition exists with the same name as and a different type from the new queue, the command will fail.

Required parameters (all commands)

QType (MQCFIN)

Queue type (parameter identifier: MQIA_Q_TYPE).

The value specified must match the type of the queue being changed.

The value can be:

MQQT_ALIAS

Alias queue definition.

MQQT_LOCAL

Local queue.

MQQT_REMOTE

Local definition of a remote queue.

MQQT_MODEL

Model queue definition.

Optional parameters

BackoutQueueName (MQCFST)

Excessive backout requeue name (parameter identifier: MQCA_BACKOUT_REQ_Q_NAME).

Specifies the local name of the queue (not necessarily a local queue) to which a message is transferred if it is backed out more times than the value of *BackoutThreshold*.

The backout queue does not need to exist at this time but it must exist when the *BackoutThreshold* value is exceeded.

The maximum length of the string is MQ_Q_NAME_LENGTH.

BackoutThreshold (MQCFIN)

Backout threshold (parameter identifier: MQIA_BACKOUT_THRESHOLD).

The number of times a message can be backed out before it is transferred to the backout queue specified by *BackoutQueueName*.

If the value is subsequently reduced, any messages already on the queue that have been backed out at least as many times as the new value remain on the queue, but such messages are transferred if they are backed out again.

Specify a value in the range 0 through 999 999 999.

BaseQName (MQCFST)

Queue name to which the alias resolves (parameter identifier: MQCA_BASE_Q_NAME).

This is the name of a local or remote queue that is defined to the local queue manager.

The maximum length of the string is MQ_Q_NAME_LENGTH.

CFStructure (MQCFST)

Coupling facility structure name (parameter identifier: MQCA_CF_STRUC_NAME). This parameter applies to z/OS only.

Specifies the name of the coupling facility structure where you want to store messages when you use shared queues. The name:

- Cannot have more than 12 characters
- Must start with an uppercase letter (A through Z)
- Can include only the characters A through Z and 0 through 9

The maximum length of the string is MQ_CF_STRUC_NAME_LENGTH.

The name of the queue-sharing group to which the queue manager is connected is prefixed to the name you supply. The name of the queue-sharing group is always four characters, padded with @ symbols if necessary. For example, if you use a queue-sharing group named NY03 and you supply the name PRODUCT7, the resultant Coupling Facility structure name is NY03PRODUCT7. Note that the administrative structure for the queue-sharing group (in this case NY03CSQ_ADMIN) cannot be used for storing messages.

For local and model queues, when you use the Create Queue command with a value of MQRP_YES in the *Replace* parameter, or the Change Queue command, the following rules apply:

- On a local queue with a value of MQQSGD_SHARED in the *QSGDisposition* parameter, *CFStructure* cannot change.
If you need to change either the *CFStructure* or *QSGDisposition* value, you must delete and redefine the queue. To preserve any of the messages on the queue you must off-load the messages before you delete the queue and reload the messages after you have redefined the queue, or move the messages to another queue.
- On a model queue with a value of MQQDT_SHARED_DYNAMIC in the *DefinitionType* parameter, *CFStructure* cannot be blank.
- On a local queue with a value other than MQQSGD_SHARED in the *QSGDisposition* parameter, or a model queue with a value other than MQQDT_SHARED_DYNAMIC in the *DefinitionType* parameter, the value of *CFStructure* does not matter.

For local and model queues, when you use the Create Queue command with a value of MQRP_NO in the *Replace* parameter, the Coupling Facility structure:

- On a local queue with a value of MQQSGD_SHARED in the *QSGDisposition* parameter, or a model queue with a value of MQQDT_SHARED_DYNAMIC in the *DefinitionType* parameter, *CFStructure* cannot be blank.
- On a local queue with a value other than MQQSGD_SHARED in the *QSGDisposition* parameter, or a model queue with a value other than MQQDT_SHARED_DYNAMIC in the *DefinitionType* parameter, the value of *CFStructure* does not matter.

Note: Before you can use the queue, the structure must be defined in the Coupling Facility Resource Management (CFRM) policy data set.

ClusterName (**MQCFST**)

Cluster name (parameter identifier: MQCA_CLUSTER_NAME).

The name of the cluster to which the queue belongs.

Changes to this parameter do not affect instances of the queue that are open.

Only one of the resultant values of *ClusterName* and *ClusterNamelist* can be nonblank; you cannot specify a value for both.

The maximum length of the string is MQ_CLUSTER_NAME_LENGTH.

ClusterNamelist (**MQCFST**)

Cluster namelist (parameter identifier: MQCA_CLUSTER_NAMELIST).

The name of the namelist, that specifies a list of clusters to which the queue belongs.

Changes to this parameter do not affect instances of the queue that are open.

Only one of the resultant values of *ClusterName* and *ClusterNamelist* can be nonblank; you cannot specify a value for both.

CLWLQueuePriority (**MQCFIN**)

Cluster workload queue priority (parameter identifier: MQIA_CLWL_Q_PRIORITY).

Specifies the priority of the queue in cluster workload management. The value must be in the range zero through 9, where zero is the lowest priority and 9 is the highest.

For more information about this parameter, see *WebSphere MQ Queue Manager Clusters*.

CLWLQueueRank (**MQCFIN**)

Cluster workload queue rank (parameter identifier: MQIA_CLWL_Q_RANK).

Specifies the rank of the queue in cluster workload management. The value must be in the range zero through 9, where zero is the lowest rank and 9 is the highest.

For more information about this parameter, see *WebSphere MQ Queue Manager Clusters*.

CLWLUseQ (**MQCFIN**)

Cluster workload use remote queue (parameter identifier: MQIA_CLWL_USEQ).

Specifies whether remote and local queues are to be used in cluster workload distribution. The value can be:

MQCLWL_USEQ_AS_Q_MGR

Use the value of the *CLWLUseQ* parameter on the queue manager's definition.

MQCLWL_USEQ_ANY

Use remote and local queues.

MQCLWL_USEQ_LOCAL

Do not use remote queues.

For more information about this parameter, see *WebSphere MQ Queue Manager Clusters*.

CommandScope (**MQCFST**)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

DefBind (**MQCFIN**)

Bind definition (parameter identifier: MQIA_DEF_BIND).

The parameter specifies the binding to be used when MQOO_BIND_AS_Q_DEF is specified on the MQOPEN call. The value can be:

MQBND_BIND_ON_OPEN

The binding is fixed by the MQOPEN call.

MQBND_BIND_NOT_FIXED

The binding is not fixed.

Changes to this parameter do not affect instances of the queue that are open.

DefinitionType (**MQCFIN**)

Queue definition type (parameter identifier: MQIA_DEFINITION_TYPE).

The value can be:

MQQDT_PERMANENT_DYNAMIC

Dynamically defined permanent queue.

MQQDT_SHARED_DYNAMIC

Dynamically defined shared queue. This option is available on z/OS only.

MQQDT_TEMPORARY_DYNAMIC

Dynamically defined temporary queue.

DefInputOpenOption (**MQCFIN**)

Default input open option (parameter identifier: MQIA_DEF_INPUT_OPEN_OPTION).

Specifies the default share option for applications opening this queue for input.

The value can be:

MQOO_INPUT_EXCLUSIVE

Open queue to get messages with exclusive access.

MQOO_INPUT_SHARED

Open queue to get messages with shared access.

DefPersistence (**MQCFIN**)

Default persistence (parameter identifier: MQIA_DEF_PERSISTENCE).

Specifies the default for message-persistence on the queue. Message persistence determines whether or not messages are preserved across restarts of the queue manager.

The value can be:

MQPER_PERSISTENT

Message is persistent.

MQPER_NOT_PERSISTENT

Message is not persistent.

DefPriority (MQCFIN)

Default priority (parameter identifier: MQIA_DEF_PRIORITY).

Specifies the default priority of messages put on the queue. The value must be in the range zero through to the maximum priority value that is supported (9).

DistLists (MQCFIN)

Distribution list support (parameter identifier: MQIA_DIST_LISTS).

Specifies whether distribution-list messages can be placed on the queue.

Note: This attribute is set by the sending message channel agent (MCA) which removes messages from the queue; this happens each time the sending MCA establishes a connection to a receiving MCA on a partnering queue manager. The attribute is not normally set by administrators, although it can be set if the need arises.

This parameter is supported in the following environments: AIX, HP-UX, i5/OS, Solaris, Windows and Linux.

The value can be:

MQDL_SUPPORTED

Distribution lists supported.

MQDL_NOT_SUPPORTED

Distribution lists not supported.

Force (MQCFIN)

Force changes (parameter identifier: MQIACF_FORCE).

Specifies whether the command should be forced to complete when conditions are such that completing the command would affect an open queue. The conditions depend upon the type of the queue that is being changed:

Alias QType: *BaseQName* is specified with a queue name and an application has the alias queue open.

Local QType: Either of the following conditions indicate that a local queue would be affected:

- *Shareability* is specified as MQQA_NOT_SHAREABLE and more than one application has the local queue open for input.
- The *Usage* value is changed and one or more applications has the local queue open, or there are one or more messages on the queue. (The *Usage* value should not normally be changed while there are messages on the queue; the format of messages changes when they are put on a transmission queue.)

Remote QType: Either of the following conditions indicate that a remote queue would be affected:

- *XmitQName* is specified with a transmission-queue name (or blank) and an application has a remote queue open that would be affected by this change.
- Any of the *RemoteQName*, *RemoteQMgrName* or *XmitQName* parameters is specified with a queue or queue-manager name, and one or more applications has a queue open that resolved through this definition as a queue-manager alias.

Model QType: This parameter is not valid for model queues.

Note: A value of MQFC_YES is not required if this definition is in use as a reply-to queue definition only.

The value can be:

MQFC_YES

Force the change.

MQFC_NO

Do not force the change.

HardenGetBackout (MQCFIN)

Whether to harden backout count (parameter identifier: MQIA_HARDEN_GET_BACKOUT).

Specifies whether the count of backed out messages is saved (hardened) across restarts of the message queue manager.

Note: WebSphere MQ for iSeries always hardens the count, regardless of the setting of this attribute.

The value can be:

MQQA_BACKOUT_HARDENED

Backout count remembered.

MQQA_BACKOUT_NOT_HARDENED

Backout count might not be remembered.

IndexType (MQCFIN)

Index type (parameter identifier: MQIA_INDEX_TYPE). This parameter applies to z/OS only.

Specifies the type of index maintained by the queue manager to expedite MQGET operations on the queue. For shared queues, the type of index determines what type of MQGETs can be used. The value can be:

MQIT_NONE

No index.

MQIT_MSG_ID

The queue is indexed using message identifiers.

MQIT_CORREL_ID

The queue is indexed using correlation identifiers.

MQIT_MSG_TOKEN

The queue is indexed using message tokens.

MQIT_GROUP_ID

The queue is indexed using group identifiers.

Messages can be retrieved using a selection criterion only if an appropriate index type is maintained, as the following table shows:

Retrieval selection criterion	<i>IndexType</i> required	
	Shared queue	Other queue
None (sequential retrieval)	Any	Any
Message identifier	MQIT_MSG_ID or MQIT_NONE	Any
Correlation identifier	MQIT_CORREL_ID	Any
Message and correlation identifiers	MQIT_MSG_ID or MQIT_CORREL_ID	Any
Group identifier	MQIT_GROUP_ID	Any
Grouping	MQIT_GROUP_ID	MQIT_GROUP_ID
Message token	Not allowed	MQIT_MSG_TOKEN

InhibitGet (MQCFIN)

Whether get operations are allowed (parameter identifier: MQIA_INHIBIT_GET).

The value can be:

MQQA_GET_ALLOWED

Get operations are allowed.

MQQA_GET_INHIBITED

Get operations are inhibited.

InhibitPut (MQCFIN)

Whether put operations are allowed (parameter identifier: MQIA_INHIBIT_PUT).

Specifies whether messages can be put on the queue.

The value can be:

MQQA_PUT_ALLOWED

Put operations are allowed.

MQQA_PUT_INHIBITED

Put operations are inhibited.

InitiationQName (MQCFST)

Initiation queue name (parameter identifier: MQCA_INITIATION_Q_NAME).

The local queue for trigger messages relating to this queue. The initiation queue must be on the same queue manager.

The maximum length of the string is MQ_Q_NAME_LENGTH.

MaxMsgLength (MQCFIN)

Maximum message length (parameter identifier: MQIA_MAX_MSG_LENGTH).

The maximum length for messages on the queue. Because applications might use the value of this attribute to determine the size of buffer they need to retrieve messages from the queue, change this value only if it is known that this will not cause an application to operate incorrectly.

Do not set a value that is greater than the queue manager's *MaxMsgLength* attribute.

The lower limit for this parameter is 0. The upper limit depends on the environment:

- On AIX, HP OpenVMS, Compaq NonStop Kernel, HP-UX, i5/OS, Solaris, Linux, Windows, and z/OS, the maximum message length is 100 MB (104 857 600 bytes).
- On UNIX systems not listed above, the maximum message length is 4 MB (4 194 304 bytes).

MaxQDepth (MQCFIN)

Maximum queue depth (parameter identifier: MQIA_MAX_Q_DEPTH).

The maximum number of messages allowed on the queue. Note that other factors may cause the queue to be treated as full; for example, it will appear to be full if there is no storage available for a message.

Specify a value greater than or equal to 0, and less than or equal to:

- 999 999 999 if the queue is on AIX, HP-UX, i5/OS, Solaris, Linux, Windows, or z/OS
- 640 000 if the queue is on any other Websphere MQ platform.

MsgDeliverySequence (MQCFIN)

Whether priority is relevant (parameter identifier: MQIA_MSG_DELIVERY_SEQUENCE).

The value can be:

MQMDS_PRIORITY

Messages are returned in priority order.

MQMDS_FIFO

Messages are returned in FIFO order (first in, first out).

NonPersistentMessageClass (MQCFIN)

The level of reliability to be assigned to non-persistent messages that are put to the queue (parameter identifier: MQIA_NPM_CLASS).

The value can be:

MQNPM_CLASS_NORMAL

Non-persistent messages persist as long as the lifetime of the queue manager session. They are discarded in the event of a queue manager restart. This is the default value.

MQNPM_CLASS_HIGH

The queue manager attempts to retain non-persistent messages for the lifetime of the queue. Non-persistent messages may still be lost in the event of a failure.

This parameter is valid only on local and model queues. It is not valid on z/OS.

ProcessName (MQCFST)

Name of process definition for the queue (parameter identifier: MQCA_PROCESS_NAME).

Specifies the local name of the WebSphere MQ process that identifies the application to be started when a trigger event occurs.

- If the queue is a transmission queue, the process definition contains the name of the channel to be started. This parameter is optional for transmission queues on AIX, HP OpenVMS, HP-UX, Linux, i5/OS, Solaris, Windows, and z/OS; if you do not specify it, the channel name is taken from the value specified for the *TriggerData* parameter.

- In other environments, the process name must be nonblank for a trigger event to occur (although it can be set after the queue has been created).

The maximum length of the string is MQ_PROCESS_NAME_LENGTH.

QDepthHighEvent (MQCFIN)

Controls whether Queue Depth High events are generated (parameter identifier: MQIA_Q_DEPTH_HIGH_EVENT).

A Queue Depth High event indicates that an application has put a message on a queue, and this has caused the number of messages on the queue to become greater than or equal to the queue depth high threshold. See the *QDepthHighLimit* parameter.

Note: The value of this attribute can change implicitly. See Chapter 3, “Definitions of the Programmable Command Formats,” on page 25.

The value can be:

MQEVR_DISABLED

Event reporting disabled.

MQEVR_ENABLED

Event reporting enabled.

QDepthHighLimit (MQCFIN)

High limit for queue depth (parameter identifier: MQIA_Q_DEPTH_HIGH_LIMIT).

The threshold against which the queue depth is compared to generate a Queue Depth High event.

This event indicates that an application has put a message to a queue, and this has caused the number of messages on the queue to become greater than or equal to the queue depth high threshold. See the *QDepthHighEvent* parameter.

The value is expressed as a percentage of the maximum queue depth (*MaxQDepth* attribute), and must be greater than or equal to zero and less than or equal to 100.

QDepthLowEvent (MQCFIN)

Controls whether Queue Depth Low events are generated (parameter identifier: MQIA_Q_DEPTH_LOW_EVENT).

A Queue Depth Low event indicates that an application has retrieved a message from a queue, and this has caused the number of messages on the queue to become less than or equal to the queue depth low threshold. See the *QDepthLowLimit* parameter.

Note: The value of this attribute can change implicitly. See Chapter 3, “Definitions of the Programmable Command Formats,” on page 25.

The value can be:

MQEVR_DISABLED

Event reporting disabled.

MQEVR_ENABLED

Event reporting enabled.

QDepthLowLimit (MQCFIN)

Low limit for queue depth (parameter identifier: MQIA_Q_DEPTH_LOW_LIMIT).

The threshold against which the queue depth is compared to generate a Queue Depth Low event.

This event indicates that an application has retrieved a message from a queue, and this has caused the number of messages on the queue to become less than or equal to the queue depth low threshold. See the *QDepthLowEvent* parameter.

Specify the value as a percentage of the maximum queue depth (*MaxQDepth* attribute), in the range 0 through 100.

QDepthMaxEvent (MQCFIN)

Controls whether Queue Full events are generated (parameter identifier: MQIA_Q_DEPTH_MAX_EVENT).

A Queue Full event indicates that an **MQPUT** call to a queue has been rejected because the queue is full, that is, the queue depth has already reached its maximum value.

Note: The value of this attribute can change implicitly. See Chapter 3, "Definitions of the Programmable Command Formats," on page 25.

The value can be:

MQEVR_DISABLED

Event reporting disabled.

MQEVR_ENABLED

Event reporting enabled.

QDesc (MQCFST)

Queue description (parameter identifier: MQCA_Q_DESC).

Text that briefly describes the object.

The maximum length of the string is MQ_Q_DESC_LENGTH.

Use characters from the character set identified by the coded character set identifier (CCSID) for the message queue manager on which the command is executing to ensure that the text is translated correctly if it is sent to another queue manager.

QServiceInterval (MQCFIN)

Target for queue service interval (parameter identifier: MQIA_Q_SERVICE_INTERVAL).

The service interval used for comparison to generate Queue Service Interval High and Queue Service Interval OK events. See the *QServiceIntervalEvent* parameter.

Specify a value in the range 0 through 999 999 999 milliseconds.

QServiceIntervalEvent (MQCFIN)

Controls whether Service Interval High or Service Interval OK events are generated (parameter identifier: MQIA_Q_SERVICE_INTERVAL_EVENT).

A Queue Service Interval High event is generated when a check indicates that no messages have been retrieved from or put to the queue for at least the time indicated by the *QServiceInterval* attribute.

A Queue Service Interval OK event is generated when a check indicates that a message has been retrieved from the queue within the time indicated by the *QServiceInterval* attribute.

Note: The value of this attribute can change implicitly. See Chapter 3, "Definitions of the Programmable Command Formats," on page 25.

The value can be:

MQQSIE_HIGH

Queue Service Interval High events enabled.

- Queue Service Interval High events are **enabled** and
- Queue Service Interval OK events are **disabled**.

MQQSIE_OK

Queue Service Interval OK events enabled.

- Queue Service Interval High events are **disabled** and
- Queue Service Interval OK events are **enabled**.

MQQSIE_NONE

No queue service interval events enabled.

- Queue Service Interval High events are **disabled** and
- Queue Service Interval OK events are also **disabled**.

QSGDisposition (MQCFIN)

Disposition of the object within the group (parameter identifier: MQIA_QSG_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves). The value can be:

QSGDisposition	Change	Copy, Create
MQQSGD_COPY	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameter MQQSGD_COPY. Any object residing in the shared repository, or any object defined using a command that had the parameters MQQSGD_Q_MGR, is not affected by this command.	The object is defined on the page set of the queue manager that executes the command using the MQQSGD_GROUP object of the same name as the <i>ToQName</i> object (for Copy) or the <i>QName</i> object (for Create). For local queues, messages are stored on the page sets of each queue manager and are available only through that queue manager.
MQQSGD_GROUP	<p>The object definition resides in the shared repository. The object was defined using a command that had the parameter MQQSGD_GROUP. Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command.</p> <p>If the command is successful, the following MQSC command is generated and sent to all active queue managers in the queue-sharing group to attempt to refresh local copies on page set zero:</p> <pre>DEFINE QUEUE(q-name) REPLACE QSGDISP(COPY)</pre> <p>The Change for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.</p>	<p>The object definition resides in the shared repository. This is allowed only in a shared queue manager environment.</p> <p>If the definition is successful, the following MQSC command is generated and sent to all active queue managers to attempt to make or refresh local copies on page set zero:</p> <pre>DEFINE QUEUE(q-name) REPLACE QSGDISP(COPY)</pre> <p>The Copy or Create for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.</p>

QSGDisposition	Change	Copy, Create
MQQSGD_PRIVATE	The object resides on the page set of the queue manager that executes the command, and was defined with MQQSGD_Q_MGR or MQQSGD_COPY. Any object residing in the shared repository is unaffected.	Not permitted.
MQQSGD_Q_MGR	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameter MQQSGD_Q_MGR. Any object residing in the shared repository, or any local copy of such an object, is not affected by this command. This is the default value.	The object is defined on the page set of the queue manager that executes the command. This is the default value. For local queues, messages are stored on the page sets of each queue manager and are available only through that queue manager.
MQQSGD_SHARED	This value applies only to local queues. The object definition resides in the shared repository. The object was defined by a command using the parameter MQQSGD_SHARED. Any object residing on the page set of the queue manager that executes the command, or any object defined by a command using the parameter MQQSGD_GROUP, is not affected by this command.	This option applies only to local queues. The object is defined in the shared repository. Messages are stored in the Coupling Facility and are available to any queue manager in the queue-sharing group. You can specify MQQSGD_SHARED only if: <ul style="list-style-type: none"> • <i>CFStructure</i> is nonblank • <i>IndexType</i> is not MQIT_MSG_TOKEN • The queue is not one of the following: <ul style="list-style-type: none"> – SYSTEM.CHANNEL.INITQ – SYSTEM.COMMAND.INPUT

QueueAccounting (MQCFIN)

Controls the collection of accounting data (parameter identifier: MQIA_ACCOUNTING_Q).

The value can be:

MQMON_Q_MGR

The collection of accounting data for the queue is performed based upon the setting of the *QueueAccounting* parameter on the queue manager.

MQMON_OFF

Accounting data collection is disabled for the queue.

MQMON_ON

If the value of the queue manager's *QueueAccounting* parameter is not MQMON_NONE, accounting data collection is enabled for the queue.

QueueMonitoring (MQCFIN)

Online monitoring data collection (parameter identifier: MQIA_MONITORING_Q).

Specifies whether online monitoring data is to be collected and, if so, the rate at which the data is collected. The value can be:

MQMON_OFF

Online monitoring data collection is turned off for this queue.

MQMON_Q_MGR

The value of the queue manager's *QueueMonitoring* parameter is inherited by the queue.

MQMON_LOW

If the value of the queue manager's *QueueMonitoring* parameter is not MQMON_NONE, online monitoring data collection is turned on, with a low rate of data collection, for this queue.

MQMON_MEDIUM

If the value of the queue manager's *QueueMonitoring* parameter is not MQMON_NONE, online monitoring data collection is turned on, with a moderate rate of data collection, for this queue.

MQMON_HIGH

If the value of the queue manager's *QueueMonitoring* parameter is not MQMON_NONE, online monitoring data collection is turned on, with a high rate of data collection, for this queue.

***QueueStatistics* (MQCFIN)**

Statistics data collection (parameter identifier: MQIA_STATISTICS_Q).

Specifies whether statistics data collection is enabled. The value can be:

MQMON_Q_MGR

The value of the queue manager's *QueueStatistics* parameter is inherited by the queue.

MQMON_OFF

Statistics data collection is disabled

MQMON_ON

If the value of the queue manager's *QueueStatistics* parameter is not MQMON_NONE, statistics data collection is enabled

This parameter is valid only on i5/OS, UNIX systems, and Windows.

***RemoteQMgrName* (MQCFST)**

Name of remote queue manager (parameter identifier: MQCA_REMOTE_Q_MGR_NAME).

If an application opens the local definition of a remote queue, *RemoteQMgrName* must not be blank or the name of the connected queue manager. If *XmitQName* is blank there must be a local queue of this name, which is to be used as the transmission queue.

If this definition is used for a queue-manager alias, *RemoteQMgrName* is the name of the queue manager, which can be the name of the connected queue manager. Otherwise, if *XmitQName* is blank, when the queue is opened there must be a local queue of this name, which is to be used as the transmission queue.

If this definition is used for a reply-to alias, this name is the name of the queue manager that is to be the reply-to queue manager.

The maximum length of the string is MQ_Q_MGR_NAME_LENGTH.

***RemoteQName* (MQCFST)**

Name of remote queue as known locally on the remote queue manager (parameter identifier: MQCA_REMOTE_Q_NAME).

If this definition is used for a local definition of a remote queue, *RemoteQName* must not be blank when the open occurs.

If this definition is used for a queue-manager alias definition, *RemoteQName* must be blank when the open occurs.

If this definition is used for a reply-to alias, this name is the name of the queue that is to be the reply-to queue.

The maximum length of the string is MQ_Q_NAME_LENGTH.

Replace (MQCFIN)

Replace attributes (parameter identifier: MQIACF_REPLACE). This parameter is not valid on a Change Queue command.

If the object already exists, the effect is similar to issuing the Change Queue command without the MQFC_YES option on the *Force* parameter, and with *all* of the other attributes specified. In particular, note that any messages which are on the existing queue are retained.

(The difference between the Change Queue command without MQFC_YES on the *Force* parameter, and the Create Queue command with MQRP_YES on the *Replace* parameter, is that the Change Queue command does not change unspecified attributes, but Create Queue with MQRP_YES sets *all* the attributes. When you use MQRP_YES, unspecified attributes are taken from the default definition, and the attributes of the object being replaced, if one exists, are ignored.)

The command fails if both of the following are true:

- The command sets attributes that would require the use of MQFC_YES on the *Force* parameter if you were using the Change Queue command
- The object is open

The Change Queue command with MQFC_YES on the *Force* parameter succeeds in this situation.

If MQSCO_CELL is specified on the *Scope* parameter on OS/2 or UNIX systems, and there is already a queue with the same name in the cell directory, the command fails, whether or not MQRP_YES is specified.

The value can be:

MQRP_YES

Replace existing definition.

MQRP_NO

Do not replace existing definition.

RetentionInterval (MQCFIN)

Retention interval (parameter identifier: MQIA_RETENTION_INTERVAL).

The number of hours for which the queue might be needed, based on the date and time when the queue was created.

This information is available to a housekeeping application or an operator and can be used to determine when a queue is no longer required. The queue manager does not delete queues nor does it prevent queues from being deleted if their retention interval has not expired. It is the user's responsibility to take any required action.

Specify a value in the range 0 through 999 999 999.

Scope (MQCFIN)

Scope of the queue definition (parameter identifier: MQIA_SCOPE).

Specifies whether the scope of the queue definition does not extend beyond the queue manager which owns the queue, or whether the queue name is contained in a cell directory, so that it is known to all of the queue managers within the cell.

If this attribute is changed from MQSCO_CELL to MQSCO_Q_MGR, the entry for the queue is deleted from the cell directory.

Model and dynamic queues cannot be changed to have cell scope.

If it is changed from MQSCO_Q_MGR to MQSCO_CELL, an entry for the queue is created in the cell directory. If there is already a queue with the same name in the cell directory, the command fails. The command also fails if no name service supporting a cell directory has been configured.

The value can be:

MQSCO_Q_MGR

Queue-manager scope.

MQSCO_CELL

Cell scope.

This value is not supported on i5/OS.

This parameter is not available on z/OS.

Shareability (MQCFIN)

Whether the queue can be shared (parameter identifier: MQIA_SHAREABILITY).

Specifies whether multiple instances of applications can open this queue for input.

The value can be:

MQQA_SHAREABLE

Queue is shareable.

MQQA_NOT_SHAREABLE

Queue is not shareable.

StorageClass (MQCFST)

Storage class (parameter identifier: MQCA_STORAGE_CLASS). This parameter applies to z/OS only.

Specifies the name of the storage class.

The maximum length of the string is MQ_STORAGE_CLASS_LENGTH.

TriggerControl (MQCFIN)

Trigger control (parameter identifier: MQIA_TRIGGER_CONTROL).

Specifies whether trigger messages are written to the initiation queue.

The value can be:

MQTC_OFF

Trigger messages not required.

MQTC_ON

Trigger messages required.

TriggerData (MQCFST)

Trigger data (parameter identifier: MQCA_TRIGGER_DATA).

Specifies user data that the queue manager includes in the trigger message. This data is made available to the monitoring application that processes the initiation queue and to the application that is started by the monitor.

The maximum length of the string is MQ_TRIGGER_DATA_LENGTH.

TriggerDepth (MQCFIN)

Trigger depth (parameter identifier: MQIA_TRIGGER_DEPTH).

Specifies (when *TriggerType* is MQTT_DEPTH) the number of messages that will initiate a trigger message to the initiation queue. The value must be in the range 1 through 999 999 999.

TriggerMsgPriority (MQCFIN)

Threshold message priority for triggers (parameter identifier: MQIA_TRIGGER_MSG_PRIORITY).

Specifies the minimum priority that a message must have before it can cause, or be counted for, a trigger event. The value must be in the range of priority values that is supported (0 through 9).

TriggerType (MQCFIN)

Trigger type (parameter identifier: MQIA_TRIGGER_TYPE).

Specifies the condition that initiates a trigger event. When the condition is true, a trigger message is sent to the initiation queue.

The value can be:

MQTT_NONE

No trigger messages.

MQTT_EVERY

Trigger message for every message.

MQTT_FIRST

Trigger message when queue depth goes from 0 to 1.

MQTT_DEPTH

Trigger message when depth threshold exceeded.

Usage (MQCFIN)

Usage (parameter identifier: MQIA_USAGE).

Specifies whether the queue is for normal usage or for transmitting messages to a remote message queue manager.

The value can be:

MQUS_NORMAL

Normal usage.

MQUS_TRANSMISSION

Transmission queue.

XmitQName (MQCFST)

Transmission queue name (parameter identifier: MQCA_XMIT_Q_NAME).

Specifies the local name of the transmission queue to be used for messages destined for either a remote queue or for a queue-manager alias definition.

If *XmitQName* is blank, a queue with the same name as *RemoteQMgrName* is used as the transmission queue.

This attribute is ignored if the definition is being used as a queue-manager alias and *RemoteQMgrName* is the name of the connected queue manager.

It is also ignored if the definition is used as a reply-to queue alias definition.
The maximum length of the string is MQ_Q_NAME_LENGTH.

Error codes

This command might return the following in the response format header, in addition to the values shown on page “Error codes applicable to all commands” on page 27.

Reason (MQLONG)

The value can be:

MQRCCF_CELL_DIR_NOT_AVAILABLE

Cell directory is not available.

MQRCCF_CLUSTER_NAME_CONFLICT

Cluster name conflict.

MQRCCF_CLUSTER_Q_USAGE_ERROR

Cluster usage conflict.

MQRCCF_DYNAMIC_Q_SCOPE_ERROR

Dynamic queue scope error.

MQRCCF_FORCE_VALUE_ERROR

Force value not valid.

MQRCCF_Q_ALREADY_IN_CELL

Queue already exists in cell.

MQRCCF_Q_TYPE_ERROR

Queue type not valid.

Change Queue Manager

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Change Queue Manager (MQCMD_CHANGE_Q_MGR) command changes the specified attributes of the queue manager.

For any optional parameters that are omitted, the value does not change.

Required parameters:

None

Optional parameters:

*AccountingConnOverride, AccountingInterval, ActivityRecording,
AdoptNewMCACheck, AdoptNewMCAType, AuthorityEvent, BridgeEvent,
ChannelAutoDef, ChannelAutoDefEvent, ChannelAutoDefExit, ChannelEvent,
ChannelInitiatorControl, ChannelMonitoring, ChannelStatistics,
ChinitAdapters, ChinitDispatchers, ChinitServiceParm,
ChinitTraceAutoStart, ChinitTraceTableSize,
ClusterSenderMonitoringDefault, ClusterSenderStatistics,
ClusterWorkloadData, ClusterWorkloadExit, ClusterWorkloadLength,
CLWLMRUCHannels, CLWLUseQ, CodedCharSetId, CommandEvent, CommandScope,
CommandServerControl, ConfigurationEvent, DeadLetterQName,*

DefXmitQName, DNSGroup, DNSWLM, ExpiryInterval, Force, IGQPutAuthority, IGQUserId, InhibitEvent, IntraGroupQueuing, IPAddressVersion, ListenerTimer, LocalEvent, LoggerEvent, LUGroupName, LUName, LU62ARMSuffix, LU62Channels, MaxActiveChannels, MaxChannels, MaxHandles, MaxMsgLength, MaxUncommittedMsgs, MQIAccounting, MQIStatistics, OutboundPortMax, OutboundPortMin, PerformanceEvent, QMgrDesc, QueueAccounting, , QueueMonitoring, QueueStatistics, ReceiveTimeout, ReceiveTimeoutMin, ReceiveTimeoutType, RemoteEvent, RepositoryName, RepositoryNamelist, , SharedQMgrName, SSLCRLNamelist, SSLCryptoHardware, SSLEvent, SSLFipsRequired, SSLKeyRepository, SSLKeyResetCount, SSLTasks, StartStopEvent, StatisticsInterval, TCPChannels, TCPKeepAlive, TCPName, TCPStackType, TraceRouteRecording, TriggerInterval

Optional parameters

AccountingConnOverride (MQCFIN)

Specifies whether applications can override the settings of the *QueueAccounting* and *MQIAccounting* queue manager parameters (parameter identifier: MQIA_ACCOUNTING_CONN_OVERRIDE).

The value can be:

MQMON_DISABLED

Applications cannot override the settings of the *QueueAccounting* and *MQIAccounting* parameters.

This is the queue manager's initial default value.

MQMON_ENABLED

Applications can override the settings of the *QueueAccounting* and *MQIAccounting* parameters by using the options field of the MQCNO structure of the MQCONNX API call.

This parameter is valid only on AIX, HP-UX, Linux, i5/OS, Solaris, and Windows.

AccountingInterval (MQCFIN)

The time interval, in seconds, at which intermediate accounting records are written (parameter identifier: MQIA_ACCOUNTING_INTERVAL).

Specify a value in the range 1 through 604 000.

This parameter is valid only on AIX, HP-UX, Linux, i5/OS, Solaris, and Windows.

ActivityRecording (MQCFIN)

Whether activity reports can be generated (parameter identifier: MQIA_ACTIVITY_RECORDING).

The value can be:

MQRECORDING_DISABLED

Activity reports cannot be generated.

MQRECORDING_MSG

Activity reports can be generated and sent to the reply queue specified by the originator in the message causing the report.

MQRECORDING_Q

Activity reports can be generated and sent to SYSTEM.ADMIN.ACTIVITY.QUEUE.

For more information about activity reports, see *WebSphere MQ Monitoring*.

AdoptNewMCACheck (MQCFIN)

The elements checked to determine whether an MCA should be adopted (restarted) when a new inbound channel is detected that has the same name as a currently active MCA (parameter identifier: MQIA_ADOPTNEWMCA_CHECK).

The value can be:

MQADOPT_CHECK_Q_MGR_NAME

Check the queue manager name.

MQADOPT_CHECK_NET_ADDR

Check the network address.

MQADOPT_CHECK_ALL

Check the queue manager name and network address. Perform this check to prevent your channels from being inadvertently shut down. This is the queue manager's initial default value.

MQADOPT_CHECK_NONE

Do not check any elements.

This parameter applies to z/OS only.

AdoptNewMCAType (MQCFIN)

Adoption of orphaned channel instances (parameter identifier: MQIA_ADOPTNEWMCA_TYPE).

Specify whether an orphaned MCA instance is to be adopted when a new inbound channel request is detected matching the *AdoptNewMCACheck* parameters.

The value can be:

MQADOPT_TYPE_NO

Do not adopt orphaned channel instances.

MQADOPT_TYPE_ALL

Adopt all channel types. This is the queue manager's initial default value.

This parameter applies to z/OS only.

AuthorityEvent (MQCFIN)

Controls whether authorization (Not Authorized) events are generated (parameter identifier: MQIA_AUTHORITY_EVENT).

The value can be:

MQEVR_DISABLED

Event reporting disabled.

MQEVR_ENABLED

Event reporting enabled. This value is not permitted on z/OS.

BridgeEvent (MQCFIN)

Controls whether IMSTM Bridge events are generated (parameter identifier: MQIA_BRIDGE_EVENT). This parameter applies to z/OS only.

The value can be:

MQEVR_DISABLED

Event reporting disabled. This is the default value.

MQEVR_ENABLED

Event reporting enabled. This value is not supported on z/OS.

ChannelAutoDef (MQCFIN)

Controls whether receiver and server-connection channels can be auto-defined (parameter identifier: MQIA_CHANNEL_AUTO_DEF).

Auto-definition for cluster-sender channels is always enabled.

This parameter is supported in the following environments: AIX, HP-UX, i5/OS, Solaris, Windows and Linux.

The value can be:

MQCHAD_DISABLED

Channel auto-definition disabled.

MQCHAD_ENABLED

Channel auto-definition enabled.

ChannelAutoDefEvent (MQCFIN)

Controls whether channel auto-definition events are generated (parameter identifier: MQIA_CHANNEL_AUTO_DEF_EVENT), when a receiver, server-connection, or cluster-sender channel is auto-defined.

This parameter is supported in the following environments: AIX, HP-UX, i5/OS, Solaris, Windows and Linux.

The value can be:

MQEVR_DISABLED

Event reporting disabled.

MQEVR_ENABLED

Event reporting enabled.

ChannelAutoDefExit (MQCFST)

Channel auto-definition exit name (parameter identifier: MQCA_CHANNEL_AUTO_DEF_EXIT).

This exit is invoked when an inbound request for an undefined channel is received, if:

1. The channel is a cluster-sender, or
2. Channel auto-definition is enabled (see *ChannelAutoDef*).

This exit is also invoked when a cluster-receiver channel is started.

The format of the name is the same as for the *SecurityExit* parameter described in “Change, Copy and Create Channel” on page 41.

The maximum length of the exit name depends on the environment in which the exit is running. MQ_EXIT_NAME_LENGTH gives the maximum length for the environment in which your application is running.

MQ_MAX_EXIT_NAME_LENGTH gives the maximum for all supported environments.

This parameter is supported in the following environments: AIX, HP-UX, i5/OS, Solaris, Windows, Linux, and z/OS. On z/OS, it applies only to cluster-sender and cluster-receiver channels.

ChannelEvent (MQCFIN)

Controls whether channel events are generated (parameter identifier: MQIA_CHANNEL_EVENT).

The value can be:

MQEVR_DISABLED

Event reporting disabled.

MQEVR_ENABLED

Event reporting enabled.

MQEVR_EXCEPTION

Reporting of exception channel events enabled.

ChannelInitiatorControl (MQCFIN)

Specifies whether the channel initiator is to be started when the queue manager starts (parameter identifier: MQIA_CHINIT_CONTROL).

The value can be:

MQSVC_CONTROL_MANUAL

The channel initiator is not to be started automatically.

MQSVC_CONTROL_Q_MGR

The channel initiator is to be started automatically when the queue manager starts.

This parameter is valid only on AIX, HP-UX, Linux, i5/OS, Solaris, and Windows.

ChannelMonitoring (MQCFIN)

Default setting for online monitoring for channels (parameter identifier: MQIA_MONITORING_CHANNEL).

The value can be:

MQMON_NONE

Online monitoring data collection is turned off for channels regardless of the setting of their *ChannelMonitoring* parameter.

MQMON_OFF

Online monitoring data collection is turned off for channels specifying a value of MQMON_Q_MGR in their *ChannelMonitoring* parameter.
This is the queue manager's initial default value.

MQMON_LOW

Online monitoring data collection is turned on, with a low ratio of data collection, for channels specifying a value of MQMON_Q_MGR in their *ChannelMonitoring* parameter.

MQMON_MEDIUM

Online monitoring data collection is turned on, with a moderate ratio of data collection, for channels specifying a value of MQMON_Q_MGR in their *ChannelMonitoring* parameter.

MQMON_HIGH

Online monitoring data collection is turned on, with a high ratio of data collection, for channels specifying a value of MQMON_Q_MGR in their *ChannelMonitoring* parameter.

ChannelStatistics (MQCFIN)

Controls whether statistics data is to be collected for channels (parameter identifier: MQIA_STATISTICS_CHANNEL).

The value can be:

MQMON_NONE

Statistics data collection is turned off for channels regardless of the setting of their *ChannelStatistics* parameter. This is the queue manager's initial default value.

MQMON_OFF

Statistics data collection is turned off for channels specifying a value of MQMON_Q_MGR in their *ChannelStatistics* parameter.

MQMON_LOW

Statistics data collection is turned on, with a low ratio of data collection, for channels specifying a value of MQMON_Q_MGR in their *ChannelStatistics* parameter.

MQMON_MEDIUM

Statistics data collection is turned on, with a moderate ratio of data collection, for channels specifying a value of MQMON_Q_MGR in their *ChannelStatistics* parameter.

MQMON_HIGH

Statistics data collection is turned on, with a high ratio of data collection, for channels specifying a value of MQMON_Q_MGR in their *ChannelStatistics* parameter.

This parameter is valid only on AIX, HP-UX, Linux, i5/OS, Solaris, and Windows.

***ChinitAdapters* (MQCFIN)**

Number of adapter subtasks (parameter identifier: MQIA_CHINIT_ADAPTERS).

The number of adapter subtasks to use for processing WebSphere MQ calls. This parameter applies to z/OS only.

Specify a value in the range 1 through 9 999. The queue manager's initial default value is 8.

***ChinitDispatchers* (MQCFIN)**

Number of dispatchers (parameter identifier: MQIA_CHINIT_DISPATCHERS).

The number of dispatchers to use for the channel initiator. This parameter applies to z/OS only.

Specify a value in the range 1 through 9 999. The queue manager's initial default value is 5.

***ChinitServiceParm* (MQCFST)**

Reserved for use by IBM (parameter identifier: MQCA_CHINIT_SERVICE_PARM).

This parameter applies to z/OS only.

***ChinitTraceAutoStart* (MQCFIN)**

Whether the channel initiator trace should start automatically (parameter identifier: MQIA_CHINIT_TRACE_AUTO_START).

The value can be:

MQTRAXSTR_YES

Channel initiator trace is to start automatically.

MQTRAXSTR_NO

Channel initiator trace is not to start automatically. This is the queue manager's initial default value.

This parameter applies to z/OS only.

ChinitTraceTableSize (MQCFIN)

The size, in megabytes, of the channel initiator's trace data space (parameter identifier: MQIA_CHINIT_TRACE_TABLE_SIZE).

Specify a value in the range 2 through 2048. The queue manager's initial default value is 2.

This parameter applies to z/OS only.

ClusterSenderMonitoringDefault (MQCFIN)

Default setting for online monitoring for automatically defined cluster-sender channels (parameter identifier: MQIA_MONITORING_AUTO_CLUSSDR).

Specifies the value to be used for the *ChannelMonitoring* attribute of automatically defined cluster-sender channels. The value can be:

MQMON_Q_MGR

Collection of online monitoring data is inherited from the setting of the queue manager's *ChannelMonitoring* parameter. This is the queue manager's initial default value.

MQMON_OFF

Monitoring for the channel is switched off.

MQMON_LOW

Unless *ChannelMonitoring* is MQMON_NONE, this specifies a low rate of data collection with a minimal impact on system performance. The data collected is not likely to be the most current.

MQMON_MEDIUM

Unless *ChannelMonitoring* is MQMON_NONE, this specifies a moderate rate of data collection with limited impact on system performance.

MQMON_HIGH

Unless *ChannelMonitoring* is MQMON_NONE, this specifies a high rate of data collection with a likely impact on system performance. The data collected is the most current available.

ClusterSenderStatistics (MQCFIN)

Controls whether statistics data is to be collected for auto-defined cluster-sender channels (parameter identifier: MQIA_STATISTICS_AUTO_CLUSSDR).

The value can be:

MQMON_Q_MGR

Collection of statistics data is inherited from the setting of the queue manager's *ChannelStatistics* parameter. This is the queue manager's initial default value.

MQMON_OFF

Statistics data collection for the channel is switched off.

MQMON_LOW

Unless *ChannelStatistics* is MQMON_NONE, this specifies a low rate of data collection with a minimal impact on system performance.

MQMON_MEDIUM

Unless *ChannelStatistics* is MQMON_NONE, this specifies a moderate rate of data collection.

MQMON_HIGH

Unless *ChannelStatistics* is MQMON_NONE, this specifies a high rate of data collection.

This parameter is valid only on AIX, HP-UX, Linux, i5/OS, Solaris, and Windows.

***ClusterWorkLoadData* (MQCFST)**

Cluster workload exit data (parameter identifier: MQCA_CLUSTER_WORKLOAD_DATA).

This is passed to the cluster workload exit when it is called.

The maximum length of the string is MQ_EXIT_DATA_LENGTH.

***ClusterWorkLoadExit* (MQCFST)**

Cluster workload exit name (parameter identifier: MQCA_CLUSTER_WORKLOAD_EXIT).

If a nonblank name is defined this exit is invoked when a message is put to a cluster queue.

The format of the name is the same as for the *SecurityExit* parameter described in “Change, Copy and Create Channel” on page 41.

The maximum length of the exit name depends on the environment in which the exit is running. MQ_EXIT_NAME_LENGTH gives the maximum length for the environment in which your application is running.

MQ_MAX_EXIT_NAME_LENGTH gives the maximum for all supported environments.

This parameter is supported only in the environments in which an MQSeries Version 5.1 product, or later, is available.

***ClusterWorkLoadLength* (MQCFIN)**

Cluster workload length (parameter identifier: MQIA_CLUSTER_WORKLOAD_LENGTH).

The maximum length of the message passed to the cluster workload exit.

The value of this attribute must be in the range 0 through 999 999 999.

***CLWLMRUCHannels* (MQCFIN)**

Cluster workload most recently used (MRU) channels (parameter identifier: MQIA_CLWL_MRU_CHANNELS).

The maximum number of active most recently used outbound channels.

Specify a value in the in the range 1 through 999 999 999.

***CLWLUseQ* (MQCFIN)**

Use of remote queue (parameter identifier: MQIA_CLWL_USEQ).

Specifies whether a cluster queue manager is to use remote puts to other queues defined in other queue managers within the cluster during workload management.

Specify either:

MQCLWL_USEQ_ANY

Use remote queues.

MQCLWL_USEQ_LOCAL

Do not use remote queues.

CodedCharSetId (**MQCFIN**)

Queue manager coded character set identifier (parameter identifier: MQIA_CODED_CHAR_SET_ID).

The coded character set identifier (CCSID) for the queue manager. The CCSID is the identifier used with all character string fields defined by the application programming interface (API). It does not apply to application data carried in the text of a message unless the CCSID in the message descriptor, when the message is put with an MQPUT or MQPUT1, is set to the value MQCCSI_Q_MGR.

Specify a value in the range 1 through 65 535.

The CCSID must specify a value that is defined for use on the platform and use an appropriate character set. The character set must be:

- EBCDIC on i5/OS
- ASCII or ASCII-related on other platforms

Stop and restart the queue manager after execution of this command so that all processes reflect the changed CCSID of the queue manager.

This parameter is supported in the following environments: AIX, Compaq NonStop Kernel, HP OpenVMS, HP-UX, i5/OS, Solaris, Windows and Linux.

CommandEvent (**MQCFIN**)

Controls whether command events are generated (parameter identifier: MQIA_COMMAND_EVENT). This parameter applies to z/OS only.

The value can be:

MQEVR_DISABLED

Event reporting disabled.

MQEVR_ENABLED

Event reporting enabled.

MQEVR_NO_DISPLAY

Event reporting enabled for all successful commands except Inquire commands.

CommandScope (**MQCFST**)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

CommandServerControl (**MQCFIN**)

Specifies whether the command server is to be started when the queue manager starts (parameter identifier: MQIA_CMD_SERVER_CONTROL).

The value can be:

MQSVC_CONTROL_MANUAL

The command server is not to be started automatically.

MQSVC_CONTROL_Q_MGR

The command server is to be started automatically when the queue manager starts.

This parameter is valid only on AIX, HP-UX, Linux, i5/OS, Solaris, and Windows.

ConfigurationEvent (**MQCFIN**)

Controls whether configuration events are generated (parameter identifier: MQIA_CONFIGURATION_EVENT). This parameter applies to z/OS only.

The value can be:

MQEVR_DISABLED

Event reporting disabled.

MQEVR_ENABLED

Event reporting enabled.

DeadLetterQName (**MQCFST**)

Dead letter (undelivered message) queue name (parameter identifier: MQCA_DEAD_LETTER_Q_NAME).

Specifies the name of the local queue that is to be used for undelivered messages. Messages are put on this queue if they cannot be routed to their correct destination. The maximum length of the string is MQ_Q_NAME_LENGTH.

DefXmitQName (**MQCFST**)

Default transmission queue name (parameter identifier: MQCA_DEF_XMIT_Q_NAME).

This is the name of the default transmission queue that is used for the transmission of messages to remote queue managers, if there is no other indication of which transmission queue to use.

The maximum length of the string is MQ_Q_NAME_LENGTH.

DNSGroup (**MQCFST**)

DNS group name (parameter identifier: MQCA_DNS_GROUP).

Specify the name of the group that the TCP listener handling inbound transmissions for the queue-sharing group should join when using Workload Manager for Dynamic Domain Name Services support (WLM/DNS). This parameter applies to z/OS only.

The maximum length of the string is MQ_DNS_GROUP_NAME_LENGTH.

DNSWLM (**MQCFIN**)

Controls whether the TCP listener that handles inbound transmissions for the queue-sharing group should register with WLM/DNS: (parameter identifier: MQIA_DNS_WLM).

The value can be:

MQDNSWLM_YES

The listener should register with WLM.

MQDNSWLM_NO

The listener is not to register with WLM. This is the queue manager's initial default value.

This parameter applies to z/OS only.

***ExpiryInterval* (MQCFIN)**

Interval between scans for expired messages (parameter identifier: MQIA_EXPIRY_INTERVAL). This parameter applies to z/OS only.

Specifies the frequency with which the queue manager scans the queues looking for expired messages. Specify a time interval in seconds in the range 1 through 99 999 999, or the following special value:

MQEXPI_OFF

No scans for expired messages.

The minimum scan interval used is 5 seconds, even if you specify a lower value.

***Force* (MQCFIN)**

Force changes (parameter identifier: MQIACF_FORCE).

Specifies whether the command will be forced to complete if both of the following are true:

- *DefXmitQName* is specified, and
- An application has a remote queue open, the resolution for which will be affected by this change.

***IGQPutAuthority* (MQCFIN)**

Command scope (parameter identifier: MQIA_IGQ_PUT_AUTHORITY). This parameter is valid only on z/OS when the queue manager is a member of a queue-sharing group.

Specifies the type of authority checking and, therefore, the user IDs to be used by the IGQ agent (IGQA). This establishes the authority to put messages to a destination queue. The value can be:

MQIGQPA_DEFAULT

Default user identifier is used.

The user identifier used for authorization is the value of the *UserIdentifier* field in the separate MQMD that is associated with the message when the message is on the shared transmission queue. This is the user identifier of the program that placed the message on the shared transmission queue, and is usually the same as the user identifier under which the remote queue manager is running.

If the RESLEVEL profile indicates that more than one user identifier is to be checked, the user identifier of the local IGQ agent (*IGQUserId*) is also checked.

MQIGQPA_CONTEXT

Context user identifier is used.

The user identifier used for authorization is the value of the *UserIdentifier* field in the separate MQMD that is associated with the message when the message is on the shared transmission queue. This is the user identifier of the program that placed the message on the

shared transmission queue, and is usually the same as the user identifier under which the remote queue manager is running.

If the RESLEVEL profile indicates that more than one user identifier is to be checked, the user identifier of the local IGQ agent (*IGQUserId*) and the value of the *UserIdentifier* field in the embedded MQMD are also checked. The latter user identifier is usually the user identifier of the application that originated the message.

MQIGQPA_ONLY_IGQ

Only the IGQ user identifier is used.

The user identifier used for authorization is the user identifier of the local IGQ agent (*IGQUserId*).

If the RESLEVEL profile indicates that more than one user identifier is to be checked, this user identifier is used for all checks.

MQIGQPA_ALTERNATE_OR_IGQ

Alternate user identifier or IGQ-agent user identifier is used.

The user identifier used for authorization is the user identifier of the local IGQ agent (*IGQUserId*).

If the RESLEVEL profile indicates that more than one user identifier is to be checked, the value of the *UserIdentifier* field in the embedded MQMD is also checked. This user identifier is usually the user identifier of the application that originated the message.

IGQUserId (MQCFST)

Intra-group queuing agent user identifier (parameter identifier: MQCA_IGQ_USER_ID). This parameter is valid only on z/OS when the queue manager is a member of a queue-sharing group.

Specifies the user identifier that is associated with the local intra-group queuing agent. This identifier is one of the user identifiers that may be checked for authorization when the IGQ agent puts messages on local queues. The actual user identifiers checked depend on the setting of the *IGQPutAuthority* attribute, and on external security options.

The maximum length is MQ_USER_ID_LENGTH.

InhibitEvent (MQCFIN)

Controls whether inhibit (Inhibit Get and Inhibit Put) events are generated (parameter identifier: MQIA_INHIBIT_EVENT).

The value can be:

MQEVR_DISABLED

Event reporting disabled.

MQEVR_ENABLED

Event reporting enabled.

IntraGroupQueuing (MQCFIN)

Command scope (parameter identifier: MQIA_INTRA_GROUP_QUEUING). This parameter is valid only on z/OS when the queue manager is a member of a queue-sharing group.

Specifies whether intra-group queuing is used. The value can be:

MQIGQ_DISABLED

Intra-group queuing disabled.

MQIGQ_ENABLED

Intra-group queuing enabled.

IPAddressVersion (MQCFIN)

IP address version selector (parameter identifier: MQIA_IP_ADDRESS_VERSION).

Specifies which IP address version, either IPv4 or IPv6, is used. The value can be:

MQIPADDR_IPV4

IPv4 is used.

MQIPADDR_IPV6

IPv6 is used.

This parameter is only relevant for systems that run both IPv4 and IPv6 and only affects channels defined as having a *TransportType* of MQXPY_TCP when one of the following conditions is true:

- The channel's *ConnectionName* is a hostname that resolves to both an IPv4 and IPv6 address and its *LocalAddress* parameter is not specified.
- The channel's *ConnectionName* and *LocalAddress* are both hostnames that resolve to both IPv4 and IPv6 addresses.

ListenerTimer (MQCFIN)

Listener restart interval (parameter identifier: MQIA_LISTENER_TIMER).

The time interval, in seconds, between attempts by WebSphere MQ to restart the listener after an APPC or TCP/IP failure. This parameter applies to z/OS only.

Specify a value in the range 5 through 9 999. The queue manager's initial default value is 60.

LocalEvent (MQCFIN)

Controls whether local error events are generated (parameter identifier: MQIA_LOCAL_EVENT).

The value can be:

MQEVR_DISABLED

Event reporting disabled.

MQEVR_ENABLED

Event reporting enabled.

LoggerEvent (MQCFIN)

Controls whether recovery log events are generated (parameter identifier: MQIA_LOGGER_EVENT).

The value can be:

MQEVR_DISABLED

Event reporting disabled.

MQEVR_ENABLED

Event reporting enabled. This value is valid only on queue managers that use linear logging.

This is valid only on AIX, HP-UX, i5/OS, Solaris, Linux, and Windows.

LUGroupName (**MQCFST**)

Generic LU name for the LU 6.2 listener (parameter identifier: MQCA_LU_GROUP_NAME).

The generic LU name to be used by the LU 6.2 listener that handles inbound transmissions for the queue-sharing group.

This parameter applies to z/OS only.

The maximum length of the string is MQ_LU_NAME_LENGTH.

LUName (**MQCFST**)

LU name to use for outbound LU 6.2 transmissions (parameter identifier: MQCA_LU_NAME).

The name of the LU to use for outbound LU 6.2 transmissions. Set this to be the same as the name of the LU to be used by the listener for inbound transmissions.

This parameter applies to z/OS only.

The maximum length of the string is MQ_LU_NAME_LENGTH.

LU62ARMSuffix (**MQCFST**)

APPCPM suffix (parameter identifier: MQCA_LU62_ARM_SUFFIX).

The suffix of the APPCPM member of SYS1.PARMLIB. This suffix nominates the LUADD for this channel initiator.

This parameter applies to z/OS only.

The maximum length of the string is MQ_ARM_SUFFIX_LENGTH.

LU62Channels (**MQCFIN**)

Maximum number of LU 6.2 channels (parameter identifier: MQIA_LU62_CHANNELS).

The maximum number of channels that can be current, or clients that can be connected, that use the LU 6.2 transmission protocol.

This parameter applies to z/OS only.

Specify a value in the range zero through 9 999. The queue manager's initial default value is 200.

MaxActiveChannels (**MQCFIN**)

Maximum number of channels (parameter identifier: MQIA_ACTIVE_CHANNELS).

The maximum number of channels that can be active at any time.

This parameter applies to z/OS only.

Specify a value in the range 1 through 9 999. The queue manager's initial default value is 200.

MaxChannels (**MQCFIN**)

Maximum number of current channels (parameter identifier: MQIA_MAX_CHANNELS).

The maximum number of channels that can be current (including server-connection channels with connected clients).

This parameter applies to z/OS only.

Specify a value in the range 1 through 9 999.

MaxHandles (MQCFIN)

Maximum number of handles (parameter identifier: MQIA_MAX_HANDLES).

The maximum number of handles that any one connection can have open at the same time.

Specify a value in the range 0 through 999 999 999.

MaxMsgLength (MQCFIN)

Maximum message length (parameter identifier: MQIA_MAX_MSG_LENGTH).

Specifies the maximum length of messages allowed on queues on the queue manager. No message that is larger than either the queue's *MaxMsgLength* or the queue manager's *MaxMsgLength* can be put on a queue.

If you reduce the maximum message length for the queue manager, you must also reduce the maximum message length of the SYSTEM.DEFAULT.LOCAL.QUEUE definition, and your other queues, to ensure that the queue manager's limit is not less than that of any of the queues in the system. If you do not do this, and applications inquire only the value of the queue's *MaxMsgLength*, they might not work correctly.

The lower limit for this parameter is 32 KB (32 768 bytes). The upper limit depends on the environment:

- On AIX, HP OpenVMS, Compaq NonStop Kernel, HP-UX, i5/OS, Solaris, Linux, and Windows, the maximum message length is 100 MB (104 857 600 bytes).
- On UNIX systems not listed above, the maximum message length is 4 MB (4 194 304 bytes).

This parameter is not valid on z/OS.

MaxUncommittedMsgs (MQCFIN)

Maximum uncommitted messages (parameter identifier: MQIA_MAX_UNCOMMITTED_MSGS).

Specifies the maximum number of uncommitted messages. That is:

- The number of messages that can be retrieved, plus
- The number of messages that can be put, plus
- Any trigger messages generated within this unit of work

under any one syncpoint. This limit does not apply to messages that are retrieved or put outside syncpoint.

Specify a value in the range 1 through 10 000.

MQIACounting (MQCFIN)

Controls whether accounting information for MQI data is to be collected (parameter identifier: MQIA_ACCOUNTING_MQI).

The value can be:

MQMON_OFF

MQI accounting data collection is disabled. This is the queue manager's initial default value.

MQMON_ON

MQI accounting data collection is enabled.

This parameter is valid only on AIX, HP-UX, Linux, i5/OS, Solaris, and Windows.

MQIStatistics (MQCFIN)

Controls whether statistics monitoring data is to be collected for the queue manager (parameter identifier: MQIA_STATISTICS_MQI).

The value can be:

MQMON_OFF

Data collection for MQI statistics is disabled. This is the queue manager's initial default value.

MQMON_ON

Data collection for MQI statistics is enabled.

This parameter is valid only on AIX, HP-UX, Linux, i5/OS, Solaris, and Windows.

OutboundPortMax (MQCFIN)

The maximum value in the range for the binding of outgoing channels (parameter identifier: MQIA_OUTBOUND_PORT_MAX).

The maximum value in the range of port numbers to be used when binding outgoing channels. This parameter applies to z/OS only.

Specify a value in the range zero through 65 535. The queue manager's initial default value is zero.

Specify a corresponding value for *OutboundPortMin* and ensure that the value of *OutboundPortMax* is greater than or equal to the value of *OutboundPortMin*.

OutboundPortMin (MQCFIN)

The minimum value in the range for the binding of outgoing channels (parameter identifier: MQIA_OUTBOUND_PORT_MIN).

The minimum value in the range of port numbers to be used when binding outgoing channels. This parameter applies to z/OS only.

Specify a value in the range zero through 65 535. The queue manager's initial default value is zero.

Specify a corresponding value for *OutboundPortMax* and ensure that the value of *OutboundPortMin* is less than or equal to the value of *OutboundPortMax*.

PerformanceEvent (MQCFIN)

Controls whether performance-related events are generated (parameter identifier: MQIA_PERFORMANCE_EVENT).

The value can be:

MQEVR_DISABLED

Event reporting disabled.

MQEVR_ENABLED

Event reporting enabled.

QMgrDesc (MQCFST)

Queue manager description (parameter identifier: MQCA_Q_MGR_DESC).

This is text that briefly describes the object.

The maximum length of the string is MQ_Q_MGR_DESC_LENGTH.

Use characters from the character set identified by the coded character set identifier (CCSID) for the queue manager on which the command is executing, to ensure that the text is translated correctly.

QueueAccounting (MQCFIN)

Controls the collection of accounting (thread-level and queue-level accounting) data for queues (parameter identifier: MQIA_ACCOUNTING_Q).

The value can be:

MQMON_NONE

Accounting data collection for queues is disabled. This may not be overridden by the value of the *QueueAccounting* parameter on the queue.

MQMON_OFF

Accounting data collection is disabled for queues specifying a value of MQMON_Q_MGR in the *QueueAccounting* parameter.

MQMON_ON

Accounting data collection is enabled for queues specifying a value of MQMON_Q_MGR in the *QueueAccounting* parameter.

QueueMonitoring (MQCFIN)

Default setting for online monitoring for queues (parameter identifier: MQIA_MONITORING_Q).

If the *QueueMonitoring* queue attribute is set to MQMON_Q_MGR, this attribute specifies the value which is assumed by the channel. The value can be:

MQMON_OFF

Online monitoring data collection is turned off. This is the queue manager's initial default value.

MQMON_NONE

Online monitoring data collection is turned off for queues regardless of the setting of their *QueueMonitoring* attribute.

MQMON_LOW

Online monitoring data collection is turned on, with a low ratio of data collection.

MQMON_MEDIUM

Online monitoring data collection is turned on, with a moderate ratio of data collection.

MQMON_HIGH

Online monitoring data collection is turned on, with a high ratio of data collection.

QueueStatistics (MQCFIN)

Controls whether statistics data is to be collected for queues (parameter identifier: MQIA_STATISTICS_Q).

The value can be:

MQMON_NONE

Statistics data collection is turned off for queues regardless of the setting of their *QueueStatistics* parameter. This is the queue manager's initial default value.

MQMON_OFF

Statistics data collection is turned off for queues specifying a value of MQMON_Q_MGR in their *QueueStatistics* parameter.

MQMON_ON

Statistics data collection is turned on for queues specifying a value of MQMON_Q_MGR in their *QueueStatistics* parameter.

This parameter is valid only on AIX, HP-UX, Linux, i5/OS, Solaris, and Windows.

ReceiveTimeout (MQCFIN)

How long a TCP/IP channel waits to receive data from its partner (parameter identifier: MQIA_RECEIVE_TIMEOUT).

The approximate length of time that a TCP/IP channel waits to receive data, including heartbeats, from its partner before returning to the inactive state.

This parameter applies to z/OS only and only to message channels (and not to MQI channels). This number can be qualified as follows:

- To specify that this number is a multiplier to be applied to the negotiated *HeartBeatInterval* value to determine how long a channel is to wait, set *ReceiveTimeoutType* to MQRCVTIME_MULTIPLY. Specify a value of zero or in the range 2 through 99. If you specify zero, the channel does not time out its wait to receive data from its partner.
- To specify that this number is a value, in seconds, to be added to the negotiated *HeartBeatInterval* value to determine how long a channel is to wait, set *ReceiveTimeoutType* to MQRCVTIME_ADD. Specify a value in the range 1 through 999 999.
- To specify that this number is a value, in seconds, that the channel is to wait, set *ReceiveTimeoutType* to MQRCVTIME_EQUAL. Specify a value in the range zero through 999 999. If you specify zero, the channel does not time out its wait to receive data from its partner.

The queue manager's initial default value is zero.

ReceiveTimeoutMin (MQCFIN)

The minimum length of time that a TCP/IP channel waits to receive data from its partner (parameter identifier: MQIA_RECEIVE_TIMEOUT_MIN).

The minimum length of time that a TCP/IP channel waits to receive data, including heartbeats, from its partner before returning to the inactive state. This parameter applies to z/OS only.

Specify a value in the range zero through 999 999.

ReceiveTimeoutType (MQCFIN)

The qualifier to apply to *ReceiveTimeout* (parameter identifier: MQIA_RECEIVE_TIMEOUT_TYPE).

The qualifier to apply to *ReceiveTimeoutType* to calculate how long a TCP/IP channel waits to receive data, including heartbeats, from its partner before returning to the inactive state. This parameter applies to z/OS only.

The value can be:

MQRCVTIME_MULTIPLY

The *ReceiveTimeout* value is a multiplier to be applied to the negotiated value of *HeartbeatInterval* to determine how long a channel will wait. This is the queue manager's initial default value.

MQRCVTIME_ADD

ReceiveTimeout is a value, in seconds, to be added to the negotiated value of *HeartbeatInterval* to determine how long a channel will wait.

MQRCVTIME_EQUAL

ReceiveTimeout is a value, in seconds, representing how long a channel will wait.

RemoteEvent (MQCFIN)

Controls whether remote error events are generated (parameter identifier: MQIA_REMOTE_EVENT).

The value can be:

MQEVR_DISABLED

Event reporting disabled.

MQEVR_ENABLED

Event reporting enabled.

RepositoryName (MQCFST)

Cluster name (parameter identifier: MQCA_REPOSITORY_NAME).

The name of a cluster for which this queue manager provides a repository manager service.

The maximum length of the string is MQ_OBJECT_NAME_LENGTH.

No more than one of the resultant values of *RepositoryName* can be nonblank.

RepositoryNamelist (MQCFST)

Repository namelist (parameter identifier: MQCA_REPOSITORY_NAMELIST).

The name, of a namelist of clusters, for which this queue manager provides a repository manager service.

This queue manager does not have a full repository, but can be a client of other repository services that are defined in the cluster, if

- Both *RepositoryName* and *RepositoryNamelist* are blank, or
- *RepositoryName* is blank and the namelist specified by *RepositoryNamelist* is empty.

No more than one of the resultant values of *RepositoryNameList* can be nonblank.

SharedQMgrName (MQCFIN)

Shared-queue queue manager name (parameter identifier: MQIA_SHARED_Q_Q_MGR_NAME).

When a queue manager makes an MQOPEN call for a shared queue and the queue manager that is specified in the *ObjectQmgrName* parameter of the MQOPEN call is in the same queue-sharing group as the processing queue manager, the SQQMNAME attribute specifies whether the *ObjectQmgrName* is used or whether the processing queue manager opens the shared queue directly. This parameter is valid only on z/OS.

The value can be:

MQSQQM_USE

ObjectQmgrName is used and the appropriate transmission queue is opened.

MQSQQM_IGNORE

The processing queue manager opens the shared queue directly. This can reduce the traffic in your queue manager network.

SSLCTRLNamelist (MQCFST)

The SSL namelist (parameter identifier: MQCA_SSL_CRL_NAMELIST).

The length of the string is MQ_NAMELIST_NAME_LENGTH.

Indicates the name of a namelist of authentication information objects to be used for CRL checking by the queue manager.

If *SSLCRLNamelist* is blank, CRL checking is not invoked.

Changes to *SSLCRLNamelist*, or to the names in a previously specified namelist, or to previously referenced authentication information objects become effective:

- On i5/OS, Windows, and UNIX systems when a new channel process is started.
- For channels that run as threads of the channel initiator on i5/OS, Windows, and UNIX systems, when the channel initiator is restarted.
- For channels that run as threads of the listener on i5/OS, Windows, and UNIX systems, when the listener is restarted.
- On z/OS, when the channel initiator is restarted.
- When a REFRESH SECURITY TYPE(SSL) command is issued.
- On i5/OS queue managers, this parameter is ignored. However, it is used to determine which authentication information objects are written to the AMQCLCHL.TAB file.

SSLCryptoHardware (MQCFST)

The SSL cryptographic hardware (parameter identifier: MQCA_SSL_CRYPTO_HARDWARE).

The length of the string is MQ_SSL_CRYPTO_HARDWARE_LENGTH.

Sets the name of the parameter string required to configure the cryptographic hardware present on the system.

This parameter is supported on AIX, HP-UX, Solaris, Linux, and Windows only.

The string can have one of the following values:

- GSK_ACCELERATOR_RAINBOW_CS_OFF
- GSK_ACCELERATOR_RAINBOW_CS_ON
- GSK_ACCELERATOR_NCIPHER_NF_OFF
- GSK_ACCELERATOR_NCIPHER_NF_ON
- GSK_PKCS11=<the PKCS #11 driver path and filename>;<the PKCS #11 token label>;<the PKCS #11 token password>;<symmetric cipher setting>;

The strings containing RAINBOW enable or disable the Rainbow CryptoSwift cryptographic hardware.

The strings containing NCIPHER enable or disable the nCipher nFast cryptographic hardware.

To use cryptographic hardware which is accessed using the PKCS #11 interface, you must specify the string containing PKCS11. The PKCS #11 driver path is an absolute path to the shared library providing support for the PKCS #11 card. The PKCS #11 driver filename is the name of the shared library. An example of the value required for the PKCS #11 driver path and filename is /usr/lib/pkcs11/PKCS11_API.so

To access symmetric cipher operations through GSKit, specify the symmetric cipher setting parameter on the PKCS11 string. The value of this parameter is either:

SYMMETRIC_CIPHER_OFF

Do not access symmetric cipher operations.

SYMMETRIC_CIPHER_ON

Access symmetric cipher operations.

If the symmetric cipher setting is not specified, this has the same effect as specifying SYMMETRIC_CIPHER_OFF.

The maximum length of the string is 256 characters. The default value is blank.

If you specify a string that does not begin with one of the cryptographic strings listed above, you get an error. If you specify the GSK_PKCS11 string, the syntax of the other parameters is also checked.

When the SSLCryptoHardware value is changed, the cryptographic hardware parameters specified become the ones used for new SSL connection environments. The new information becomes effective:

- When a new channel process is started.
- For channels that run as threads of the channel initiator, when the channel initiator is restarted.
- For channels that run as threads of the listener, when the listener is restarted.
- When a Refresh Security command is issued to refresh the contents of the SSL key repository.

SSLEvent (MQCFIN)

Controls whether SSL events are generated (parameter identifier: MQIA_SSL_EVENT).

The value can be:

MQEVR_DISABLED

Event reporting disabled.

MQEVR_ENABLED

Event reporting enabled.

SSLFipsRequired (MQCFIN)

Specifies whether only FIPS-certified algorithms are to be used if WebSphere MQ itself is to perform cryptography (parameter identifier: MQIA_SSL_FIPS_REQUIRED).

If cryptographic hardware is configured, the cryptographic modules used are those provided by the hardware product, and these may, or may not, be FIPS-certified to a particular level. This depends on the hardware product in use. This parameter applies to Windows and UNIX platforms only.

The value can be:

MQSSL_FIPS_NO

WebSphere MQ provides an implementation of SSL cryptography which supplies some FIPS-certified modules on some platforms. If you set *SSLFipsRequired* to MQSSL_FIPS_NO, any CipherSpec supported on a particular platform can be used. This is the queue manager's initial default value.

If the queue manager runs without using cryptographic hardware, the following CipherSpecs run using FIPS 140-2 certified cryptography:

- TLS_RSA_WITH_DES_CBC_SHA

- TLS_RSA_WITH_3DES_EDE_CBC_SHA
- FIPS_WITH_DES_CBC_SHA
- FIPS_WITH_3DES_EDE_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_256_CBC_SHA

MQSSL_FIPS_YES

Specifies that only FIPS-certified algorithms are to be used in the CipherSpecs allowed on all SSL connections from and to this queue manager.

Inbound and outbound SSL channel connections succeed only if one of the following CipherSpecs is used:

- TLS_RSA_WITH_DES_CBC_SHA
- TLS_RSA_WITH_3DES_EDE_CBC_SHA
- FIPS_WITH_DES_CBC_SHA
- FIPS_WITH_3DES_EDE_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_256_CBC_SHA

SSLKeyRepository (MQCFST)

The SSL key repository (parameter identifier: MQCA_SSL_KEY_REPOSITORY).

The length of the string is MQ_SSL_KEY_REPOSITORY_LENGTH.

Indicates the name of the Secure Sockets Layer key repository.

The format of the name depends on the environment:

- On z/OS, it is the name of a key ring.
- On i5/OS, it is of the form *pathname/keyfile*, where *keyfile* is specified without the suffix (.kdb), and identifies a GSKit key database file. The default value is /QIBM/UserData/ICSS/Cert/Server/Default.

If you specify *SYSTEM, WebSphere MQ utilizes the system certificate store as the key repository for the queue manager. As a result, the queue manager is registered as a server application in Digital Certificate Manager (DCM) and you can assign any server/client certificate in the system store to this application.

If you change the SSLKEYR parameter to a value other than *SYSTEM, WebSphere MQ deregisters the queue manager as an application with DCM.

- On UNIX it is of the form *pathname/keyfile* and on Windows *pathname\keyfile*, where *keyfile* is specified without the suffix (.kdb), and identifies a GSKit key database file. The default value for UNIX platforms is /var/mqm/qmgrs/QMGR/ss1/key, and on Windows it is C:\Program Files\IBM\WebSphere\QM\qmgrs\QMGR\ss1\key, where QMGR is replaced by the queue manager name (on UNIX and Windows).

On i5/OS, Windows, and UNIX systems, the syntax of this parameter is validated to ensure that it contains a valid, absolute, directory path.

If SSLKEYR is blank, or is set to a value that does not correspond to a key ring or key database file, channels using SSL fail to start.

Changes to SSLKeyRepository become effective:

- On i5/OS, Windows, and UNIX platforms, when a new channel process is started.

- For channels that run as threads of the channel initiator on i5/OS, Windows, and UNIX platforms, when the channel initiator is restarted.
- For channels that run as threads of the listener on i5/OS, Windows, and UNIX platforms, when the listener is restarted.
- On z/OS, when the channel initiator is restarted.

SSLKeyResetCount (MQCFIN)

SSL key reset count (parameter identifier: MQIA_SSL_RESET_COUNT).

Specifies when SSL channel MCAs that initiate communication reset the secret key used for encryption on the channel. The value of this parameter represents the total number of unencrypted bytes that are sent and received on the channel before the secret key is renegotiated. This number of bytes includes control information sent by the MCA.

The secret key is renegotiated when (whichever occurs first):

- The total number of unencrypted bytes sent and received by the initiating channel MCA exceeds the specified value, or,
- If channel heartbeats are enabled, before data is sent or received following a channel heartbeat.

Specify a value in the range zero through 999 999 999. A value of zero, the queue manager's initial default value, signifies that secret keys are never renegotiated.

SSLTasks (MQCFIN)

Number of server subtasks to use for processing SSL calls (parameter identifier: MQIA_SSL_TASKS). This parameter applies to z/OS only.

The number of server subtasks to use for processing SSL calls. To use SSL channels, you must have at least two of these tasks running.

Specify a value in the range zero through 9 999. However, to avoid problems with storage allocation, do not set this parameter to a value greater than 50.

StartStopEvent (MQCFIN)

Controls whether start and stop events are generated (parameter identifier: MQIA_START_STOP_EVENT).

The value can be:

MQEVR_DISABLED

Event reporting disabled.

MQEVR_ENABLED

Event reporting enabled.

StatisticsInterval (MQCFIN)

The time interval, in seconds, at which statistics monitoring data is written to the monitoring queue (parameter identifier: MQIA_STATISTICS_INTERVAL).

Specify a value in the range 1 through 604 000.

This parameter is valid only on AIX, HP-UX, Linux, i5/OS, Solaris, and Windows.

TCPChannels (MQCFIN)

The maximum number of channels that can be current, or clients that can be connected, that use the TCP/IP transmission protocol (parameter identifier: MQIA_TCP_CHANNELS).

Specify a value in the range zero to 9 999. The queue manager's initial default value is 200.

This parameter applies to z/OS only.

TCPKeepAlive (**MQCFIN**)

Whether the TCP KEEPALIVE facility is to be used to check whether the other end of a connection is still available (parameter identifier: MQIA_TCP_KEEP_ALIVE).

The value can be:

MQTCPKEEP_YES

The TCP KEEPALIVE facility is to be used as specified in the TCP profile configuration data set. The interval is specified in the *KeepAliveInterval* channel attribute.

MQTCPKEEP_NO

The TCP KEEPALIVE facility is not to be used. This is the queue manager's initial default value.

This parameter applies to z/OS only.

TCPName (**MQCFST**)

The name of the TCP/IP system that you are using (parameter identifier: MQIA_TCP_NAME).

The maximum length of the string is MQ_TCP_NAME_LENGTH.

This parameter applies to z/OS only.

TCPStackType (**MQCFIN**)

Whether the channel initiator may use only the TCP/IP address space specified in *TCPName*, or may optionally bind to any selected TCP/IP address (parameter identifier: MQIA_TCP_STACK_TYPE).

The value can be:

MQTCPSTACK_SINGLE

The channel initiator may only use the TCP/IP address space specified in *TCPName*. This is the queue manager's initial default value.

MQTCPSTACK_MULTIPLE

The channel initiator may use any TCP/IP address space available to it. It defaults to the one specified in *TCPName* if no other is specified for a channel or listener.

This parameter applies to z/OS only.

TraceRouteRecording (**MQCFIN**)

Whether trace-route information can be recorded and a reply message generated (parameter identifier: MQIA_TRACE_ROUTE_RECORDING).

The value can be:

MQRECORDING_DISABLED

Trace-route information cannot be recorded.

MQRECORDING_MSG

Trace-route information can be recorded and replies sent to the destination specified by the originator of the message causing the trace-route record.

MQRECORDING_Q

Trace-route information can be recorded and replies sent to SYSTEM.ADMIN TRACE.ROUTE.QUEUE.

If participation in route tracing is enabled using this queue manager attribute (by the attribute being not set to MQRECORDING_DISABLED) then the value of the attribute is only important should a reply be generated. The reply should go either to SYSTEM.ADMIN.TRACE.ROUTE.QUEUE, or to the destination specified by the message itself. Provided the attribute is not disabled then messages not yet at the final destination may have information added to them. For more information about trace-route records, see *WebSphere MQ Monitoring*.

TriggerInterval (MQCFIN)

Trigger interval (parameter identifier: MQIA_TRIGGER_INTERVAL).

Specifies the trigger time interval, expressed in milliseconds, for use only with queues where *TriggerType* has a value of MQTT_FIRST.

In this case trigger messages are normally generated only when a suitable message arrives on the queue, and the queue was previously empty. Under certain circumstances, however, an additional trigger message can be generated with MQTT_FIRST triggering, even if the queue was not empty. These additional trigger messages are not generated more often than every *TriggerInterval* milliseconds.

Specify a value in the range 0 through 999 999 999.

Error codes

This command might return the following in the response format header, in addition to the values shown on page “Error codes applicable to all commands” on page 27.

Reason (MQLONG)

The value can be:

MQRCCF_CHAD_ERROR

Channel automatic definition error.

MQRCCF_CHAD_EVENT_ERROR

Channel automatic definition event error.

MQRCCF_CHAD_EVENT_WRONG_TYPE

Channel automatic definition event parameter not allowed for this channel type.

MQRCCF_CHAD_EXIT_ERROR

Channel automatic definition exit name error.

MQRCCF_CHAD_EXIT_WRONG_TYPE

Channel automatic definition exit parameter not allowed for this channel type.

MQRCCF_CHAD_WRONG_TYPE

Channel automatic definition parameter not allowed for this channel type.

MQRCCF_FORCE_VALUE_ERROR

Force value not valid.

MQRCCF_PATH_NOT_VALID
Path not valid.
MQRCCF_PWD_LENGTH_ERROR
Password length error.
MQRCCF_Q_MGR_CCSID_ERROR
Coded character set value not valid.
MQRCCF_REPOS_NAME_CONFLICT
Repository names not valid.
MQRCCF_UNKNOWN_Q_MGR
Queue manager not known.

Change Security

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The Change Security (MQCMD_CHANGE_SECURITY) command defines system-wide security options.

Required parameters

None

Optional parameters:

CommandScope, SecurityInterval, SecurityTimeout,

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

SecurityInterval (MQCFIN)

Timeout check interval (parameter identifier: MQIACF_SECURITY_INTERVAL).

Specifies the interval between checks for user IDs and associated resources to determine whether the *SecurityTimeout* has occurred. The value specifies a number of minutes in the range zero through 10080 (one week). If *SecurityInterval* is specified as zero, no user timeouts occur. If

SecurityInterval is specified as nonzero, the user ID times out at a time between *SecurityTimeout* and *SecurityTimeout* plus *SecurityInterval*.

SecurityTimeout (MQCFIN)

Security information timeout (parameter identifier: MQIACF_SECURITY_TIMEOUT).

Specifies how long security information about an unused user ID and associated resources is retained by WebSphere MQ. The value specifies a number of minutes in the range zero through 10080 (one week). If *SecurityTimeout* is specified as zero, and *SecurityInterval* is nonzero, all such information is discarded by the queue manager every *SecurityInterval* number of minutes.

Change, Copy, and Create Service

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
		X	X	X	

The Change Service (MQCMD_CHANGE_SERVICE) command changes the specified attributes of an existing WebSphere MQ service definition. For any optional parameters that are omitted, the value does not change.

The Copy Service (MQCMD_COPY_SERVICE) command creates a new WebSphere MQ service definition, using, for attributes not specified in the command, the attribute values of an existing service definition.

The Create Service (MQCMD_CREATE_SERVICE) command creates a new WebSphere MQ service definition. Any attributes that are not defined explicitly are set to the default values on the destination queue manager.

Required parameter (Change and Create Service):

ServiceName

Required parameters (Copy Service):

FromServiceName, ToServiceName

Optional parameters:

Replace, ServiceDesc, ServiceType, StartArguments, StartCommand, StartMode, StderrDestination, StdoutDestination, StopArguments, StopCommand

Required parameter (Change and Create Service)

ServiceName (MQCFST)

The name of the service definition to be changed or created (parameter identifier: MQCA_SERVICE_NAME).

The maximum length of the string is MQ_OBJECT_NAME_LENGTH.

Required parameters (Copy Service)

FromServiceName (MQCFST)

The name of the service definition to be copied from (parameter identifier: MQCACF_FROM_SERVICE_NAME).

This specifies the name of the existing service definition that contains values for the attributes not specified in this command.

The maximum length of the string is MQ_OBJECT_NAME_LENGTH.

ToServiceName (MQCFST)

To service name (parameter identifier: MQCACF_TO_SERVICE_NAME).

This specifies the name of the new service definition. If a service definition with this name already exists, *Replace* must be specified as MQRP_YES.

The maximum length of the string is MQ_OBJECT_NAME_LENGTH.

Optional parameters

Replace (MQCFIN)

Replace attributes (parameter identifier: MQIACF_REPLACE).

If a namelist definition with the same name as *ToServiceName* already exists, this specifies whether it is to be replaced. The value can be:

MQRP_YES

Replace existing definition.

MQRP_NO

Do not replace existing definition.

ServiceDesc (MQCFST)

Description of service definition (parameter identifier: MQCA_SERVICE_DESC).

This is a plain-text comment that provides descriptive information about the service definition. It should contain only displayable characters.

If characters are used that are not in the coded character set identifier (CCSID) for the queue manager on which the command is executing, they might be translated incorrectly.

The maximum length of the string is MQ_SERVICE_DESC_LENGTH.

ServiceType (MQCFIN)

The mode in which the service is to run (parameter identifier: MQIA_SERVICE_TYPE).

Specify either:

MQSVC_TYPE_SERVER

Only one instance of the service can be executed at a time, with the status of the service made available by the Inquire Service Status command.

MQSVC_TYPE_COMMAND

Multiple instances of the service can be started.

StartArguments (MQCFST)

Arguments to be passed to the program on startup (parameter identifier: MQCA_SERVICE_START_ARGS).

Specify each argument within the string as you would on a command line, with a space to separate each argument to the program.

The maximum length of the string is MQ_SERVICE_ARGS_LENGTH.

StartCommand (MQCFST)

Service program name (parameter identifier: MQCA_SERVICE_START_COMMAND).

Specifies the name of the program which is to run. You must specify a fully qualified path name to the executable program.

The maximum length of the string is MQ_SERVICE_COMMAND_LENGTH.

***StartMode* (MQCFIN)**

Service mode (parameter identifier: MQIA_SERVICE_CONTROL).

Specifies how the service is to be started and stopped. The value can be:

MQSVC_CONTROL_MANUAL

The service is not to be started automatically or stopped automatically.
It is to be controlled by user command. This is the default value.

MQSVC_CONTROL_Q_MGR

The service being defined is to be started and stopped at the same time as the queue manager is started and stopped.

MQSVC_CONTROL_Q_MGR_START

The service is to be started at the same time as the queue manager is started, but is not requested to stop when the queue manager is stopped.

***StderrDestination* (MQCFST)**

Specifies the path to a file to which the standard error (stderr) of the service program should be redirected (parameter identifier: MQCA_STDERR_DESTINATION).

If the file does not exist when the service program is started, the file is created.

The maximum length of the string is MQ_SERVICE_PATH_LENGTH.

***StdoutDestination* (MQCFST)**

Specifies the path to a file to which the standard output (stdout) of the service program should be redirected (parameter identifier: MQCA_STDOUT_DESTINATION).

If the file does not exist when the service program is started, the file is created.

The maximum length of the string is MQ_SERVICE_PATH_LENGTH.

***StopArguments* (MQCFST)**

Specifies the arguments to be passed to the stop program when instructed to stop the service (parameter identifier: MQCA_SERVICE_STOP_ARGS).

Specify each argument within the string as you would on a command line, with a space to separate each argument to the program.

The maximum length of the string is MQ_SERVICE_ARGS_LENGTH.

***StopCommand* (MQCFST)**

Service program stop command (parameter identifier: MQCA_SERVICE_STOP_COMMAND).

This is the name of the program that is to run when the service is requested to stop. You must specify a fully qualified path name to the executable program.

The maximum length of the string is MQ_SERVICE_COMMAND_LENGTH.

Change, Copy, and Create Storage Class

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The Change Storage Class (MQCMD_CHANGE_STG_CLASS) command changes the characteristics of a storage class. For any optional parameters that are omitted, the value does not change.

The Copy Storage Class (MQCMD_COPY_STG_CLASS) command creates a new storage class to page set mapping using, for attributes not specified in the command, the attribute values of an existing storage class.

The Create Storage Class (MQCMD_CREATE_STG_CLASS) command creates a storage class to page set mapping. Any attributes that are not defined explicitly are set to the default values on the destination queue manager.

Required parameters (Change and Create Storage Class):

StorageClassName

Required parameters (Copy CF Storage Class):

FromStorageClassName, ToStorageClassName

Optional parameters:

*CommandScope, PageSetId, PassTicketApplication, QSGDisposition, Replace,
StorageClassDesc, XCFGroupName, XCFMemberName*

Required parameters (Change and Create Storage Class)

StorageClassName (MQCFST)

The name of the storage class to be changed or created (parameter identifier: MQCA_STORAGE_CLASS).

The maximum length of the string is MQ_STORAGE_CLASS_LENGTH.

Required parameters (Copy Storage Class)

FromStorageClassName (MQCFST)

The name of the storage class to be copied from (parameter identifier: MQCACF_FROM_STORAGE_CLASS).

On z/OS, the queue manager searches for an object with the name you specify and a disposition of MQQSGD_Q_MGR or MQQSGD_COPY to copy from.

This parameter is ignored if a value of MQQSGD_COPY is specified for *QSGDisposition*. In this case, an object with the name specified by *ToStorageClassName* and the disposition MQQSGD_GROUP is searched for to copy from.

The maximum length of the string is MQ_STORAGE_CLASS_LENGTH.

ToStorageClassName (MQCFST)

The name of the storage class to copy to (parameter identifier: MQCACF_TO_STORAGE_CLASS).

The maximum length of the string is MQ_STORAGE_CLASS_LENGTH.

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

PageSetId (MQCFIN)

Page set identifier that the storage class is to be associated with (parameter identifier: MQIA_PAGESET_ID).

Specify a string of two numeric characters in the range 00 through 99.

If you do not specify this, the default is taken from the default storage class SYSTEMST.

No check is made that the page set has been defined; an error is raised only if you try to put a message to a queue that specifies this storage class (MQRC_PAGESET_ERROR).

PassTicketApplication (MQCFST)

Pass ticket application (parameter identifier: MQCA_PASS_TICKET_APPL).

The application name that is passed to RACF when authenticating the passticket specified in the MQIIH header.

The maximum length is MQ_PASS_TICKET_APPL_LENGTH.

QSGDisposition (MQCFIN)

Disposition of the object within the group (parameter identifier: MQIA_QSG_DISP).

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves). The value can be:

QSGDisposition	Change	Copy, Create
MQQSGD_COPY	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameter MQQSGD_COPY. Any object residing in the shared repository, or any object defined using a command that had the parameters MQQSGD_Q_MGR, is not affected by this command.	The object is defined on the page set of the queue manager that executes the command using the MQQSGD_GROUP object of the same name as the <i>ToStorageClassName</i> object (for Copy) or the <i>StorageClassName</i> object (for Create).

QSGDisposition	Change	Copy, Create
MQQSGD_GROUP	<p>The object definition resides in the shared repository. The object was defined using a command that had the parameter MQQSGD_GROUP. Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command.</p> <p>If the command is successful, the following MQSC command is generated and sent to all active queue managers in the queue-sharing group to attempt to refresh local copies on page set zero:</p> <pre>DEFINE STGCLASS(storage-class) REPLACE QSGDISP(COPY)</pre> <p>The Change for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.</p>	<p>The object definition resides in the shared repository. This is allowed only if the queue manager is in a queue-sharing group.</p> <p>If the definition is successful, the following MQSC command is generated and sent to all active queue managers in the queue-sharing group to attempt to make or refresh local copies on page set zero:</p> <pre>DEFINE STGCLASS(storage-class) REPLACE QSGDISP(COPY)</pre> <p>The Copy or Create for the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.</p>
MQQSGD_PRIVATE	The object resides on the page set of the queue manager that executes the command, and was defined with MQQSGD_Q_MGR or MQQSGD_COPY. Any object residing in the shared repository is unaffected.	Not permitted.
MQQSGD_Q_MGR	The object definition resides on the page set of the queue manager that executes the command. The object was defined using a command that had the parameter MQQSGD_Q_MGR. Any object residing in the shared repository, or any local copy of such an object, is not affected by this command. This is the default value.	The object is defined on the page set of the queue manager that executes the command. This is the default value.

Replace (MQCFIN)

Replace attributes (parameter identifier: MQIACF_REPLACE).

If a storage class definition with the same name as *ToStorageClassName* already exists, this specifies whether it is to be replaced. The value can be:

MQRP_YES

Replace existing definition.

MQRP_NO

Do not replace existing definition.

StorageClassDesc (MQCFST)

The description of the storage class (parameter identifier: MQCA_STORAGE_CLASS_DESC).

The maximum length is MQ_STORAGE_CLASS_DESC_LENGTH.

XCFGroupName (MQCFST)

XCF group name (parameter identifier: MQCA_XCF_GROUP_NAME).

If you are using the IMS bridge, this is the name of the XCF group to which the IMS system belongs.

The maximum length is MQ_XCF_GROUP_NAME_LENGTH.

XCFMemberName (MQCFST)

XCF member name (parameter identifier: MQCA_XCF_MEMBER_NAME).

If you are using the IMS bridge, this is the XCF member name of the IMS system within the XCF group specified in *XCFGROUPNAME*.

The maximum length is MQ_XCF_MEMBER_NAME_LENGTH.

Clear Queue

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Clear Queue (MQCMD_CLEAR_Q) command deletes all the messages from a local queue.

The command fails if the queue contains uncommitted messages.

Required parameters:

QName

Optional parameters:

CommandScope, QSGDisposition

Required parameters

QName (MQCFST)

Queue name (parameter identifier: MQCA_Q_NAME).

The name of the local queue to be cleared. The maximum length of the string is MQ_Q_NAME_LENGTH.

Note: The target queue must be type local.

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

QSGDisposition (MQCFIN)

Disposition of the object within the group (parameter identifier: MQIA_QSG_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves). The value can be:

MQQSGD_PRIVATE

Clear the private queue named in *QName*. The queue is private if it was created using a command with the attributes MQQSGD_PRIVATE or MQQSGD_Q_MGR. This is the default value.

MQQSGD_SHARED

Clear the shared queue named in *QName*. The queue is shared if it was created using a command with the attribute MQQSGD_SHARED. This applies only to local queues.

Error codes

This command might return the following in the response format header, in addition to the values shown on page "Error codes applicable to all commands" on page 27.

Reason (MQLONG)

The value can be:

MQRC_Q_NOT_EMPTY

(2055, X'807') Queue contains one or more messages or uncommitted put or get requests.

This reason occurs only if there are uncommitted updates.

MQRCCF_Q_WRONG_TYPE

Action not valid for the queue of specified type.

Delete Authentication Information Object

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Delete authentication information (MQCMD_DELETE_AUTH_INFO) command deletes the specified authentication information object.

Required parameters :

AuthInfoName

Optional parameters :

CommandScope, *QSGDisposition*

Required parameters

AuthInfoName (MQCFST)

Authentication information object name (parameter identifier: MQCA_AUTH_INFO_NAME).

The maximum length of the string is MQ_AUTH_INFO_NAME_LENGTH.

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

QSGDisposition (MQCFIN)

Disposition of the object within the group (parameter identifier: MQIA_QSG_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves). The value can be:

MQQSGD_COPY

The object definition resides on the page set of the queue manager which executes this command. The object was defined by a command using the parameter MQQSGD_COPY. Any object in the shared repository, or any object defined by a command using the parameter MQQSGD_Q_MGR, is not affected by this command.

MQQSGD_GROUP

The object definition resides in the shared repository. The object was defined by a command using the parameter MQQSGD_GROUP. Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command.

If the command is successful, the following MQSC command is generated and sent to all active queue managers in the queue-sharing group to delete local copies on page set zero:

```
DELETE AUTHINFO(name) QSGDISP(COPY)
```

The deletion of the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.

MQQSGD_Q_MGR

The object definition resides on the page set of the queue manager that executes the command. The object was defined by a command using the parameter MQQSGD_Q_MGR. Any object residing in the shared repository, or any local copy of such an object, is not affected by this command.

This is the default value.

Delete Authority Record

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
		X	X	X	

The Delete Authority Record (MQCMD_DELETE_AUTH_REC) command deletes an authority record. The authorizations associated with the profile no longer apply to WebSphere MQ objects with names that match the profile name specified.

Required parameters:

ProfileName, ObjectType

Optional parameters:

GroupNames, PrincipalNames

Required parameters

ObjectType (MQCFIN)

The type of object for which to delete authorizations (parameter identifier: MQIACF_OBJECT_TYPE).

The value can be:

MQOT_AUTH_INFO

Authentication information.

MQOT_CHANNEL

Channel object.

MQOT_CLNTCONN_CHANNEL

Client-connection channel object.

MQOT_LISTENER

Listener object.

MQOT_NAMELIST

Namelist.

MQOT_PROCESS

Process.

MQOT_Q

Queue, or queues, that match the object name parameter.

MQOT_Q_MGR

Queue manager.

MQOT_SERVICE

Service object.

ProfileName (MQCFST)

Name of the profile to be deleted (parameter identifier: MQCACF_AUTH_PROFILE_NAME).

If you have defined a generic profile then you may specify it here, using wildcard characters to specify a named generic profile to be removed. If you specify an explicit profile name, the object must exist.

The maximum length of the string is MQ_AUTH_PROFILE_NAME_LENGTH.

Optional parameters

GroupNames (MQCFSL)

Group names (parameter identifier: MQCACF_GROUP_ENTITY_NAMES).

The names of groups having a profile deleted. At least one group name or principal name must be specified. An error occurs if neither are specified.

Each member in this list can be a maximum length of MQ_ENTITY_NAME_LENGTH.

PrincipalNames (MQCFSL)

Principal names (parameter identifier: MQCACF_PRINCIPAL_ENTITY_NAMES).

The names of principals having a profile deleted. At least one group name or principal name must be specified. An error occurs if neither are specified.

Each member in this list can be a maximum length of MQ_ENTITY_NAME_LENGTH.

Error codes

This command might return the following in the response format header, in addition to the values shown on page “Error codes applicable to all commands” on page 27.

Reason (MQLONG)

The value can be:

MQRC_OBJECT_TYPE_ERROR

Invalid object type.

MQRC_UNKNOWN_ENTITY

Userid not authorized, or unknown.

MQRCCF_ENTITY_NAME_MISSING

Entity name missing.

MQRCCF_OBJECT_TYPE_MISSING

Object type missing.

MQRCCF_PROFILE_NAME_ERROR

Invalid profile name.

Delete CF Structure

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The Delete CF Structure (MQCMD_DELETE_CF_STRUC) command deletes an existing CF application structure definition.

Note: This command is supported only on z/OS when the queue manager is a member of a queue-sharing group.

Required parameters:

CFStrucName

Optional parameters:
None

Required parameters

CFStrucName (MQCFST)

CF structure name (parameter identifier: MQCA_CF_STRUC_NAME).

The CF application structure definition to be deleted. The maximum length of the string is MQ_CF_STRUC_NAME_LENGTH.

Delete Channel

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Delete Channel (MQCMD_DELETE_CHANNEL) command deletes the specified channel definition.

Required parameters:
ChannelName

Optional parameters:
ChannelTable, CommandScope, QSGDisposition

Required parameters

ChannelName (MQCFST)

Channel name (parameter identifier: MQCACH_CHANNEL_NAME).

The name of the channel definition to be deleted. The maximum length of the string is MQ_CHANNEL_NAME_LENGTH.

Optional parameters

ChannelTable (MQCFIN)

Channel table (parameter identifier: MQIACH_CHANNEL_TABLE).

Specifies the ownership of the channel definition table that contains the specified channel definition.

The value can be:

MQCHTAB_Q_MGR

Queue-manager table.

This is the default. This table contains channel definitions for channels of all types except MQCHT_CLNTCONN.

MQCHTAB_CLNTCONN

Client-connection table.

This table only contains channel definitions for channels of type MQCHT_CLNTCONN.

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

QSGDisposition (MQCFIN)

Disposition of the object within the group (parameter identifier: MQIA_QSG_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves). The value can be:

MQQSGD_COPY

The object definition resides on the page set of the queue manager that executes the command. The object was defined by a command using the parameter MQQSGD_COPY. Any object residing in the shared repository, or any object defined by a command using the parameter MQQSGD_Q_MGR, is not affected by this command.

MQQSGD_GROUP

The object definition resides in the shared repository. The object was defined by a command using the parameters MQQSGD_GROUP. Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command.

If the command is successful, the following MQSC command is generated and sent to all active queue managers in the queue-sharing group to delete local copies on page set zero:

```
DELETE CHANNEL(name) QSGDISP(COPY)
```

The deletion of the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.

MQQSGD_Q_MGR

The object definition resides on the page set of the queue manager that executes the command. The object was defined by a command using the parameter MQQSGD_Q_MGR. Any object residing in the shared repository, or any local copy of such an object, is not affected by this command.

This is the default value.

Error codes

This command might return the following in the response format header, in addition to the values shown on page “Error codes applicable to all commands” on page 27.

Reason (MQLONG)

The value can be:

MQRCCF_CHANNEL_NOT_FOUND

Channel not found.

MQRCCF_CHANNEL_TABLE_ERROR

Channel table value not valid.

Delete Channel Listener

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
		X	X	X	

The Delete Channel Listener (MQCMD_DELETE_LISTENER) command deletes an existing channel listener definition.

Required parameters:

ListenerName

Optional parameters:

None

Required parameters

ListenerName (MQCFST)

Listener name (parameter identifier: MQCACH_LISTENER_NAME).

This is the name of the listener definition to be deleted. The maximum length of the string is MQ_LISTENER_NAME_LENGTH.

Delete Namelist

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Delete Namelist (MQCMD_DELETE_NAMELIST) command deletes an existing namelist definition.

Required parameters:

NamelistName

Optional parameters:

CommandScope, *QSGDisposition*

Required parameters

NamelistName (MQCFST)

Namelist name (parameter identifier: MQCA_NAMELIST_NAME).

This is the name of the namelist definition to be deleted. The maximum length of the string is MQ_NAMELIST_NAME_LENGTH.

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

QSGDisposition (MQCFIN)

Disposition of the object within the group (parameter identifier: MQIA_QSG_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves). The value can be:

MQQSGD_COPY

The object definition resides on the page set of the queue manager that executes the command. The object was defined by a command using the parameter MQQSGD_COPY. Any object residing in the shared repository, or any object defined using a command that had the parameters MQQSGD_Q_MGR, is not affected by this command.

MQQSGD_GROUP

The object definition resides in the shared repository. The object was defined by a command using the parameter MQQSGD_GROUP. Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command.

If the command is successful, the following MQSC command is generated and sent to all active queue managers in the queue-sharing group to delete local copies on page set zero:

`DELETE NAMELIST(name) QSGDISP(COPY)`

The deletion of the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.

MQQSGD_Q_MGR

The object definition resides on the page set of the queue manager that executes the command. The object was defined by a command using the parameter MQQSGD_Q_MGR. Any object residing in the shared repository, or any local copy of such an object, is not affected by this command.

This is the default value.

Delete Process

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Delete Process (MQCMD_DELETE_PROCESS) command deletes an existing process definition.

Required parameters:

ProcessName

Optional parameters:

CommandScope, QSGDisposition

Required parameters

***ProcessName* (MQCFST)**

Process name (parameter identifier: MQCA_PROCESS_NAME).

The process definition to be deleted. The maximum length of the string is MQ_PROCESS_NAME_LENGTH.

Optional parameters

***CommandScope* (MQCFST)**

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

***QSGDisposition* (MQCFIN)**

Disposition of the object within the group (parameter identifier: MQIA_QSG_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves). The value can be:

MQQSGD_COPY

The object definition resides on the page set of the queue manager that executes the command. The object was defined by a command using the parameter MQQSGD_COPY. Any object residing in the shared

repository, or any object defined using a command that had the parameters MQQSGD_Q_MGR, is not affected by this command.

MQQSGD_GROUP

The object definition resides in the shared repository. The object was defined by a command using the parameter MQQSGD_GROUP. Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command.

If the command is successful, the following MQSC command is generated and sent to all active queue managers in the queue-sharing group to delete local copies on page set zero:

```
DELETE PROCESS(name) QSGDISP(COPY)
```

The deletion of the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.

MQQSGD_Q_MGR

The object definition resides on the page set of the queue manager that executes the command. The object was defined by a command using the parameter MQQSGD_Q_MGR. Any object residing in the shared repository, or any local copy of such an object, is not affected by this command.

This is the default value.

Delete Queue

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Delete Queue (MQCMD_DELETE_Q) command deletes a queue.

Required parameters:

QName

Optional parameters (any QType):

CommandScope, QSGDisposition, QType

Optional parameters (local QType only):

Purge

Required parameters

***QName* (MQCFST)**

Queue name (parameter identifier: MQCA_Q_NAME).

The name of the queue to be deleted.

If the *Scope* attribute of the queue is MQSCO_CELL, the entry for the queue is deleted from the cell directory.

The maximum length of the string is MQ_Q_NAME_LENGTH.

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

Purge (MQCFIN)

Purge queue (parameter identifier: MQIACF_PURGE).

If there are messages on the queue MQPO_YES must be specified, otherwise the command will fail. If this parameter is not present the queue is not purged.

Valid only for queue of type local.

The value can be:

MQPO_YES

Purge the queue.

MQPO_NO

Do not purge the queue.

QSGDisposition (MQCFIN)

Disposition of the object within the group (parameter identifier: MQIA_QSG_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves). The value can be:

MQQSGD_COPY

The object definition resides on the page set of the queue manager that executes the command. The object was defined by a command using the parameter MQQSGD_COPY. Any object residing in the shared repository, or any object defined using a command that had the parameters MQQSGD_Q_MGR, is not affected by this command.

MQQSGD_GROUP

The object definition resides in the shared repository. The object was defined by a command using the parameter MQQSGD_GROUP. Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command.

If the deletion is successful, the following MQSC command is generated and sent to all active queue managers in the queue-sharing group to make, or delete, local copies on page set zero:

```
DELETE queue(q-name) QSGDISP(COPY)
```

or, for a local queue only:

```
DELETE QLOCAL(q-name) NOPURGE QSGDISP(COPY)
```

The deletion of the group object takes effect even if the generated command with *QSGDISP(COPY)* fails.

Note: You always get the NOPURGE option even if you specify MQPO_YES for *Purge*. To delete messages on local copies of the queues, you must explicitly issue, for each copy, the Delete Queue command with a *QSGDisposition* value of MQQSGD_COPY and a *Purge* value of MQPO_YES.

MQQSGD_Q_MGR

The object definition resides on the page set of the queue manager that executes the command. The object was defined by a command using the parameter MQQSGD_Q_MGR. Any object residing in the shared repository, or any local copy of such an object, is not affected by this command.

This is the default value.

MQQSGD_SHARED

Valid only for queue of type local.

The object resides in the shared repository. The object was defined by a command using the parameter MQQSGD_SHARED. Any object residing on the page set of the queue manager that executes the command, or any object defined by a command using the parameter MQQSGD_GROUP, is not affected by this command.

QType (MQCFIN)

Queue type (parameter identifier: MQIA_Q_TYPE).

If this parameter is present, the queue must be of the specified type.

The value can be:

MQQT_ALIAS

Alias queue definition.

MQQT_LOCAL

Local queue.

MQQT_REMOTE

Local definition of a remote queue.

MQQT_MODEL

Model queue definition.

Error codes

This command might return the following in the response format header, in addition to the values shown on page “Error codes applicable to all commands” on page 27.

Reason (MQLONG)

The value can be:

MQRC_Q_NOT_EMPTY

(2055, X'807') Queue contains one or more messages or uncommitted put or get requests.

Delete Service

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
		X	X	X	

The Delete Service (MQCMD_DELETE_SERVICE) command deletes an existing service definition.

Required parameters:

ServiceName

Optional parameters:

None

Required parameters

ServiceName (MQCFST)

Service name (parameter identifier: MQCA_SERVICE_NAME).

This is the name of the service definition to be deleted.

The maximum length of the string is MQ_OBJECT_NAME_LENGTH.

Delete Storage Class

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The Delete Storage Class (MQCMD_DELETE_STG_CLASS) command deletes an existing storage class definition.

Required parameters:

StorageClassName

Optional parameters:

CommandScope, QSGDisposition

Required parameters

StorageClassName (MQCFST)

Storage class name (parameter identifier: MQCA_STORAGE_CLASS).

The storage class definition to be deleted. The maximum length of the string is MQ_STORAGE_CLASS_LENGTH.

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

QSGDisposition (MQCFIN)

Disposition of the object within the group (parameter identifier: MQIA_QSG_DISP).

Specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves). The value can be:

MQQSGD_COPY

The object definition resides on the page set of the queue manager that executes the command. The object was defined by a command using the parameter MQQSGD_COPY. Any object residing in the shared repository, or any object defined using a command that had the parameters MQQSGD_Q_MGR, is not affected by this command.

MQQSGD_GROUP

The object definition resides in the shared repository. The object was defined by a command using the parameter MQQSGD_GROUP. Any object residing on the page set of the queue manager that executes the command (except a local copy of the object) is not affected by this command.

If the command is successful, the following MQSC command is generated and sent to all active queue managers in the queue-sharing group to delete local copies on page set zero:

```
DELETE STGCLASS(name) QSGDISP(COPY)
```

The deletion of the group object takes effect regardless of whether the generated command with QSGDISP(COPY) fails.

MQQSGD_Q_MGR

The object definition resides on the page set of the queue manager that executes the command. The object was defined by a command using the parameter MQQSGD_Q_MGR. Any object residing in the shared repository, or any local copy of such an object, is not affected by this command.

This is the default value.

Escape

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	

The Escape (MQCMD_ESCAPE) command conveys any WebSphere MQ command (MQSC) to a remote queue manager. Use it when the queue manager (or application) sending the command does not support the functionality of the particular WebSphere MQ command, and so does not recognize it and cannot construct the required PCF command.

The Escape command can also be used to send a command for which no Programmable Command Format has been defined.

The only type of command that can be carried is one that is identified as an MQSC, that is recognized at the receiving queue manager.

Required parameters:

EscapeType, *EscapeText*

Optional parameters:

None

Required parameters

***EscapeType* (MQCFIN)**

Escape type (parameter identifier: MQIACF_ESCAPE_TYPE).

The only value supported is:

MQET_MQSC

WebSphere MQ command.

***EscapeText* (MQCFST)**

Escape text (parameter identifier: MQCACF_ESCAPE_TEXT).

A string to hold a command. The length of the string is limited only by the size of the message.

Error codes

This command might return the following in the response format header, in addition to the values shown on page “Error codes applicable to all commands” on page 27.

***Reason* (MQLONG)**

The value can be:

MQRCCF_ESCAPE_TYPE_ERROR

Escape type not valid.

Escape (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	

The response to the Escape (MQCMD_ESCAPE) command consists of the response header followed by two parameter structures, one containing the escape type, and

the other containing the text response. More than one such message might be issued, depending upon the command contained in the Escape request.

The *Command* field in the response header MQCFH contains the MQCMD_** command identifier of the text command contained in the *EscapeText* parameter in the original Escape command. For example, if *EscapeText* in the original Escape command specified PING QMGR, *Command* in the response has the value MQCMD_PING_Q_MGR.

If it is possible to determine the outcome of the command, the *CompCode* in the response header identifies whether the command was successful. The success or otherwise can therefore be determined without the recipient of the response having to parse the text of the response.

If it is not possible to determine the outcome of the command, *CompCode* in the response header has the value MQCC_UNKNOWN, and *Reason* is MQRC_NONE.

Always returned:

EscapeType, *EscapeText*

Returned if requested:

None

Parameters

EscapeType (MQCFIN)

Escape type (parameter identifier: MQIACF_ESCAPE_TYPE).

The only value supported is:

MQET_MQSC

WebSphere MQ command.

EscapeText (MQCFST)

Escape text (parameter identifier: MQCACF_ESCAPE_TEXT).

A string holding the response to the original command.

Inquire Archive

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The Inquire Archive (MQCMD_INQUIRE_ARCHIVE) command returns archive system parameters and information.

Required parameters:

None

Optional parameters:

CommandScope

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

Inquire Archive (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The response to the Inquire Archive (MQCMD_INQUIRE_ARCHIVE) command consists of the response header followed by the *ParameterType* structure and the combination of attribute parameter structures determined by the value of *ParameterType*.

Always returned:

ParameterType Specifies the type of archive information being returned.
The value can be:

MQSYSP_TYPE_INITIAL

The initial settings of the archive parameters.

MQSYSP_TYPE_SET

The settings of the archive parameters if they have been altered since their initial setting.

MQSYSP_TYPE_ARCHIVE_TAPE

Parameters relating to the tape unit (if in use). There is one such message per tape unit in use for archive logging.

Returned if *ParameterType* is MQSYSP_TYPE_INITIAL (one message is returned):

AllocPrimary, *AllocSecondary*, *AllocUnits*, *ArchivePrefix1*,
ArchivePrefix2, *ArchiveRetention*, *ArchiveUnit1*, *ArchiveUnit2*,
ArchiveWTOR, *BlockSize*, *Catalog*, *Compact*, *Protect*, *QuiesceInterval*,
RoutingCode, *TimeStampFormat*

Returned if *ParameterType* is MQSYSP_TYPE_SET and any value is set (one message is returned):

AllocPrimary, *AllocSecondary*, *AllocUnits*, *ArchivePrefix1*,
ArchivePrefix2, *ArchiveRetention*, *ArchiveUnit1*, *ArchiveUnit2*,
ArchiveWTOR, *BlockSize*, *Catalog*, *Compact*, *Protect*, *QuiesceInterval*,
RoutingCode, *TimeStampFormat*

Returned if *ParameterType* is MQSYSP_TYPE_ARCHIVE_TAPE (one message is returned for each tape unit in use for archive logging):

DataSetName, LogCorrelId, UnitAddress, UnitStatus, UnitVolser

Response data - archive parameter information

AllocPrimary (MQCFIN)

Primary space allocation for DASD data sets (parameter identifier: MQIACF_SYSP_ALLOC_PRIMARY).

Specifies the primary space allocation for DASD data sets in the units specified in the *AllocUnits* parameter.

AllocSecondary (MQCFIN)

Primary space allocation for DASD data sets (parameter identifier: MQIACF_SYSP_ALLOC_SECONDARY).

Specifies the secondary space allocation for DASD data sets in the units specified in the *AllocUnits* parameter.

AllocUnits (MQCFIN)

Allocation unit (parameter identifier: MQIACF_SYSP_ALLOC_UNIT).

Specifies the unit in which primary and secondary space allocations are made. The value can be:

MQSYSP_ALLOC_BLK
Blocks.

MQSYSP_ALLOC_TRK
Tracks.

MQSYSP_ALLOC_CYL
Cylinders.

ArchivePrefix1 (MQCFST)

Prefix for the first archive log data set name (parameter identifier: MQCACF_SYSP_ARCHIVE_PFX1).

The maximum length of the string is MQ_ARCHIVE_PFX_LENGTH.

ArchivePrefix2 (MQCFST)

Prefix for the second archive log data set name (parameter identifier: MQCACF_SYSP_ARCHIVE_PFX2).

The maximum length of the string is MQ_ARCHIVE_PFX_LENGTH.

ArchiveRetention (MQCFIN)

Archive retention period (parameter identifier: MQIACF_SYSP_ARCHIVE_RETAIN).

Specifies the retention period, in days, to be used when the archive log data set is created.

ArchiveUnit1 (MQCFST)

Specifies the device type or unit name of the device that is used to store the first copy of the archive log data set (parameter identifier: MQCACF_SYSP_ARCHIVE_UNIT1).

The maximum length of the string is MQ_ARCHIVE_UNIT_LENGTH.

ArchiveUnit2 (MQCFST)

Specifies the device type or unit name of the device that is used to store the second copy of the archive log data set (parameter identifier: MQCACF_SYSP_ARCHIVE_UNIT2).

The maximum length of the string is MQ_ARCHIVE_UNIT_LENGTH.

ArchiveWTOR (MQCFIN)

Specifies whether a message is to be sent to the operator and a reply is received before attempting to mount an archive log data set (parameter identifier: MQIACF_SYSP_ARCHIVE_WTOR).

The value can be:

MQSYSP_YES

A message is to be sent and a reply received before an attempt to mount an archive log data set.

MQSYSP_NO

A message is not to be sent and a reply received before an attempt to mount an archive log data set.

BlockSize (MQCFIN)

Block size of the archive log data set (parameter identifier: MQIACF_SYSP_BLOCK_SIZE).

Catalog (MQCFIN)

Specifies whether archive log data sets are cataloged in the primary integrated catalog facility (parameter identifier: MQIACF_SYSP_CATALOG).

The value can be:

MQSYSP_YES

Archive log data sets are cataloged.

MQSYSP_NO

Archive log data sets are not cataloged.

Compact (MQCFIN)

Specifies whether data written to archive logs is to be compacted (parameter identifier: MQIACF_SYSP_COMPACT).

The value can be:

MQSYSP_YES

Data is to be compacted.

MQSYSP_NO

Data is not to be compacted.

Protect (MQCFIN)

Protection by external security manager (ESM) (parameter identifier: MQIACF_SYSP_PROTECT).

Specifies whether archive log data sets are protected by ESM profiles when the data sets are created.

The value can be:

MQSYSP_YES

Data set profiles are created when logs are off-loaded.

MQSYSP_NO

Profiles are not created.

QuiesceInterval (MQCFIN)

Maximum time allowed for the quiesce (parameter identifier: MQIACF_SYSP QUIESCE_INTERVAL).

Specifies the maximum time, in seconds, allowed for the quiesce.

RoutingCode (**MQCFIL**)

z/OS routing code list (parameter identifier: MQIACF_SYSP_ROUTING_CODE).

Specifies the list of z/OS routing codes for messages about the archive log data sets to the operator. There can be between 1 and 14 entries in the list.

TimeStampFormat (**MQCFIN**)

Time stamp included (parameter identifier: MQIACF_SYSP_TIMESTAMP).

Specifies whether the archive log data set name has a time stamp in it.

The value can be:

MQSYSP_YES

Names include a time stamp.

MQSYSP_NO

Names do not include a time stamp.

MQSYSP_EXTENDED

Names include a time stamp.

Response data - tape unit status information

DataSetName (**MQCFST**)

Data set name (parameter identifier: MQCACF_DATA_SET_NAME).

Specifies the data set name on the tape volume that is being processed, or was last processed.

The maximum length of the string is MQ_DATA_SET_NAME_LENGTH.

LogCorrelId (**MQCFST**)

Correlation identifier (parameter identifier: MQCACF_SYSP_LOG_CORREL_ID).

Specifies the correlation ID associated with the user of the tape being processed. This is blank if there is no current user.

The maximum length of the string is MQ_LOG_CORREL_ID_LENGTH.

UnitAddress (**MQCFIN**)

Tape unit address: MQIACF_SYSP_UNIT_ADDRESS).

Specifies the physical address of the tape unit allocated to read the archive log.

UnitStatus (**MQCFIN**)

Status if the tape unit: MQIACF_SYSP_UNIT_STATUS).

The value can be:

MQSYSP_STATUS_BUSY

The tape unit is busy, actively processing an archive log data set.

MQSYSP_STATUS_PREMOUNT

The tape unit is active and allocated for premounting.

MQSYSP_STATUS_AVAILABLE

The tape unit is available, inactive and waiting for work.

MQSYSP_STATUS_UNKNOWN

The tape unit status us unknown.

UnitVolser (**MQCFST**)

The volume serial number of the tape that is mounted (parameter identifier: MQCACF_SYSP_UNIT_VOLSER).

The maximum length of the string is MQ_VOLSER_LENGTH.

Inquire Authentication Information Object

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Inquire authentication information object (MQCMD_INQUIRE_AUTH_INFO) command inquires about the attributes of authentication information objects.

Required parameters :

AuthInfoName

Optional parameters:

*AuthInfoAttrs, CommandScope, IntegerFilterCommand, QSGDisposition,
StringFilterCommand*

Required parameters

***AuthInfoName* (MQCFST)**

Authentication information object name (parameter identifier: MQCA_AUTH_INFO_NAME).

Specifies the name of the authentication information object about which information is to be returned.

Generic authentication information object names are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all authentication information objects having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length of the string is MQ_AUTH_INFO_NAME_LENGTH.

Optional parameters

***AuthInfoAttrs* (MQCFIL)**

Authentication information object attributes (parameter identifier: MQIACF_AUTH_INFO_ATTRS).

The attribute list can specify the following on its own (this is the default value if the parameter is not specified) :

MQIACF_ALL

All attributes.

or a combination of the following :

MQCA_ALTERATION_DATE

Date on which the definition was last altered.

MQCA_ALTERATION_TIME

Time at which the definition was last altered.

MQCA_AUTH_INFO_NAME

Name of the authentication information object.

MQIA_AUTH_INFO_TYPE

Type of authentication information object.

MQCA_AUTH_INFO_CONN_NAME

Connection name of the authentication information object.

MQCA_LDAP_USER_NAME

LDAP user name in the authentication information object.

MQCA_LDAP_PASSWORD

LDAP password in the authentication information object.

MQCA_AUTH_INFO_DESC

Description of the authentication information object.

***CommandScope* (MQCFST)**

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

You cannot use *CommandScope* as a parameter to filter on.

***IntegerFilterCommand* (MQCFIF)**

Integer filter command descriptor. The parameter identifier must be any integer type parameter of those allowed in *AuthInfoAttrs*, except MQIACF_ALL. Use this to restrict the output from the command by specifying a filter condition. See “MQCFIF - PCF integer filter parameter” on page 427 for information about using this filter condition.

If you specify an integer filter, you cannot also specify a string filter using the *StringFilterCommand* parameter.

***QSGDisposition* (MQCFIN)**

Disposition of the object within the group (parameter identifier: MQIA_QSG_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object for which information is to be returned (that is, where it is defined and how it behaves). The value can be:

MQQSGD_LIVE

The object is defined as MQQSGD_Q_MGR or MQQSGD_COPY. This is the default value if the parameter is not specified.

MQQSGD_ALL

The object is defined as MQQSGD_Q_MGR or MQQSGD_COPY.

If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this option also displays information for objects defined with MQQSGD_GROUP.

If MQQSGD_LIVE is specified or defaulted, or if MQQSGD_ALL is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

MQQSGD_COPY

The object is defined as MQQSGD_COPY.

MQQSGD_GROUP

The object is defined as MQQSGD_GROUP. This is permitted only in a shared queue environment.

MQQSGD_Q_MGR

The object is defined as MQQSGD_Q_MGR.

MQQSGD_PRIVATE

The object is defined as either MQQSGD_Q_MGR or MQQSGD_COPY. Note that MQQSGD_PRIVATE returns the same information as MQQSGD_LIVE.

You cannot use *QSGDisposition* as a parameter to filter on.

StringFilterCommand (MQCFSF)

String filter command descriptor. The parameter identifier must be any string type parameter of those allowed in *AuthInfoAttrs*, except MQCA_AUTH_INFO_NAME. Use this to restrict the output from the command by specifying a filter condition. See "MQCFSF - PCF string filter parameter" on page 434 for information about using this filter condition.

If you specify a string filter, you cannot also specify an integer filter using the *IntegerFilterCommand* parameter.

Inquire Authentication Information Object (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The response of the Inquire authentication information (MQCMD_INQUIRE_AUTH_INFO) command consists of the response header followed by the *AuthInfoName* structure (and on z/OS only, the *QSGDisposition* structure), and the requested combination of attribute parameter structures (where applicable).

Always returned:

AuthInfoName, *QSGDisposition*

Returned if requested:

AlterationDate, *AlterationTime*, *AuthInfoConnName*, *AuthInfoDesc*,
AuthInfoType, *LDAPPassword*, *LDAPUserName*

Response data

AlterationDate (MQCFST)

Alteration date of the authentication information object, in the form yyyy-mm-dd (parameter identifier: MQCA_ALTERATION_DATE).

AlterationTime (**MQCFST**)

Alteration time of the authentication information object, in the form hh.mm.ss (parameter identifier: MQCA_ALTERATION_TIME).

AuthInfoConnName (**MQCFST**)

The connection name of the authentication information object (parameter identifier: MQCA_AUTH_INFO_CONN_NAME).

The maximum length of the string is MQ_AUTH_INFO_CONN_NAME_LENGTH. On z/OS, it is MQ_LOCAL_ADDRESS_LENGTH.

AuthInfoDesc (**MQCFST**)

The description of the authentication information object (parameter identifier: MQCA_AUTH_INFO_DESC).

The maximum length is MQ_AUTH_INFO_DESC_LENGTH.

AuthInfoName (**MQCFST**)

authentication information object name (parameter identifier: MQCA_AUTH_INFO_NAME).

The maximum length of the string is MQ_AUTH_INFO_NAME_LENGTH.

AuthInfoType (**MQCFIN**)

The type of authentication information object (parameter identifier: MQIA_AUTH_INFO_TYPE).

The value can be:

MQAIT_CRL_LDAP

This defines this authentication information object as specifying Certificate Revocation Lists that are held on the LDAP. See the *WebSphere MQ Security* book for more information.

LDAPPassword (**MQCFST**)

The LDAP password (parameter identifier: MQCA_LDAP_PASSWORD).

The maximum length is MQ_LDAP_PASSWORD_LENGTH.

LDAPUserName (**MQCFST**)

The LDAP user name (parameter identifier: MQCA_LDAP_USER_NAME).

The Distinguished Name of the user who is binding to the directory.

The maximum length is MQ_DISTINGUISHED_NAME_LENGTH. On z/OS, it is MQ_SHORT_DNAME_LENGTH.

QSGDisposition (**MQCFIN**)

QSG disposition (parameter identifier: MQIA_QSG_DISP).

Specifies the disposition of the object (that is, where it is defined and how it behaves). This parameter is valid on z/OS only. The value can be:

MQQSGD_COPY

The object is defined as MQQSGD_COPY.

MQQSGD_GROUP

The object is defined as MQQSGD_GROUP.

MQQSGD_Q_MGR

The object is defined as MQQSGD_Q_MGR.

Inquire Authentication Information Object Names

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Inquire authentication information names (MQCMD_INQUIRE_AUTH_INFO_NAMES) command asks for a list of authentication information names that match the generic authentication information name specified.

Required parameters:

AuthInfoName

Optional parameters:

CommandScope, QSGDisposition

Required parameters

AuthInfoName (MQCFST)

Authentication information object name (parameter identifier: MQCA_AUTH_INFO_NAME).

Specifies the name of the authentication information object about which information is to be returned.

Generic authentication information object names are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all authentication information objects having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length of the string is MQ_AUTH_INFO_NAME_LENGTH.

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

QSGDisposition (MQCFIN)

Disposition of the object within the group (parameter identifier: MQIA_QSG_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object for which information is to be returned (that is, where it is defined and how it behaves). The value can be:

MQQSGD_LIVE

The object is defined as MQQSGD_Q_MGR or MQQSGD_COPY. This is the default value if the parameter is not specified.

MQQSGD_ALL

The object is defined as MQQSGD_Q_MGR or MQQSGD_COPY.

If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this option also displays information for objects defined with MQQSGD_GROUP.

If MQQSGD_LIVE is specified or defaulted, or if MQQSGD_ALL is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

MQQSGD_COPY

The object is defined as MQQSGD_COPY.

MQQSGD_GROUP

The object is defined as MQQSGD_GROUP. This is permitted only in a shared queue environment.

MQQSGD_Q_MGR

The object is defined as MQQSGD_Q_MGR.

MQQSGD_PRIVATE

The object is defined as either MQQSGD_Q_MGR or MQQSGD_COPY. Note that MQQSGD_PRIVATE returns the same information as MQQSGD_LIVE.

Inquire Authentication Information Object Names (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The response to the inquire authentication information names (MQCMD_INQUIRE_AUTH_INFO_NAMES) command consists of the response header followed by a parameter structure giving zero or more names that match the specified authentication information name.

In addition to this, on z/OS only, a parameter structure, *QSGDispositions*, (with the same number of entries as the *AuthInfoNames* structure) is returned. Each entry in this structure indicates the disposition of the object with the corresponding entry in the *AuthInfoNames* structure.

Always returned:

AuthInfoNames, *QSGDispositions*

Returned if requested:

None

Response data

AuthInfoNames (MQCFSL)

List of authentication information object names (parameter identifier: MQCACF_AUTH_INFO_NAMES).

QSGDispositions (MQCFIL)

List of QSG dispositions (parameter identifier: MQIACF_QSG_DISPS).

Specifies the disposition of the object (that is, where it is defined and how it behaves). This parameter is valid on z/OS only. The value can be:

MQQSGD_COPY

The object is defined as MQQSGD_COPY.

MQQSGD_GROUP

The object is defined as MQQSGD_GROUP.

MQQSGD_Q_MGR

The object is defined as MQQSGD_Q_MGR.

Inquire Authority Records

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
		X	X	X	

The Inquire Authority Records (MQCMD_INQUIRE_AUTH_RECS) command retrieves authority records associated with a profile name.

Required parameters:

Options, ProfileName

Optional parameters:

EntityName, EntityType, ObjectType, ProfileAttrs, ServiceComponent

Required parameters

Options (MQCFIN)

Options to control the set of authority records that is returned (parameter identifier: MQIACF_AUTH_OPTIONS).

This parameter is required and you should include one of the following two values:

MQAUTHOPT_NAME_ALL_MATCHING

Return all profiles the names of which match the specified *ProfileName*. This means that a *ProfileName* of ABCD results in the profiles ABCD, ABC*, and AB* being returned (if ABC* and AB* have been defined as profiles).

MQAUTHOPT_NAME_EXPLICIT

Return only those profiles the names of which exactly match the *ProfileName*. No matching generic profiles are returned unless the *ProfileName* is, itself, a generic profile. You cannot specify this and MQAUTHOPT_ENTITY_SET.

and one of the following two values:

MQAUTHOPT_ENTITY_EXPLICIT

Return all profiles the entity fields of which match the specified *EntityName*. No profiles are returned for any group in which *EntityName* is a member; only the profile defined for the specified *EntityName*.

MQAUTHOPT_ENTITY_SET

Return the profile the entity field of which matches the specified *EntityName* and the profiles pertaining to any groups in which *EntityName* is a member that contribute to the cumulative authority for the specified entity. You cannot specify this and MQAUTHOPT_NAME_EXPLICIT.

You can also optionally specify:

MQAUTHOPT_NAME_AS_WILDCARD

Interpret *ProfileName* as a filter on the profile name of the authority records. If you do not specify this attribute and *ProfileName* contains wildcard characters, it is interpreted as a generic profile and only those authority records where the generic profile names match the value of *ProfileName* are returned.

You cannot specify MQAUTHOPT_NAME_AS_WILDCARD if you also specify MQAUTHOPT_ENTITY_SET.

***ProfileName* (MQCFST)**

Profile name (parameter identifier: MQCACF_AUTH_PROFILE_NAME).

This is the name of the profile for which to retrieve authorizations. Generic profile names are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all profiles having names that start with the selected character string. An asterisk on its own matches all possible names.

If you have defined a generic profile, you can return information about it by not setting MQAUTHOPT_NAME_AS_WILDCARD in *Options*.

If you set *Options* to MQAUTHOPT_NAME_AS_WILDCARD, the only valid value for *ProfileName* is a single asterisk (*). This means that all authority records that satisfy the values specified in the other parameters are returned.

Do not specify *ProfileName* if the value of *ObjectType* is MQOT_Q_MGR.

The profile name is always returned regardless of the attributes requested.

The maximum length of the string is MQ_AUTH_PROFILE_NAME_LENGTH.

***ObjectType* (MQCFIN)**

The type of object referred to by the profile (parameter identifier: MQIACF_OBJECT_TYPE).

The value can be:

MQOT_ALL

All object types. This is the default if you do not specify a value for *ObjectType*.

MQOT_AUTH_INFO

Authentication information.

MQOT_CHANNEL

Channel object.

MQOT_CLNTCONN_CHANNEL
Client-connection channel object.

MQOT_LISTENER
Listener object.

MQOT_NAMELIST
Namelist.

MQOT_PROCESS
Process.

MQOT_Q
Queue, or queues, that match the object name parameter.

MQOT_Q_MGR
Queue manager.

MQOT_SERVICE
Service object.

Optional parameters

EntityName (**MQCFST**)

Entity name (parameter identifier: MQCACF_ENTITY_NAME).

Depending on the value of *EntityType*, this is either:

- A principal name. This is the name of a user for whom to retrieve authorizations to the specified object. On WebSphere MQ for Windows, the name of the principal can optionally include a domain name, specified in this format: user@domain.
- A group name. This is the name of the user group for which to retrieve authorizations. You can specify one name only and this must be the name of an existing user group. On WebSphere MQ for Windows, you can only use local groups.

The maximum length of the string is MQ_ENTITY_NAME_LENGTH.

EntityType (**MQCFIN**)

Entity type (parameter identifier: MQIACF_ENTITY_TYPE).

The value can be:

MQZAET_GROUP

The value of the *EntityName* parameter refers to a group name.

MQZAET_PRINCIPAL

The value of the *EntityName* parameter refers to a principal name.

ProfileAttrs (**MQCFIL**)

Profile attributes (parameter identifier: MQIACF_AUTH_PROFILE_ATTRS).

The attribute list might specify the following on its own (this is the default value if the parameter is not specified):

MQIACF_ALL
All attributes.

or a combination of the following:

MQCACF_ENTITY_NAME
Entity name.

MQIACF_AUTHORIZATION_LIST

Authorization list.

MQIACF_ENTITY_TYPE

Entity type.

MQIACF_OBJECT_TYPE

Object type.

ServiceComponent (MQCFST)

Service component (parameter identifier: MQCACF_SERVICE_COMPONENT).

If installable authorization services are supported, this specifies the name of the authorization service from which to retrieve authorization.

If you omit this parameter, the authorization inquiry is made to the first installable component for the service.

The maximum length of the string is MQ_SERVICE_COMPONENT_LENGTH.

Error codes

This command might return the following in the response format header, in addition to the values shown on page “Error codes applicable to all commands” on page 27.

Reason (MQLONG)

The value can be:

MQRC_OBJECT_TYPE_ERROR

Invalid object type.

MQRC_UNKNOWN_ENTITY

Userid not authorized, or unknown.

MQRCCF_CFST_CONFLICTING_PARM

Conflicting parameters.

MQRCCF_PROFILE_NAME_ERROR

Invalid profile name.

Inquire Authority Records (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
		X	X	X	

One PCF message is returned for each authority record that is found the profile name of which matches the options specified in the Inquire Authority Records request. Each response to the Inquire Authority Records (MQCMD_INQUIRE_AUTH_RECS) command consists of the response header followed by the *QMgrName*, *Options*, *ProfileName*, and *ObjectType* structures and the requested combination of attribute parameter structures.

Always returned:

ObjectType, *Options*, *ProfileName*, *QMgrName*

Returned if requested:

AuthorizationList, *EntityName*, *EntityType*

Response data

AuthorizationList (MQCFIL)

Authorization list (parameter identifier: MQIACF_AUTHORIZATION_LIST).

This list can contain zero or more authorization values. Each returned authorization value means that any user ID in the specified group or principal has the authority to perform the operation defined by that value. The value can be:

MQAUTH_NONE

The entity has authority set to 'none'.

MQAUTH_ALT_USER_AUTHORITY

Specify an alternate user ID on an MQI call.

MQAUTH_BROWSE

Retrieve a message from a queue by issuing an MQGET call with the BROWSE option.

MQAUTH_CHANGE

Change the attributes of the specified object, using the appropriate command set.

MQAUTH_CLEAR

Clear a queue.

MQAUTH_CONNECT

Connect the application to the specified queue manager by issuing an MQCONN call.

MQAUTH_CREATE

Create objects of the specified type using the appropriate command set.

MQAUTH_DELETE

Delete the specified object using the appropriate command set.

MQAUTH_DISPLAY

Display the attributes of the specified object using the appropriate command set.

MQAUTH_INPUT

Retrieve a message from a queue by issuing an MQGET call.

MQAUTH_INQUIRE

Make an inquiry on a specific queue by issuing an MQINQ call.

MQAUTH_OUTPUT

Put a message on a specific queue by issuing an MQPUT call.

MQAUTH_PASS_ALL_CONTEXT

Pass all context.

MQAUTH_PASS_IDENTITY_CONTEXT

Pass the identity context.

MQAUTH_SET

Set attributes on a queue from the MQI by issuing an MQSET call.

MQAUTH_SET_ALL_CONTEXT

Set all context on a queue.

MQAUTH_SET_IDENTITY_CONTEXT

Set the identity context on a queue.

Use the *Count* field in the MQCFIL structure to determine how many values are returned.

EntityName (MQCFST)

Entity name (parameter identifier: MQCACF_ENTITY_NAME).

This can either be a principal name or a group name.

The maximum length of the string is MQ_ENTITY_NAME_LENGTH.

EntityType (MQCFIN)

Entity type (parameter identifier: MQIACF_ENTITY_TYPE).

The value can be:

MQZAET_GROUP

The value of the *EntityName* parameter refers to a group name.

MQZAET_PRINCIPAL

The value of the *EntityName* parameter refers to a principal name.

MQZAET_UNKNOWN

On Windows, an authority record still exists from a previous queue manager which did not originally contain entity type information.

ObjectType (MQCFIN)

Object type (parameter identifier: MQIACF_OBJECT_TYPE).

The value can be:

MQOT_AUTH_INFO

Authentication information.

MQOT_CHANNEL

Channel object.

MQOT_CLNTCONN_CHANNEL

Client-connection channel object.

MQOT_LISTENER

Listener object.

MQOT_NAMELIST

Namelist.

MQOT_PROCESS

Process.

MQOT_Q

Queue, or queues, that match the object name parameter.

MQOT_Q_MGR

Queue manager.

MQOT_SERVICE

Service object.

Options (MQCFIN)

Options used to indicate the level of information that is returned (parameter identifier: MQIACF_AUTH_OPTIONS).

ProfileName (MQCFST)

Profile name (parameter identifier: MQCACF_AUTH_PROFILE_NAME).

The maximum length of the string is MQ_AUTH_PROFILE_NAME_LENGTH.

QMgrName (**MQCFST**)

Name of the queue manager on which the Inquire command is issued (parameter identifier: MQCA_Q_MGR_NAME).

The maximum length of the string is MQ_Q_MGR_NAME_LENGTH.

Inquire Authority Service

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
		X	X	X	

The Inquire Authority Service (MQCMD_INQUIRE_AUTH_SERVICE) command retrieves information about the level of function supported by installed authority managers.

Required parameters:

AuthServiceAttrs

Optional parameters:

ServiceComponent

Required parameters

AuthServiceAttrs (**MQCFIL**)

Authority service attributes (parameter identifier: MQIACF_AUTH_SERVICE_ATTRS).

The attribute list might specify the following on its own (this is the default value if the parameter is not specified):

MQIACF_ALL

All attributes.

or a combination of the following:

MQIACF_INTERFACE_VERSION

Current interface version of the authority service.

MQIACF_USER_ID_SUPPORT

Whether the authority service supports user IDs.

Optional parameters

ServiceComponent (**MQCFST**)

Name of authorization service (parameter identifier: MQCACF_SERVICE_COMPONENT).

The name of the authorization service which is to handle the Inquire Authority Service command.

If this parameter is omitted, or specified as a blank or null string, the inquire function is called in each installed authorization service in reverse order to the order in which the services have been installed, until all authorization services have been called or until one returns a value of MQZCI_STOP in the Continuation field.

The maximum length of the string is MQ_SERVICE_COMPONENT_LENGTH.

Error codes

This command might return the following in the response format header, in addition to the values shown on page “Error codes applicable to all commands” on page 27.

Reason (MQLONG)

The value can be:

MQRC_SELECTOR_ERROR

Attribute selector not valid.

MQRC_UNKNOWN_COMPONENT_NAME

Unknown service component name.

Inquire Authority Service (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
		X	X	X	

The response to the Inquire Authority Service (MQCMD_INQUIRE_AUTH_SERVICE) command consists of the response header followed by the *ServiceComponent* structure and the requested combination of attribute parameter structures.

Always returned:

ServiceComponent

Returned if requested:

InterfaceVersion, *UserIDSupport*

Response data

InterfaceVersion (MQCFIN)

Interface version (parameter identifier: MQIACF_INTERFACE_VERSION).

This is the current interface version of the OAM.

ServiceComponent (MQCFSL)

Name of authorization service (parameter identifier: MQCACF_SERVICE_COMPONENT).

If you included a specific value for *ServiceComponent* on the Inquire Authority Service command, this field contains the name of the authorization service that handled the command. If you did not include a specific value for *ServiceComponent* on the Inquire Authority Service command, the list contains the names of all the installed authorization services.

The maximum length of each element in the list is MQ_SERVICE_COMPONENT_LENGTH.

UserIDSupport (MQCFIN)

User ID support (parameter identifier: MQIACF_USER_ID_SUPPORT).

The value can be:

MQUIDSUPP_YES

The authority service supports user IDs.

MQUIDSUPP_NO

The authority service does not support user IDs.

Inquire CF Structure

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The Inquire CF Structure (MQCMD_INQUIRE_CF_STRUC) command returns information about the attributes of one or more CF application structures.

Note: This command is supported only on z/OS when the queue manager is a member of a queue-sharing group.

Required parameters:

CFStrucName

Optional parameters:

CFStrucAttrs, *IntegerFilterCommand*, *StringFilterCommand*

Required parameters

***CFStrucName* (MQCFST)**

CF Structure name (parameter identifier: MQCA_CF_STRUC_NAME).

Specifies the name of the CF application structure about which information is to be returned.

Generic CF structure names are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all CF application structures having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length is MQ_CF_STRUC_NAME_LENGTH.

Optional parameters

***CFStrucAttrs* (MQCFIL)**

CF application structure attributes (parameter identifier: MQIACF_CF_STRUC_ATTRS).

The attribute list might specify the following on its own (this is the default value used if the parameter is not specified):

MQIACF_ALL

All attributes.

or a combination of the following:

MQCA_ALTERATION_DATE

The date on which the definition was last altered.

MQCA_ALTERATION_TIME

The time at which the definition was last altered.

MQCA_CF_STRUC_DESC

Description of CF application structure.

MQCA_CF_STRUC_NAME

Name of CF application structure.

MQIA_CF_LEVEL

Functional capability level for the CF application structure.

MQIA_CF_RECOVER

Whether CF recovery for the application structure is supported.

IntegerFilterCommand (MQCFIF)

Integer filter command descriptor. The parameter identifier must be any integer type parameter of those allowed in *CFStrucAttrs* except MQIACF_ALL. Use this to restrict the output from the command by specifying a filter condition. See “MQCFIF - PCF integer filter parameter” on page 427 for information about using this filter condition.

If you specify an integer filter, you cannot also specify a string filter using the *StringFilterCommand* parameter.

StringFilterCommand (MQCFSF)

String filter command descriptor. The parameter identifier must be any string type parameter of those allowed in *CFStrucAttrs* except MQCA_CF_STRUC_NAME. Use this to restrict the output from the command by specifying a filter condition. See “MQCFSF - PCF string filter parameter” on page 434 for information about using this filter condition.

If you specify a string filter, you cannot also specify an integer filter using the *IntegerFilterCommand* parameter.

Inquire CF Structure (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The response to the Inquire CF Structure (MQCMD_INQUIRE_CF_STRUC) command consists of the response header followed by the *CFStrucName* structure and the requested combination of attribute parameter structures. If a generic CF application structure name was specified, one such message is generated for each CF application structure found.

Always returned:

CFStrucName

Returned if requested:

AlterationDate, AlterationTime, CFLevel, CFStrucDesc, Recovery

Response data

AlterationDate (MQCFST)

Alteration date (parameter identifier: MQCA_ALTERATION_DATE).

The date on which the definition was last altered, in the form yyyy-mm-dd.

The maximum length of the string is MQ_DATE_LENGTH.

AlterationTime (MQCFST)

Alteration time (parameter identifier: MQCA_ALTERATION_TIME).

The time at which the definition was last altered, in the form hh.mm.ss.

The maximum length of the string is MQ_TIME_LENGTH.

CFLevel (MQCFIN)

The functional capability level for this CF application structure (parameter identifier: MQIA_CF_LEVEL).

Specifies the functional capability level for the CF application structure. The value can be:

- 1 A CF structure that can be "auto-created" by a queue manager at command level 520.
- 2 A CF structure at command level 520 that can only be created or deleted by a queue manager at command level 530 or greater. This is the default *CFLevel* for queue managers at command level 530 or greater.
- 3
- 4 A CF structure at command level 600. This *CFLevel* is required if you want to use persistent messages on shared queues, or for message grouping, or both.

CFStrucDesc (MQCFST)

The description of the CF structure (parameter identifier: MQCA_CF_STRUC_DESC).

The maximum length is MQ_CF_STRUC_DESC_LENGTH.

CFStrucName (MQCFST)

CF Structure name (parameter identifier: MQCA_CF_STRUC_NAME).

The maximum length is MQ_CF_STRUC_NAME_LENGTH.

Recovery (MQCFIN)

Recovery (parameter identifier: MQCA_CF_RECOVER).

Specifies whether CF recovery is supported for the application structure. The value can be:

MQCFR_YES

Recovery is supported.

MQCFR_NO

Recovery is not supported.

Inquire CF Structure Names

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The Inquire CF Structure Names (MQCMD_INQUIRE_CF_STRUC_NAMES) command inquires for a list of CF application structure names that match the generic CF structure name specified.

Note: This command is supported only on z/OS when the queue manager is a member of a queue-sharing group.

Required parameters:

CFStrucName

Optional parameters:

None

Required parameters

***CFStrucName* (MQCFST)**

CF Structure name (parameter identifier: MQCA_CF_STRUC_NAME).

Specifies the name of the CF application structure about which information is to be returned.

Generic CF structure names are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all CF application structures having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length is MQ_CF_STRUC_NAME_LENGTH.

Inquire CF Structure Names (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The response to the Inquire CF Structure Names (MQCMD_INQUIRE_CF_STRUC_NAMES) command consists of the response header followed by a single parameter structure giving zero or more names that match the specified CF application structure name.

Always returned:

CFStrucNames

Returned if requested:

None

Response data

***CFStrucNames* (MQCFSL)**

List of CF application structure names (parameter identifier: MQCACF_CF_STRUC_NAMES).

Inquire CF Structure Status

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The Inquire CF Structure Status (MQCMD_INQUIRE_CF_STRUC_STATUS) command inquires about the status of a CF application structure.

Note: This command is supported only on z/OS when the queue manager is a member of a queue-sharing group.

Required parameters:

CFStrucName

Optional parameters:

CFStatusType, *IntegerFilterCommand*, *StringFilterCommand*

Required parameters

***CFStrucName* (MQCFST)**

CF Structure name (parameter identifier: MQCA_CF_STRUC_NAME).

Specifies the name of the CF application structure for which status information is to be returned.

Generic CF structure names are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all CF application structures having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length is MQ_CF_STRUC_NAME_LENGTH.

Optional parameters

***CFStatusType* (MQCFIN)**

Status information type (parameter identifier: MQIACF_CF_STATUS_TYPE).

Specifies the type of status information you want to be returned. You can specify one of the following:

MQIACF_CF_STATUS_SUMMARY

Summary status information for the CF application structure. This is the default.

MQIACF_CF_STATUS_CONNECT

Connection status information for each CF application structure for each active queue manager.

MQIACF_CF_STATUS_BACKUP

Backup status information for each CF application structure.

***IntegerFilterCommand* (MQCFIF)**

Integer filter command descriptor. The parameter identifier must be any integer type parameter of those possible in the response data except MQIACF_CF_STATUS_TYPE. Use this to restrict the output from the command by specifying a filter condition. See “MQCFIF - PCF integer filter parameter” on page 427 for information about using this filter condition.

If you specify an integer filter, you cannot also specify a string filter using the *StringFilterCommand* parameter.

***StringFilterCommand* (MQCFSF)**

String filter command descriptor. The parameter identifier must be any string type parameter of those possible in the response data except MQCA_CF_STRUC_NAME. Use this to restrict the output from the command by specifying a filter condition. See “MQCFSF - PCF string filter parameter” on page 434 for information about using this filter condition.

If you specify a string filter, you cannot also specify an integer filter using the *IntegerFilterCommand* parameter.

Inquire CF Structure Status (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The response to the Inquire CF Structure Status (MQCMD_INQUIRE_CF_STRUC_STATUS) command consists of the response header followed by the *CFStrucName* and *CFStatusType* structures and a set of attribute parameter structures determined by the value of *CFStatusType* in the Inquire command.

Always returned:

CFStrucName, *CFStatusType*.

CFStatusType specifies the type of status information being returned. The value can be:

MQIACF_CF_STATUS_SUMMARY

Summary status information for the CF application structure. This is the default.

MQIACF_CF_STATUS_CONNECT

Connection status information for each CF application structure for each active queue manager.

MQIACF_CF_STATUS_BACKUP

Backup status information for each CF application structure.

Returned if *CFStatusType* is MQIACF_CF_STATUS_SUMMARY:

CFStrucStatus, *CFStrucType*, *EntriesMax*, *EntriesUsed*, *FailDate*, *FailTime*, *SizeMax*, *SizeUsed*

Returned if *CFStatusType* is MQIACF_CF_STATUS_CONNECT:

CFStrucStatus, *FailDate*, *FailTime*, *QMgrName*, *SysName*

Returned if *CFStatusType* is MQIACF_CF_STATUS_BACKUP:

BackupDate, *BackupEndRBA*, *BackupSize*, *BackupStartRBA*, *BackupTime*, *CFStrucStatus*, *FailDate*, *FailTime*, *LogQMgrNames*, *QmgrName*

Response data

BackupDate (MQCFST)

The date, in the form yyyy-mm-dd, on which the last successful backup was taken for this CF application structure (parameter identifier: MQCACF_BACKUP_DATE).

The maximum length of the string is MQ_DATE_LENGTH.

BackupEndRBA (MQCFST)

The backup dataset end RBA for the end of the last successful backup taken for this CF application structure (parameter identifier: MQCACF_CF_STRUC_BACKUP_END).

The maximum length of the string is MQ_RBA_LENGTH.

BackupSize (MQCFIN)

The size, in megabytes, of the last successful backup taken for this CF application structure (parameter identifier: MQIACF_CF_STRUC_BACKUP_SIZE).

BackupStartRBA (**MQCFST**)

The backup dataset start RBA for the start of the last successful backup taken for this CF application structure (parameter identifier: MQCACF_CF_STRUC_BACKUP_START).

The maximum length of the string is MQ_RBA_LENGTH.

BackupTime (**MQCFST**)

The end time, in the form hh.mm.ss, of the last successful backup taken for this CF application structure (parameter identifier: MQCACF_BACKUP_TIME).

The maximum length of the string is MQ_TIME_LENGTH.

CFStatusType (**MQCFIN**)

Status information type (parameter identifier: MQIACF_CF_STATUS_TYPE).

Specifies the type of status information being returned. The value can be:

MQIACF_CF_STATUS_SUMMARY

Summary status information for the CF application structure. This is the default.

MQIACF_CF_STATUS_CONNECT

Connection status information for each CF application structure for each active queue manager.

MQIACF_CF_STATUS_BACKUP

Backup status information for each CF application structure.

CFStrucName (**MQCFST**)

CF Structure name (parameter identifier: MQCA_CF_STRUC_NAME).

The maximum length is MQ_CF_STRUC_NAME_LENGTH.

CFStrucStatus (**MQCFIN**)

CF Structure status (parameter identifier: MQIACF_CF_STRUC_STATUS).

The status of the CF application structure. If *CFStatusType* is MQIACF_CF_STATUS_SUMMARY, the value can be:

MQCFSTATUS_ACTIVE

The structure is active.

MQCFSTATUS_FAILED

The structure has failed.

MQCFSTATUS_NOT_FOUND

The structure is not allocated in the CF, but has been defined to DB2[®].

MQCFSTATUS_IN_BACKUP

The structure is in the process of being backed up.

MQCFSTATUS_IN_RECOVER

The structure is in the process of being recovered.

MQCFSTATUS_UNKNOWN

The status of the CF structure is unknown because, for example, DB2 may be unavailable.

If *CFStatusType* is MQIACF_CF_STATUS_CONNECT, the value can be:

MQCFSTATUS_ACTIVE

The structure is connected to this queue manager.

MQCFSTATUS_FAILED

The queue manager connection to this structure has failed.

MQCFSTATUS_NONE

The structure has never been connected to this queue manager.

If *CFStatusType* is MQIACF_CF_STATUS_BACKUP, the value can be:

MQCFSTATUS_ACTIVE

The structure is active.

MQCFSTATUS_FAILED

The structure has failed.

MQCFSTATUS_NONE

The structure has never been backed up.

MQCFSTATUS_IN_BACKUP

The structure is in the process of being backed up.

MQCFSTATUS_IN_RECOVER

The structure is in the process of being recovered.

***CFStructType* (MQCFIN)**

CF Structure type (parameter identifier: MQIACF_CF_STRUC_TYPE).

The value can be:

MQCFTYPE_ADMIN

This is the CF administration structure.

MQCFTYPE_APPL

This is a CF application structure.

***EntriesMax* (MQCFIN)**

Number of CF list entries defined for this CF application structure (parameter identifier: MQIACF_CF_STRUC_ENTRIES_MAX).

***EntriesUsed* (MQCFIN)**

Number of CF list entries defined for this CF application structure that are in use (parameter identifier: MQIACF_CF_STRUC_ENTRIES_USED).

***FailDate* (MQCFST)**

The date, in the form yyyy-mm-dd, on which this CF application structure failed (parameter identifier: MQCACF_FAIL_DATE).

If *CFStatusType* is MQIACF_CF_STATUS_CONNECT, this is the date on which the queue manager lost connectivity to this application structure. For the other values of *CFStatusType*, this is the date on which this CF application structure failed. This parameter is only applicable when *CFStrucStatus* is MQCFSTATUS_FAILED or MQCFSTATUS_IN_RECOVER.

The maximum length of the string is MQ_DATE_LENGTH.

***FailTime* (MQCFST)**

The time, in the form hh.mm.ss, that this CF application structure failed (parameter identifier: MQCACF_FAIL_TIME).

If *CFStatusType* is MQIACF_CF_STATUS_CONNECT, this is the time that the queue manager lost connectivity to this application structure. For the other values of *CFStatusType*, this is the time that this CF application structure failed. This parameter is only applicable when *CFStrucStatus* is MQCFSTATUS_FAILED or MQCFSTATUS_IN_RECOVER.

The maximum length of the string is MQ_TIME_LENGTH.

***LogQMgrNames* (MQCFSL)**

A list of queue managers, the logs of which are required to perform a recovery (parameter identifier: MQCACF_CF_STRUC_LOG_Q_MGRS).

The maximum length of each name is MQ_Q_MGR_NAME_LENGTH.

***QMgrName* (MQCFST)**

Queue manager name (parameter identifier: MQCA_Q_MGR_NAME).

This is the name of the queue manager. If *CFStatusType* is MQIACF_CF_STATUS_BACKUP, this is the name of the queue manager that took the last successful backup.

The maximum length is MQ_Q_MGR_NAME_LENGTH.

***SizeMax* (MQCFIN)**

Size of the CF application structure (parameter identifier: MQIACF_CF_STRUC_SIZE_MAX).

This is the size, in kilobytes, of the CF application structure.

***SizeUsed* (MQCFIN)**

Percentage of the CF application structure that is in use (parameter identifier: MQIACF_CF_STRUC_SIZE_USED).

This is the percentage of the size of the CF application structure that is in use.

***SysName* (MQCFST)**

Queue manager name (parameter identifier: MQCACF_SYSTEM_NAME).

This is the name of the z/OS image of the queue manager that last connected to the CF application structure.

The maximum length is MQ_SYSTEM_NAME_LENGTH.

Inquire Channel

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Inquire Channel (MQCMD_INQUIRE_CHANNEL) command inquires about the attributes of WebSphere MQ channel definitions.

Required parameters:

ChannelName

Optional parameters:

ChannelAttrs, *ChannelType*, *CommandScope*, , *IntegerFilterCommand*, *QSGDisposition*, *StringFilterCommand*

Required parameters

***ChannelName* (MQCFST)**

Channel name (parameter identifier: MQCACH_CHANNEL_NAME).

Generic channel names are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all channels having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length of the string is MQ_CHANNEL_NAME_LENGTH.

Optional parameters

ChannelAttrs (MQCFIL)

Channel attributes (parameter identifier: MQIACF_CHANNEL_ATTRS).

The attribute list can specify the following on its own (this is the default value used if the parameter is not specified):

MQIACF_ALL

All attributes.

or a combination of the parameters in the following table:

Parameter	Sender	Server	Receiver	Requester	Client conn	Server conn	Cluster sender	Cluster receiver
MQCA_ALTERATION_DATE Date on which the definition was last altered	X	X	X	X	X	X	X	X
MQCA_ALTERATION_TIME Time at which the definition was last altered	X	X	X	X	X	X	X	X
MQCA_CLUSTER_NAME Name of local queue manager							X	X
MQCA_CLUSTER_NAMELIST Name of local queue manager							X	X
MQCA_Q_MGR_NAME Name of local queue manager					X			
MQCACH_CHANNEL_NAME Channel name. You cannot use this attribute as a filter keyword.	X	X	X	X	X	X	X	X
MQCACH_CONNECTION_NAME Connection name	X	X		X	X		X	X
MQCACH_DESC Description	X	X	X	X	X	X	X	X
MQCACH_LOCAL_ADDRESS Local communications address for the channel	X	X		X	X		X	X
MQCACH_MCA_NAME Message channel agent name	X	X		X			X	
MQCACH_MCA_USER_ID MCA user identifier	X	X	X	X		X	X	X
MQCACH_MODE_NAME Mode name	X	X		X	X		X	X

Parameter	Sender	Server	Receiver	Requester	Client conn	Server conn	Cluster sender	Cluster receiver
MQCACH_MR_EXIT_NAME Message-retry exit name			X	X				X
MQCACH_MR_EXIT_USER_DATA Message-retry exit name			X	X				X
MQCACH_MSG_EXIT_NAME Message exit name	X	X	X	X			X	X
MQCACH_MSG_EXIT_USER_DATA Message exit user data	X	X	X	X			X	X
MQCACH_PASSWORD Password	X	X		X	X		X	
MQCACH_RCV_EXIT_NAME Receive exit name	X	X	X	X	X	X	X	X
MQCACH_RCV_EXIT_USER_DATA Receive exit user data	X	X	X	X	X	X	X	X
MQCACH_SEC_EXIT_NAME Security exit name	X	X	X	X	X	X	X	X
MQCACH_SEC_EXIT_USER_DATA Security exit user data	X	X	X	X	X	X	X	X
MQCACH_SEND_EXIT_NAME Send exit name	X	X	X	X	X	X	X	X
MQCACH_SEND_EXIT_USER_DATA Send exit user data	X	X	X	X	X	X	X	X
MQCACH_SSL_CIPHER_SPEC SSL cipher spec	X	X	X	X	X	X	X	X
MQIACH_SSL_CLIENT_AUTH SSL client authentication	X	X	X	X		X		X
MQCACH_SSL_PEER_NAME SSL peer name	X	X	X	X	X	X	X	X
MQCACH_TP_NAME Transaction program name	X	X		X	X	X	X	X
MQCACH_USER_ID User identifier	X	X		X	X		X	
MQCACH_XMIT_Q_NAME Transmission queue name	X	X						

Parameter	Sender	Server	Receiver	Requester	Client conn	Server conn	Cluster sender	Cluster receiver
MQIA_MONITORING_CHANNEL Online monitoring data collection	X	X	X	X		X	X	X
MQIA_STATISTICS_CHANNEL Online statistics collection	X	X	X	X			X	X
MQIACH_BATCH_HB Value to use for batch heartbeating	X	X					X	X
MQIACH_BATCH_INTERVAL Batch wait interval (seconds)	X	X					X	X
MQIACH_BATCH_SIZE Batch size	X	X	X	X			X	X
MQIACH_CHANNEL_TYPE Channel type	X	X	X	X	X	X	X	X
MQIACH_CLWL_CHANNEL_PRIORITY Cluster workload channel priority							X	X
MQIACH_CLWL_CHANNEL_RANK Cluster workload channel rank							X	X
MQIACH_CLWL_CHANNEL_WEIGHT Cluster workload channel weight							X	X
MQIACH_DATA_CONVERSION Whether sender should convert application data	X	X					X	X
MQIACH_DISC_INTERVAL Disconnection interval	X	X				X	X	X
MQIACH_HB_INTERVAL Heartbeat interval (seconds)	X	X	X	X	X	X	X	X
MQIACH_HDR_COMPRESSION List of header data compression techniques supported by the channel	X	X	X	X	X	X	X	X
MQIACH_KEEP_ALIVE_INTERVAL KeepAlive interval	X	X	X	X	X	X	X	X
MQIACH_LONG_RETRY Long retry count	X	X					X	X
MQIACH_LONG_TIMER Long timer	X	X					X	X

Parameter	Sender	Server	Receiver	Requester	Client conn	Server conn	Cluster sender	Cluster receiver
MQIACH_MAX_MSG_LENGTH Maximum message length	X	X	X	X	X	X	X	X
MQIACH_MCA_TYPE MCA type	X	X		X			X	X
MQIACH_MR_COUNT Message retry count			X	X				X
MQIACH_MSG_COMPRESSION List of message data compression techniques supported by the channel	X	X	X	X	X	X	X	X
MQIACH_MR_INTERVAL Message retry interval (milliseconds)			X	X				X
MQIACH_NPM_SPEED Speed of nonpersistent messages	X	X	X	X			X	X
MQIACH_PUT_AUTHORITY Put authority			X	X		X		X
MQIACH_SEQUENCE_NUMBER_WRAP Sequence number wrap	X	X	X	X			X	X
MQIACH_SHORT_RETRY Short retry count	X	X					X	X
MQIACH_SHORT_TIMER Short timer	X	X					X	X
MQIACH_XMIT_PROTOCOL_TYPE Transport (transmission protocol) type	X	X	X	X	X	X	X	X

Channel Type (MQCFIN)

Channel type (parameter identifier: MQIACH_CHANNEL_TYPE).

If this parameter is present, eligible channels are limited to those of the specified type. Any attribute selector specified in the *ChannelAttrs* list which is only valid for channels of a different type or types is ignored; no error is raised.

If this parameter is not present (or if MQCHT_ALL is specified), channels of all types are eligible. Each attribute specified must be a valid channel attribute selector (that is, it must be one of those in the following list), but it might not be applicable to all (or any) of the channels actually returned. Channel attribute selectors that are valid but not applicable to the channel are ignored, no error messages occur, and no attribute is returned.

The value can be:

MQCHT_SENDER
Sender.

MQCHT_SERVER	Server.
MQCHT_RECEIVER	Receiver.
MQCHT_REQUESTER	Requester.
MQCHT_SVRCONN	Server-connection (for use by clients).
MQCHT_CLNTCONN	Client connection.
MQCHT_CLUSRCVR	Cluster-receiver.
MQCHTCLUSSDR	Cluster-sender.
MQCHT_ALL	All types.

The default value if this parameter is not specified is MQCHT_ALL.

Note: If this parameter is present, it must occur immediately after the *ChannelName* parameter on platforms other than z/OS. Failure to do this can result in a MQRCCF_MSG_LENGTH_ERROR error message.

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

You cannot use *CommandScope* as a parameter to filter on.

IntegerFilterCommand (MQCFIF)

Integer filter command descriptor. The parameter identifier must be any integer type parameter allowed in *ChannelAttrs* except MQIACF_ALL. Use this to restrict the output from the command by specifying a filter condition. See “MQCFIF - PCF integer filter parameter” on page 427 for information about using this filter condition.

If you specify an integer filter for channel type, you cannot also specify the *ChannelType* parameter.

If you specify an integer filter, you cannot also specify a string filter using the *StringFilterCommand* parameter.

QSGDisposition (**MQCFIN**)

Disposition of the object within the group (parameter identifier: **MQIA_QSG_DISP**). This parameter applies to z/OS only.

Specifies the disposition of the object for which information is to be returned (that is, where it is defined and how it behaves). The value can be:

MQQSGD_LIVE

The object is defined as **MQQSGD_Q_MGR** or **MQQSGD_COPY**. This is the default value if the parameter is not specified.

MQQSGD_ALL

The object is defined as **MQQSGD_Q_MGR** or **MQQSGD_COPY**.

If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this option also displays information for objects defined with **MQQSGD_GROUP**.

If **MQQSGD_LIVE** is specified or defaulted, or if **MQQSGD_ALL** is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

MQQSGD_COPY

The object is defined as **MQQSGD_COPY**.

MQQSGD_GROUP

The object is defined as **MQQSGD_GROUP**. This is permitted only in a shared queue environment.

MQQSGD_Q_MGR

The object is defined as **MQQSGD_Q_MGR**.

MQQSGD_PRIVATE

The object is defined as either **MQQSGD_Q_MGR** or **MQQSGD_COPY**. Note that **MQQSGD_PRIVATE** returns the same information as **MQQSGD_LIVE**.

You cannot use *QSGDisposition* as a parameter to filter on.

StringFilterCommand (**MQCFSF**)

String filter command descriptor. The parameter identifier must be any string type parameter allowed in *ChannelAttrs* except **MQCACH_CHANNEL_NAME** and **MQCACH_MCA_NAME**. Use this to restrict the output from the command by specifying a filter condition. See “**MQCFSF - PCF string filter parameter**” on page 434 for information about using this filter condition.

If you specify a string filter, you cannot also specify an integer filter using the *IntegerFilterCommand* parameter.

Error codes

This command might return the following in the response format header, in addition to the values shown on page “Error codes applicable to all commands” on page 27.

Reason (**MQLONG**)

The value can be:

MQRCCF_CHANNEL_NAME_ERROR

Channel name error.

MQRCCF_CHANNEL_NOT_FOUND

Channel not found.

MQRCCF_CHANNEL_TYPE_ERROR

Channel type not valid.

Inquire Channel (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The response to the Inquire Channel (MQCMD_INQUIRE_CHANNEL) command consists of the response header followed by the *ChannelName* and *ChannelType* structures (and on z/OS only, the *QSGDisposition* structure), and the requested combination of attribute parameter structures (where applicable). If a generic channel name was specified, one such message is generated for each channel found.

Always returned:

ChannelName, *ChannelType*, *DefaultChannelDisposition*, *QSGDisposition*

Returned if requested:

AlterationDate, *AlterationTime*, *BatchHeartbeat*, *BatchInterval*,
BatchSize, *ChannelDesc*, *ChannelMonitoring*, *ChannelStatistics*,
ClusterName, *ClusterNameList*, *CLWLChannelPriority*, *CLWLChannelRank*,
CLWLChannelWeight, *ConnectionName*, *DataConversion*, *DiscInterval*,
HeaderCompression, *HeartbeatInterval*, *KeepAliveInterval*, *LocalAddress*,
LongRetryCount, *LongRetryInterval*, *MaxMsgLength*, *MCAName*, *MCAType*,
MCAUserIdentifier, *MessageCompression*, *ModeName*, *MsgExit*, *MsgRetryCount*,
MsgRetryExit, *MsgRetryInterval*, *MsgRetryUserData*, *MsgUserData*,
NetworkPriority, *NonPersistentMsgSpeed*, *Password*, *PutAuthority*,
QMgrName, *ReceiveExit*, *ReceiveUserData*, *SecurityExit*, *SecurityUserData*,
SendExit, *SendUserData*, *SeqNumberWrap*, , *ShortRetryCount*,
ShortRetryInterval, *SSLCipherSpec*, *SSLClientAuth*, *SSLPeerName*, *TpName*,
TransportType, *UserIdentifier*, *XmitQName*

Response data

***AlterationDate* (MQCFST)**

Alteration date, in the form yyyy-mm-dd (parameter identifier:
MQCA_ALTERATION_DATE).

The date when the information was last altered.

***AlterationTime* (MQCFST)**

Alteration time, in the form hh.mm.ss (parameter identifier:
MQCA_ALTERATION_TIME).

The time when the information was last altered.

***BatchHeartbeat* (MQCFIN)**

The value being used for the batch heartbeating (parameter identifier:
MQIACH_BATCH_HB).

The value can be between 0 and 999 999. A value of 0 indicates that heartbeating is not in use.

BatchInterval (**MQCFIN**)

Batch interval (parameter identifier: MQIACH_BATCH_INTERVAL).

BatchSize (**MQCFIN**)

Batch size (parameter identifier: MQIACH_BATCH_SIZE).

ChannelDesc (**MQCFST**)

Channel description (parameter identifier: MQCACH_DESC).

The maximum length of the string is MQ_CHANNEL_DESC_LENGTH.

ChannelMonitoring (**MQCFIN**)

Online monitoring data collection (parameter identifier: MQIA_MONITORING_CHANNEL).

The value can be:

MQMON_OFF

Online monitoring data collection is turned off for this channel.

MQMON_Q_MGR

The value of the queue manager's *ChannelMonitoring* parameter is inherited by the channel.

MQMON_LOW

Online monitoring data collection is turned on, with a low rate of data collection, for this channel unless the queue manager's *ChannelMonitoring* parameter is MQMON_NONE.

MQMON_MEDIUM

Online monitoring data collection is turned on, with a moderate rate of data collection, for this channel unless the queue manager's *ChannelMonitoring* parameter is MQMON_NONE.

MQMON_HIGH

Online monitoring data collection is turned on, with a high rate of data collection, for this channel unless the queue manager's *ChannelMonitoring* parameter is MQMON_NONE.

ChannelName (**MQCFST**)

Channel name (parameter identifier: MQCACH_CHANNEL_NAME).

The maximum length of the string is MQ_CHANNEL_NAME_LENGTH.

ChannelStatistics (**MQCFIN**)

Statistics data collection (parameter identifier: MQIA_STATISTICS_CHANNEL).

The value can be:

MQMON_OFF

Statistics data collection is turned off for this channel.

MQMON_Q_MGR

The value of the queue manager's *ChannelStatistics* parameter is inherited by the channel.

MQMON_LOW

Statistics data collection is turned on, with a low rate of data collection, for this channel unless the queue manager's *ChannelStatistics* parameter is MQMON_NONE.

MQMON_MEDIUM

Statistics data collection is turned on, with a moderate rate of data collection, for this channel unless the queue manager's *ChannelStatistics* parameter is MQMON_NONE.

MQMON_HIGH

Statistics data collection is turned on, with a high rate of data collection, for this channel unless the queue manager's *ChannelStatistics* parameter is MQMON_NONE.

This is valid only on AIX, HP-UX, Linux, i5/OS, Solaris, and Windows.

ChannelType (MQCFIN)

Channel type (parameter identifier: MQIACH_CHANNEL_TYPE).

The value can be:

MQCHT_SENDER

Sender.

MQCHT_SERVER

Server.

MQCHT_RECEIVER

Receiver.

MQCHT_REQUESTER

Requester.

MQCHT_SVRCONN

Server-connection (for use by clients).

MQCHT_CLNTCONN

Client connection.

MQCHT_CLUSRCVR

Cluster-receiver.

MQCHTCLUSSDR

Cluster-sender.

ClusterName (MQCFST)

Cluster name (parameter identifier: MQCA_CLUSTER_NAME).

ClusterNamelist (MQCFSL)

Cluster namelist (parameter identifier: MQCA_CLUSTER_NAMELIST).

CLWLChannelPriority (MQCFIN)

Channel priority (parameter identifier:
MQIACH_CLWL_CHANNEL_PRIORITY).

CLWLChannelRank (MQCFIN)

Channel rank (parameter identifier: MQIACH_CLWL_CHANNEL_RANK).

CLWLChannelWeight (MQCFIN)

Channel weighting (parameter identifier:
MQIACH_CLWL_CHANNEL_WEIGHT).

ConnectionName (MQCFST)

Connection name (parameter identifier: MQCACH_CONNECTION_NAME).

The maximum length of the string is MQ_CONN_NAME_LENGTH. On z/OS, it is MQ_LOCAL_ADDRESS_LENGTH.

DataConversion (MQCFIN)

Whether sender should convert application data (parameter identifier:
MQIACH_DATA_CONVERSION).

The value can be:

MQCDC_NO_SENDER_CONVERSION

No conversion by sender.

MQCDC_SENDER_CONVERSION

Conversion by sender.

***DiscInterval* (MQCFIN)**

Disconnection interval (parameter identifier: MQIACH_DISC_INTERVAL).

***HeaderCompression* (MQCFIL)**

Header data compression techniques supported by the channel (parameter identifier: MQIACH_HDR_COMPRESSION). For sender, server, cluster-sender, cluster-receiver, and client-connection channels, the values specified are in order of preference.

The value can be one, or more, of

MQCOMPRESS_NONE

No header data compression is performed.

MQCOMPRESS_SYSTEM

Header data compression is performed.

***HeartbeatInterval* (MQCFIN)**

Heartbeat interval (parameter identifier: MQIACH_HB_INTERVAL).

***KeepAliveInterval* (MQCFIN)**

KeepAlive interval (parameter identifier: MQIACH_KEEP_ALIVE_INTERVAL).

***LocalAddress* (MQCFST)**

Local communications address for the channel (parameter identifier: MQCACH_LOCAL_ADDRESS).

The maximum length of the string is MQ_LOCAL_ADDRESS_LENGTH.

***LongRetryCount* (MQCFIN)**

Long retry count (parameter identifier: MQIACH_LONG_RETRY).

***LongRetryInterval* (MQCFIN)**

Long timer (parameter identifier: MQIACH_LONG_TIMER).

***MaxMsgLength* (MQCFIN)**

Maximum message length (parameter identifier: MQIACH_MAX_MSG_LENGTH).

***MCAName* (MQCFST)**

Message channel agent name (parameter identifier: MQCACH_MCA_NAME).

The maximum length of the string is MQ_MCA_NAME_LENGTH.

***MCAType* (MQCFIN)**

Message channel agent type (parameter identifier: MQIACH_MCA_TYPE).

The value can be:

MQMCAT_PROCESS

Process.

MQMCAT_THREAD

Thread (Windows only).

***MCAUserIdentity* (MQCFST)**

Message channel agent user identifier (parameter identifier: MQCACH_MCA_USER_ID).

The maximum length of the MCA user identifier depends on the environment in which the MCA is running. MQ_MCA_USER_ID_LENGTH gives the maximum length for the environment for which your application is running. MQ_MAX_MCA_USER_ID_LENGTH gives the maximum for all supported environments.

On Windows, the user identifier may be qualified with the domain name in the following format:

user@domain

MessageCompression (MQCFIL)

Message data compression techniques supported by the channel (parameter identifier: MQIACH_MSG_COMPRESSION). For sender, server, cluster-sender, cluster-receiver, and client-connection channels, the values specified are in order of preference.

The value can be one, or more, of:

MQCOMPRESS_NONE

No message data compression is performed.

MQCOMPRESS_RLE

Message data compression is performed using run-length encoding.

MQCOMPRESS_ZLIBFAST

Message data compression is performed using ZLIB encoding with speed prioritized.

MQCOMPRESS_ZLIBHIGH

Message data compression is performed using ZLIB encoding with compression prioritized.

MQCOMPRESS_ANY

Any compression technique supported by the queue manager can be used. This is only valid for receiver, requester, and server-connection channels.

ModeName (MQCFST)

Mode name (parameter identifier: MQCACH_MODE_NAME).

The maximum length of the string is MQ_MODE_NAME_LENGTH.

MsgExit (MQCFST)

Message exit name (parameter identifier: MQCACH_MSG_EXIT_NAME).

The maximum length of the exit name depends on the environment in which the exit is running. MQ_EXIT_NAME_LENGTH gives the maximum length for the environment in which your application is running.

MQ_MAX_EXIT_NAME_LENGTH gives the maximum for all supported environments.

In the following environments, if more than one message exit has been defined for the channel, the list of names is returned in an MQCFSL structure instead of an MQCFST structure: AIX, HP-UX, i5/OS, Solaris, Linux, and Windows. An MQCFSL structure is always used on z/OS.

MsgRetryCount (MQCFIN)

Message retry count (parameter identifier: MQIACH_MR_COUNT).

MsgRetryExit (MQCFST)

Message retry exit name (parameter identifier: MQCACH_MR_EXIT_NAME).

The maximum length of the exit name depends on the environment in which the exit is running. MQ_EXIT_NAME_LENGTH gives the maximum length for the environment in which your application is running.

MQ_MAX_EXIT_NAME_LENGTH gives the maximum for all supported environments.

MsgRetryInterval (MQCFIN)

Message retry interval (parameter identifier: MQIACH_MR_INTERVAL).

MsgRetryUserData (MQCFST)

Message retry exit user data (parameter identifier: MQCACH_MR_EXIT_USER_DATA).

The maximum length of the string is MQ_EXIT_DATA_LENGTH.

MsgUserData (MQCFST)

Message exit user data (parameter identifier: MQCACH_MSG_EXIT_USER_DATA).

The maximum length of the string is MQ_EXIT_DATA_LENGTH.

In the following environments, if more than one message exit user data string has been defined for the channel, the list of strings is returned in an MQCFSL structure instead of an MQCFST structure: AIX, HP-UX i5/OS, Solaris, Linux, and Windows. An MQCFSL structure is always used on z/OS.

NetworkPriority (MQCFIN)

Network priority (parameter identifier: MQIACH_NETWORK_PRIORITY).

NonPersistentMsgSpeed (MQCFIN)

Speed at which non-persistent messages are to be sent (parameter identifier: MQIACH_NPM_SPEED).

The value can be:

MQNPMS_NORMAL

Normal speed.

MQNPMS_FAST

Fast speed.

Password (MQCFST)

Password (parameter identifier: MQCACH_PASSWORD).

If a nonblank password is defined, it is returned as asterisks. Otherwise, it is returned as blanks.

The maximum length of the string is MQ_PASSWORD_LENGTH. However, only the first 10 characters are used.

PutAuthority (MQCFIN)

Put authority (parameter identifier: MQIACH_PUT_AUTHORITY).

The value can be:

MQPA_DEFAULT

Default user identifier is used.

MQPA_CONTEXT

Context user identifier is used.

QMgrName (MQCFST)

Queue manager name (parameter identifier: MQCA_Q_MGR_NAME).

The maximum length of the string is MQ_Q_MGR_NAME_LENGTH.

QSGDisposition (MQCFIN)

QSG disposition (parameter identifier: MQIA_QSG_DISP).

Specifies the disposition of the object (that is, where it is defined and how it behaves). This parameter is valid only on z/OS. The value can be:

MQQSGD_COPY

The object is defined as MQQSGD_COPY.

MQQSGD_GROUP

The object is defined as MQQSGD_GROUP.

MQQSGD_Q_MGR

The object is defined as MQQSGD_Q_MGR.

ReceiveExit (MQCFST)

Receive exit name (parameter identifier: MQCACH_RCV_EXIT_NAME).

The maximum length of the exit name depends on the environment in which the exit is running. MQ_EXIT_NAME_LENGTH gives the maximum length for the environment in which your application is running.

MQ_MAX_EXIT_NAME_LENGTH gives the maximum for all supported environments.

In the following environments, if more than one receive exit has been defined for the channel, the list of names is returned in an MQCFSL structure instead of an MQCFST structure: AIX, HP-UX, i5/OS, Solaris, Linux, and Windows. An MQCFSL structure is always used on z/OS.

ReceiveUserData (MQCFST)

Receive exit user data (parameter identifier:
MQCACH_RCV_EXIT_USER_DATA).

The maximum length of the string is MQ_EXIT_DATA_LENGTH.

In the following environments, if more than one receive exit user data string has been defined for the channel, the list of strings is returned in an MQCFSL structure instead of an MQCFST structure: AIX, HP-UX, i5/OS, Solaris, Linux, and Windows. An MQCFSL structure is always used on z/OS.

SecurityExit (MQCFST)

Security exit name (parameter identifier: MQCACH_SEC_EXIT_NAME).

The maximum length of the exit name depends on the environment in which the exit is running. MQ_EXIT_NAME_LENGTH gives the maximum length for the environment in which your application is running.

MQ_MAX_EXIT_NAME_LENGTH gives the maximum for all supported environments.

SecurityUserData (MQCFST)

Security exit user data (parameter identifier:
MQCACH_SEC_EXIT_USER_DATA).

The maximum length of the string is MQ_EXIT_DATA_LENGTH.

SendExit (MQCFST)

Send exit name (parameter identifier: MQCACH_SEND_EXIT_NAME).

The maximum length of the exit name depends on the environment in which the exit is running. MQ_EXIT_NAME_LENGTH gives the maximum length for the environment in which your application is running.

MQ_MAX_EXIT_NAME_LENGTH gives the maximum for all supported environments.

In the following environments, if more than one send exit has been defined for the channel, the list of names is returned in an MQCFSL structure instead of an MQCFST structure: AIX, HP-UX, i5/OS, Solaris, Linux, and Windows. An MQCFSL structure is always used on z/OS.

SendUserData (**MQCFST**)

Send exit user data (parameter identifier:
MQCACH_SEND_EXIT_USER_DATA).

The maximum length of the string is MQ_EXIT_DATA_LENGTH.

In the following environments, if more than one send exit user data string has been defined for the channel, the list of strings is returned in an MQCFSL structure instead of an MQCFST structure: AIX, HP-UX, i5/OS, Solaris, Linux, and Windows. An MQCFSL structure is always used on z/OS.

SeqNumberWrap (**MQCFIN**)

Sequence wrap number (parameter identifier:
MQIACH_SEQUENCE_NUMBER_WRAP).

ShortRetryCount (**MQCFIN**)

Short retry count (parameter identifier: MQIACH_SHORT_RETRY).

ShortRetryInterval (**MQCFIN**)

Short timer (parameter identifier: MQIACH_SHORT_TIMER).

SSLCipherSpec (**MQCFST**)

CipherSpec (parameter identifier: MQCACH_SSL_CIPHER_SPEC).

The length of the string is MQ_SSL_CIPHER_SPEC_LENGTH.

SSLClientAuth (**MQCFIN**)

Client authentication (parameter identifier: MQIACH_SSL_CLIENT_AUTH).

The value can be

MQSCA_REQUIRED

Client authentication required

MQSCA_OPTIONAL

Client authentication is optional.

Defines whether WebSphere MQ requires a certificate from the SSL client.

SSLPeerName (**MQCFST**)

Peer name (parameter identifier: MQCACH_SSL_PEER_NAME).

The length of the string is MQ_SSL_PEER_NAME_LENGTH. On z/OS, it is MQ_SSL_SHORT_PEER_NAME_LENGTH.

Specifies the filter to use to compare with the Distinguished Name of the certificate from the peer queue manager or client at the other end of the channel. (A Distinguished Name is the identifier of the SSL certificate.) If the Distinguished Name in the certificate received from the peer does not match the SSLPEER filter, the channel does not start.

TpName (**MQCFST**)

Transaction program name (parameter identifier: MQCACH_TP_NAME).

The maximum length of the string is MQ_TP_NAME_LENGTH.

TransportType (**MQCFIN**)

Transmission protocol type (parameter identifier:
MQIACH_XMIT_PROTOCOL_TYPE).

The value may be:

MQXPT_LU62
LU 6.2.

MQXPT_TCP
TCP.

MQXPT_NETBIOS
NetBIOS.

MQXPT_SPX
SPX.

MQXPT_DECNET
DECnet.

UserIdentifier (MQCFST)

Task user identifier (parameter identifier: MQCACH_USER_ID).

The maximum length of the string is MQ_USER_ID_LENGTH. However, only the first 10 characters are used.

XmitQName (MQCFST)

Transmission queue name (parameter identifier: MQCACH_XMIT_Q_NAME).

The maximum length of the string is MQ_Q_NAME_LENGTH.

Inquire Channel Initiator

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The Inquire Channel Initiator (MQCMD_INQUIRE_CHANNEL_INIT) command returns information about the channel initiator.

Required parameters:

None

Optional parameters:

CommandScope

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

Inquire Channel Initiator (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The response to the Inquire Channel Initiator (MQCMD_INQUIRE_CHANNEL_INIT) command consists of one response with a series of attribute parameter structures showing the status of the channel initiator (shown by the *ChannelInitiatorStatus* parameter), and one response for each listener (shown by the *ListenerStatus* parameter).

Always returned (one message with channel initiator information):

*ActiveChannels, ActiveChannelsMax, ActiveChannelsPaused,
ActiveChannelsRetrying, ActiveChannelsStarted, ActiveChannelsStopped,
AdaptersMax, AdaptersStarted, ChannelInitiatorStatus, CurrentChannels,
CurrentChannelsLU62, CurrentChannelsMax, CurrentChannelsTCP,
DispatchersMax, DispatchersStarted, SSLTasksStarted, TCPName*

Always returned (one message for each listener):

InboundDisposition, ListenerStatus, TransportType

Returned if applicable for the listener:

IPAddress, LUName, Port

Response data - channel initiator information

ActiveChannels (MQCFIN)

The number of active channel connections (parameter identifier: MQIACH_ACTIVE_CHL).

ActiveChannelsMax (MQCFIN)

The requested number of active channel connections (parameter identifier: MQIACH_ACTIVE_CHL_MAX).

ActiveChannelsPaused (MQCFIN)

The number of active channel connections that have paused, waiting to become active, because the limit for active channels has been reached (parameter identifier: MQIACH_ACTIVE_CHL_PAUSED).

ActiveChannelsRetrying (MQCFIN)

The number of active channel connections that are attempting to reconnect following a temporary error (parameter identifier: MQIACH_ACTIVE_CHL_RETRY).

ActiveChannelsStarted (MQCFIN)

The number of active channel connections that have started (parameter identifier: MQIACH_ACTIVE_CHL_STARTED).

ActiveChannelsStopped (MQCFIN)

The number of active channel connections that have stopped, requiring manual intervention (parameter identifier: MQIACH_ACTIVE_CHL_STOPPED).

AdaptersMax (MQCFIN)

The requested number of adapter subtasks (parameter identifier: MQIACH_ADAPS_MAX).

AdaptersStarted (**MQCFIN**)

The number of active adapter subtasks (parameter identifier: MQIACH_ADAPS_STARTED).

ChannelInitiatorStatus (**MQCFIN**)

Status of the channel initiator (parameter identifier: MQIACF_CHINIT_STATUS).

The value can be:

MQSVC_STATUS_STOPPED

The channel initiator is not running.

MQSVC_STATUS_RUNNING

The channel initiator is fully initialized and is running.

CurrentChannels (**MQCFIN**)

The number of current channel connections (parameter identifier: MQIACH_CURRENT_CHL).

CurrentChannelsLU62 (**MQCFIN**)

The number of current LU 6.2 channel connections (parameter identifier: MQIACH_CURRENT_CHL_LU62).

CurrentChannelsMax (**MQCFIN**)

The requested number of channel connections (parameter identifier: MQIACH_CURRENT_CHL_MAX).

CurrentChannelsTCP (**MQCFIN**)

The number of current TCP/IP channel connections (parameter identifier: MQIACH_CURRENT_CHL_TCP).

DispatchersMax (**MQCFIN**)

The requested number of dispatchers (parameter identifier: MQIACH_DISPS_MAX).

DispatchersStarted (**MQCFIN**)

The number of active dispatchers (parameter identifier: MQIACH_DISPS_STARTED).

SSLTasksMax (**MQCFIN**)

The requested number of SSL server subtasks (parameter identifier: MQIACH_SSLTASKS_MAX).

SSLTasksStarted (**MQCFIN**)

The number of active SSL server subtasks (parameter identifier: MQIACH_SSLTASKS_STARTED).

TCPName (**MQCFST**)

TCP system name (parameter identifier: MQCACH_TCP_NAME).

The maximum length is MQ_TCP_NAME_LENGTH.

Response data - listener information

InboundDisposition (**MQCFIN**)

Inbound transmission disposition (parameter identifier: MQIACH_INBOUND_DISP).

Specifies the disposition of the inbound transmissions that the listener handles.
The value can be:

MQINBD_Q_MGR

Handling for transmissions directed to the queue manager. This is the default.

MQINBD_GROUP

Handling for transmissions directed to the queue-sharing group. This is permitted only if there is a shared queue manager environment.

***IPAddress* (MQCFST)**

IP address on which the listener listens (parameter identifier: MQCACH_IP_ADDRESS).

***ListenerStatus* (MQCFIN)**

Listener status (parameter identifier: MQIACH_LISTENER_STATUS).

The value can be:

MQSVC_STATUS_RUNNING

The listener has started.

MQSVC_STATUS_STOPPED

The listener has stopped.

MQSVC_STATUS_RETRYING

The listener is retrying.

***LUName* (MQCFST)**

LU name on which the listener listens (parameter identifier: MQCACH_LU_NAME).

The maximum length is MQ_LU_NAME_LENGTH.

***Port* (MQCFIN)**

Port number on which the listener listens (parameter identifier: MQIACH_PORT_NUMBER).

***TransportType* (MQCFIN)**

Transmission protocol type that the listener is using (parameter identifier: MQIACH_XMIT_PROTOCOL_TYPE).

The value can be:

MQXPT_LU62

LU62.

MQXPT_TCP

TCP.

Inquire Channel Listener

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
		X	X	X	

The Inquire Channel Listener (MQCMD_INQUIRE_LISTENER) command inquires about the attributes of existing WebSphere MQ listeners.

Required parameters:

ListenerName

Optional parameters:

IntegerFilterCommand, *ListenerAttrs*, *StringFilterCommand*,
TransportType

Required parameters

ListenerName (**MQCFST**)

Listener name (parameter identifier: MQCACH_LISTENER_NAME).

This is the name of the listener whose attributes are required. Generic listener names are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all listeners having names that start with the selected character string. An asterisk on its own matches all possible names.

The listener name is always returned regardless of the attributes requested.

The maximum length of the string is MQ_LISTENER_NAME_LENGTH.

Optional parameters

IntegerFilterCommand (**MQCFIF**)

Integer filter command descriptor. The parameter identifier must be any integer type parameter allowed in *ListenerAttrs* except MQIACF_ALL. Use this to restrict the output from the command by specifying a filter condition. See “MQCFIF - PCF integer filter parameter” on page 427 for information about using this filter condition.

If you specify an integer filter, you cannot also specify a string filter using the *StringFilterCommand* parameter.

ListenerAttrs (**MQCFIL**)

Listener attributes (parameter identifier: MQIACF_LISTENER_ATTRS).

The attribute list might specify the following on its own (this is the default value if the parameter is not specified):

MQIACF_ALL

All attributes.

or a combination of the following:

MQCA_ALTERATION_DATE

Date on which the definition was last altered.

MQCA_ALTERATION_TIME

Time at which the definition was last altered.

MQCACH_IP_ADDRESS

IP address for the listener.

MQCACH_LISTENER_DESC

Description of listener definition.

MQCACH_LISTENER_NAME

Name of listener definition.

MQCACH_LOCAL_NAME

NetBIOS local name that the listener uses. This is valid only on Windows.

MQCACH_TP_NAME

The LU 6.2 transaction program name. This is valid only on Windows.

MQIACH_ADAPTER

Adapter number on which NetBIOS listens. This is valid only on Windows.

MQIACH_BACKLOG

Number of concurrent connection requests that the listener supports.

MQIACH_COMMAND_COUNT

Number of commands that the listener can use. This is valid only on Windows.

MQIACH_LISTENER_CONTROL

Specifies when the queue manager should start and stop the listener.

MQIACH_NAME_COUNT

Number of names that the listener can use. This is valid only on Windows.

MQIACH_PORT

Port number.

MQIACH_SESSION_COUNT

Number of sessions that the listener can use. This is valid only on Windows.

MQIACH_SOCKET

SPX socket on which to listen. This is valid only on Windows.

StringFilterCommand (MQCFSF)

String filter command descriptor. The parameter identifier must be any string type parameter allowed in *ListenerAttrs* except MQCACH_LISTENER_NAME. Use this to restrict the output from the command by specifying a filter condition. See “MQCFSF - PCF string filter parameter” on page 434 for information about using this filter condition.

If you specify a string filter, you cannot also specify an integer filter using the *IntegerFilterCommand* parameter.

TransportType (MQCFIN)

Transport protocol type (parameter identifier: MQIACH_XMIT_PROTOCOL_TYPE).

If you specify this parameter, information is returned relating only to those listeners defined with the specified transport protocol type. If you specify an attribute in the *ListenerAttrs* list which is valid only for listeners of a different transport protocol type, it is ignored and no error is raised. If you specify this parameter, it must occur immediately after the *ListenerName* parameter.

If you do not specify this parameter, or if you specify it with a value of MQXPT_ALL, information about all listeners is returned. Valid attributes in the *ListenerAttrs* list which are not applicable to the listener are ignored, and no error messages are issued. The value can be:

MQXPT_ALL

All transport types.

MQXPT_LU62

SNA LU 6.2. This is valid only on Windows.

MQXPT_NETBIOS

NetBIOS. This is valid only on Windows.

MQXPT_SPX

SPX. This is valid only on Windows.

MQXPT_TCP

Transmission Control Protocol /Internet Protocol (TCP /IP).

Inquire Channel Listener (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
		X	X	X	

The response to the Inquire Channel Listener (MQCMD_INQUIRE_LISTENER) command consists of the response header followed by the *ListenerName* structure and the requested combination of attribute parameter structures. If a generic listener name was specified, one such message is generated for each listener found.

Always returned:

ListenerName

Returned if requested:

Adapter, AlterationDate, AlterationTime, Backlog, Commands, IPAddress, ListenerDesc, LocalName, NetbiosNames, Port, Sessions, Socket, StartMode, TPname, TransportType

Response data

AlterationDate (MQCFST)

Alteration date (parameter identifier: MQCA_ALTERATION_DATE).

The date, in the form yyyy-mm-dd, on which the information was last altered.

AlterationTime (MQCFST)

Alteration time (parameter identifier: MQCA_ALTERATION_TIME).

The time, in the form hh.mm.ss, at which the information was last altered.

Adapter (MQCFIN)

Adapter number (parameter identifier: MQIACH_ADAPTER).

The adapter number on which NetBIOS listens. This is valid only on Windows.

Backlog (MQCFIN)

Backlog (parameter identifier: MQIACH_BACKLOG).

The number of concurrent connection requests that the listener supports.

Commands (MQCFIN)

Adapter number (parameter identifier: MQIACH_COMMAND_COUNT).

The number of commands that the listener can use. This is valid only on Windows.

IPAddress (MQCFST)

IP address (parameter identifier: MQCACH_IP_ADDRESS).

IP address for the listener specified in IPv4 dotted decimal, IPv6 hexadecimal notation, or alphanumeric hostname form.

The maximum length of the string is MQ_CONN_NAME_LENGTH

ListenerDesc (**MQCFST**)

Description of listener definition (parameter identifier: MQCACH_LISTENER_DESC).

The maximum length of the string is MQ_LISTENER_DESC_LENGTH.

ListenerName (**MQCFST**)

Name of listener definition (parameter identifier: MQCACH_LISTENER_NAME).

The maximum length of the string is MQ_LISTENER_NAME_LENGTH.

LocalName (**MQCFST**)

NetBIOS local name (parameter identifier: MQCACH_LOCAL_NAME).

The NetBIOS local name that the listener uses. This is valid only on Windows.

The maximum length of the string is MQ_CONN_NAME_LENGTH

NetbiosNames (**MQCFIN**)

NetBIOS names (parameter identifier: MQIACH_NAME_COUNT).

The number of names that the listener supports. This is valid only on Windows.

Port (**MQCFIN**)

Port number (parameter identifier: MQIACH_PORT).

The port number for TCP/IP. This is valid only if the value of *TransportType* is MQXPT_TCP.

Sessions (**MQCFIN**)

NetBIOS sessions (parameter identifier: MQIACH_SESSION_COUNT).

The number of sessions that the listener can use. This is valid only on Windows.

Socket (**MQCFIN**)

SPX socket number (parameter identifier: MQIACH_SOCKET).

The SPX socket on which to listen. This is valid only if the value of *TransportType* is MQXPT_SPX.

StartMode (**MQCFIN**)

Service mode (parameter identifier: MQIACH_LISTENER_CONTROL).

Specifies how the listener is to be started and stopped. The value can be:

MQSVC_CONTROL_MANUAL

The listener is not to be started automatically or stopped automatically. It is to be controlled by user command. This is the default value.

MQSVC_CONTROL_Q_MGR

The listener being defined is to be started and stopped at the same time as the queue manager is started and stopped.

MQSVC_CONTROL_Q_MGR_START

The listener is to be started at the same time as the queue manager is started, but is not request to stop when the queue manager is stopped.

TPName (**MQCFST**)

Transaction program name (parameter identifier: MQCACH_TP_NAME).

The LU 6.2 transaction program name. This is valid only on Windows.

The maximum length of the string is MQ_TP_NAME_LENGTH

TransportType (MQCFIN)

Transmission protocol (parameter identifier: MQIACH_XMIT_PROTOCOL_TYPE).

The value can be:

MQXPT_TCP
TCP.

MQXPT_LU62

LU 6.2. This is valid only on Windows.

MQXPT_NETBIOS

NetBIOS. This is valid only on Windows.

MQXPT_SPX

SPX. This is valid only on Windows.

Inquire Channel Listener Status

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
		X	X	X	

The Inquire Channel Listener Status (MQCMD_INQUIRE_LISTENER_STATUS) command inquires about the status of one or more WebSphere MQ listener instances. You must specify the name of a listener for which you want to receive status information. You can specify a listener by using either a specific listener name or a generic listener name. By using a generic listener name, you can display either:

- Status information for all listener definitions, by using a single asterisk (*), or
- Status information for one or more listeners that match the specified name.

Required parameters:

ListenerName

Optional parameters:

IntegerFilterCommand, *ListenerStatusAttrs*, *StringFilterCommand*

Required parameters

ListenerName (MQCFST)

Listener name (parameter identifier: MQCACH_LISTENER_NAME).

Generic listener names are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all listeners having names that start with the selected character string. An asterisk on its own matches all possible names.

The listener name is always returned, regardless of the attributes requested.

The maximum length of the string is MQ_LISTENER_NAME_LENGTH.

Optional parameters

IntegerFilterCommand (MQCFIF)

Integer filter command descriptor. The parameter identifier must be any integer type parameter allowed in *ListenerStatusAttrs* except MQIACF_ALL.

Use this to restrict the output from the command by specifying a filter condition. See “MQCFIF - PCF integer filter parameter” on page 427 for information about using this filter condition.

If you specify an integer filter, you cannot also specify a string filter using the *StringFilterCommand* parameter.

ListenerStatusAttrs (MQCFIL)

Listener status attributes (parameter identifier:
MQIACF_LISTENER_STATUS_ATTRS).

The attribute list can specify the following on its own (this is the default value used if the parameter is not specified):

MQIACF_ALL

All attributes.

or a combination of the following:

MQCACH_IP_ADDRESS

Listener’s IP address.

MQCACH_LISTENER_DESC

Description of listener definition.

MQCACH_LISTENER_NAME

Name of listener definition.

MQCACH_LISTENER_START_DATE

The date on which the listener was started.

MQCACH_LISTENER_START_TIME

The time at which the listener was started.

MQCACH_LOCAL_NAME

NetBIOS local name that the listener uses. This is valid only on Windows.

MQCACH_TP_NAME

LU6.2 transaction program name. This is valid only on Windows.

MQIACF_PROCESS_ID

Operating system process identifier associated with the listener.

MQIACH_ADAPTER

Adapter number on which NetBIOS listens. This is valid only on Windows.

MQIACH_BACKLOG

Number of concurrent connection requests that the listener supports.

MQIACH_COMMAND_COUNT

Number of commands that the listener can use. This is valid only on Windows.

MQIACH_LISTENER_CONTROL

How the listener is to be started and stopped.

MQIACH_LISTENER_STATUS

Current status of the listener.

MQIACH_NAME_COUNT

Number of names that the listener can use. This is valid only on Windows.

MQIACH_PORT

Port number for TCP/IP.

MQIACH_SESSION_COUNT

Number of sessions that the listener can use. This is valid only on Windows.

MQIACH_SOCKET

SPX socket. This is valid only on Windows.

MQIACH_XMIT_PROTOCOL_TYPE

Transport type.

StringFilterCommand (MQCFSF)

String filter command descriptor. The parameter identifier must be any string type parameter allowed in *ListenerStatusAttrs* except MQCACH_LISTENER_NAME. Use this to restrict the output from the command by specifying a filter condition. See “MQCFSF - PCF string filter parameter” on page 434 for information about using this filter condition.

If you specify a string filter, you cannot also specify an integer filter using the *IntegerFilterCommand* parameter.

Error codes

This command might return the following in the response format header, in addition to the values shown on page “Error codes applicable to all commands” on page 27.

Reason (MQLONG)

The value can be:

MQRCCF_LSTR_STATUS_NOT_FOUND

Listener status not found.

Inquire Channel Listener Status (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
		X	X	X	

The response to the Inquire Channel Listener Status (MQCMD_INQUIRE_LISTENER_STATUS) command consists of the response header followed by the *ListenerName* structure and the requested combination of attribute parameter structures. If a generic listener name was specified, one such message is generated for each listener found.

Always returned:

ListenerName

Returned if requested:

Adapter, Backlog, ChannelCount, Commands, IPAddress, ListenerDesc, LocalName, NetbiosNames, Port, ProcessId, Sessions, Socket, StartDate, StartMode, StartTime, Status, TPname, TransportType

Response data

Adapter (MQCFIN)

Adapter number (parameter identifier: MQIACH_ADAPTER).

The adapter number on which NetBIOS listens.

Backlog (MQCFIN)

Backlog (parameter identifier: MQIACH_BACKLOG).

The number of concurrent connection requests that the listener supports.

Commands (MQCFIN)

Adapter number (parameter identifier: MQIACH_COMMAND_COUNT).

The number of commands that the listener can use.

IPAddress (MQCFST)

IP address (parameter identifier: MQCACH_IP_ADDRESS).

IP address for the listener specified in IPv4 dotted decimal, IPv6 hexadecimal notation, or alphanumeric hostname form.

The maximum length of the string is MQ_CONN_NAME_LENGTH

ListenerDesc (MQCFST)

Description of listener definition (parameter identifier: MQCACH_LISTENER_DESC).

The maximum length of the string is MQ_LISTENER_DESC_LENGTH.

ListenerName (MQCFST)

Name of listener definition (parameter identifier: MQCACH_LISTENER_NAME).

The maximum length of the string is MQ_LISTENER_NAME_LENGTH.

LocalName (MQCFST)

NetBIOS local name (parameter identifier: MQCACH_LOCAL_NAME).

The NetBIOS local name that the listener uses.

The maximum length of the string is MQ_CONN_NAME_LENGTH

NetbiosNames (MQCFIN)

NetBIOS names (parameter identifier: MQIACH_NAME_COUNT).

The number of names that the listener supports.

Port (MQCFIN)

Port number (parameter identifier: MQIACH_PORT).

The port number for TCP/IP.

ProcessId (MQCFIN)

Process identifier (parameter identifier: MQIACF_PROCESS_ID).

The operating system process identifier associated with the listener.

Sessions (MQCFIN)

NetBIOS sessions (parameter identifier: MQIACH_SESSION_COUNT).

The number of sessions that the listener can use.

Socket (MQCFIN)

SPX socket number (parameter identifier: MQIACH_SOCKET).

The SPX socket on which the listener is to listen.

StartDate (MQCFST)

Start date (parameter identifier: MQIACH_LISTENER_START_DATE).

The date, in the form yyyy-mm-dd, on which the listener was started.

The maximum length of the string is MQ_DATE_LENGTH

StartMode (MQCFIN)

Service mode (parameter identifier: MQIACH_LISTENER_CONTROL).

Specifies how the listener is to be started and stopped. The value can be:

MQSVC_CONTROL_MANUAL

The listener is not to be started automatically or stopped automatically.

It is to be controlled by user command. This is the default value.

MQSVC_CONTROL_Q_MGR

The listener being defined is to be started and stopped at the same time as the queue manager is started and stopped.

MQSVC_CONTROL_Q_MGR_START

The listener is to be started at the same time as the queue manager is started, but is not request to stop when the queue manager is stopped.

StartTime (MQCFST)

Start date (parameter identifier: MQIACH_LISTENER_START_TIME).

The time, in the form hh.mm.ss, at which the listener was started.

The maximum length of the string is MQ_TIME_LENGTH

Status (MQCFIN)

Listener status (parameter identifier: MQIACH_LISTENER_STATUS).

The current status of the listener. The value can be:

MQSVC_STATUS_STARTING

The listener is in the process of initializing.

MQSVC_STATUS_RUNNING

The listener is running.

MQSVC_STATUS_STOPPING

The listener is stopping.

TPName (MQCFST)

Transaction program name (parameter identifier: MQCACH_TP_NAME).

The LU 6.2 transaction program name.

The maximum length of the string is MQ_TP_NAME_LENGTH

TransportType (MQCFIN)

Transmission protocol (parameter identifier: MQIACH_XMIT_PROTOCOL_TYPE).

The value can be:

MQXPT_TCP

TCP.

MQXPT_LU62

LU 6.2. This is valid only on Windows.

MQXPT_NETBIOS

NetBIOS. This is valid only on Windows.

MQXPT_SPX

SPX. This is valid only on Windows.

Inquire Channel Names

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Inquire Channel Names (MQCMD_INQUIRE_CHANNEL_NAMES) command inquires a list of WebSphere MQ channel names that match the generic channel name, and the optional channel type specified.

Required parameters:

ChannelName

Optional parameters:

ChannelType, CommandScope, QSGDisposition

Required parameters

ChannelName (MQCFST)

Channel name (parameter identifier: MQCACH_CHANNEL_NAME).

Generic channel names are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all objects having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length of the string is MQ_CHANNEL_NAME_LENGTH.

Optional parameters

ChannelType (MQCFIN)

Channel type (parameter identifier: MQIACH_CHANNEL_TYPE).

If present, this parameter limits the channel names returned to channels of the specified type.

The value can be:

MQCHT_SENDER

Sender.

MQCHT_SERVER

Server.

MQCHT_RECEIVER

Receiver.

MQCHT_REQUESTER

Requester.

MQCHT_SVRCONN

Server-connection (for use by clients).

MQCHT_CLNTCONN

Client connection.

MQCHT_CLUSRCVR
Cluster-receiver.

MQCHTCLUSSDR
Cluster-sender.

MQCHT_ALL
All types.

The default value if this parameter is not specified is MQCHT_ALL, which means that channels of all types except MQCHT_CLNTCONN are eligible.

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

QSGDisposition (MQCFIN)

Disposition of the object within the group (parameter identifier: MQIA_QSG_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object for which information is to be returned (that is, where it is defined and how it behaves). The value can be:

MQQSGD_LIVE

The object is defined as MQQSGD_Q_MGR or MQQSGD_COPY. This is the default value if the parameter is not specified.

MQQSGD_ALL

The object is defined as MQQSGD_Q_MGR or MQQSGD_COPY.

If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this option also displays information for objects defined with MQQSGD_GROUP.

If MQQSGD_LIVE is specified or defaulted, or if MQQSGD_ALL is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

MQQSGD_COPY

The object is defined as MQQSGD_COPY.

MQQSGD_GROUP

The object is defined as MQQSGD_GROUP. This is permitted only in a shared queue environment.

MQQSGD_Q_MGR

The object is defined as MQQSGD_Q_MGR.

MQQSGD_PRIVATE

The object is defined with either MQQSGD_Q_MGR or MQQSGD_COPY. Note that MQQSGD_PRIVATE returns the same information as MQQSGD_LIVE.

Error codes

This command might return the following in the response format header, in addition to the values shown on page "Error codes applicable to all commands" on page 27.

Reason (MQLONG)

The value can be:

MQRCCF_CHANNEL_NAME_ERROR

Channel name error.

MQRCCF_CHANNEL_TYPE_ERROR

Channel type not valid.

Inquire Channel Names (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The response to the Inquire Channel Names (MQCMD_INQUIRE_CHANNEL_NAMES) command consists of the response header followed by a single parameter structure giving zero or more names that match the specified channel name.

In addition to this, on z/OS only, two parameter structures (each with the same number of entries as the *ChannelNames* structure) are returned. Each entry in the first structure, *ChannelTypes*, indicates the channel type of the object with the corresponding entry in the *ChannelNames* structure. Each entry in the second structure, *QSGDispositions* indicates the disposition of the object with the corresponding entry in the *ChannelNames* structure.

Always returned:

ChannelNames, *ChannelTypes*, *QSGDispositions*

Returned if requested:

None

Response data

ChannelNames (MQCFSL)

List of channel names (parameter identifier: MQCACH_CHANNEL_NAMES).

ChannelTypes (MQCFIL)

List of channel types (parameter identifier: MQIACH_CHANNEL_TYPES).

Possible values for fields in this structure are those permitted for the *ChannelType* parameter, except MQCHT_ALL.

QSGDispositions (MQCFIL)

List of QSG dispositions (parameter identifier: MQIACF_QSG_DISPS). This is valid only on z/OS. The value can be:

MQQSGD_COPY

The object is defined as MQQSGD_COPY.

MQQSGD_GROUP

The object is defined as MQQSGD_GROUP. This is permitted only in a shared queue environment.

MQQSGD_Q_MGR

The object is defined as MQQSGD_Q_MGR.

Inquire Channel Status

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Inquire Channel Status (MQCMD_INQUIRE_CHANNEL_STATUS) command inquires about the status of one or more channel instances.

You must specify the name of the channel for which you want to inquire status information. This can be a specific channel name or a generic channel name. By using a generic channel name, you can inquire either:

- Status information for all channels, or
- Status information for one or more channels that match the specified name.

You must also specify whether you want:

- The current status data (of current channels only), or
- The saved status data of all channels, or
- On z/OS only, the short status data of the channel.

Status for all channels that meet the selection criteria is given, whether the channels were defined manually or automatically.

There are three classes of data available for channel status. These are **saved**, **current**, and **short**. The status fields available for saved data are a subset of the fields available for current data and are called **common** status fields. Note that although the common data *fields* are the same, the data *values* might be different for saved and current status. The rest of the fields available for current data are called **current-only** status fields.

- **Saved** data consists of the common status fields. This data is reset at the following times:
 - For all channels:
 - When the channel enters or leaves STOPPED or RETRY state
 - For a sending channel:
 - Before requesting confirmation that a batch of messages has been received
 - When confirmation has been received
 - For a receiving channel:
 - Just before confirming that a batch of messages has been received
 - For a server connection channel:
 - No data is saved

Therefore, a channel which has never been current will not have any saved status.

- **Current** data consists of the common status fields and current-only status fields. The data fields are continually updated as messages are sent or received.
- **Short** data consists of the queue manager name that owns the channel instance. This class of data is available only on z/OS.

This method of operation has the following consequences:

- An inactive channel might not have any saved status –if it has never been current or has not yet reached a point where saved status is reset.
- The “common” data fields might have different values for saved and current status.
- A current channel always has current status and might have saved status.

Channels can be current or inactive:

Current channels

These are channels that have been started, or on which a client has connected, and that have not finished or disconnected normally. They may not yet have reached the point of transferring messages, or data, or even of establishing contact with the partner. Current channels have **current** status and can also have **saved** or **short**status.

The term **Active** is used to describe the set of current channels which are not stopped.

Inactive channels

These are channels that have either not been started or on which a client has not connected, or that have finished or disconnected normally. (Note that if a channel is stopped, it is not yet considered to have finished normally – and is, therefore, still current.) Inactive channels have either **saved** status or no status at all.

There can be more than one instance of a receiver, requester, cluster-sender, cluster-receiver, or server-connection channel current at the same time (the requester is acting as a receiver). This occurs if several senders, at different queue managers, each initiate a session with this receiver, using the same channel name. For channels of other types, there can only be one instance current at any time.

For all channel types, however, there can be more than one set of saved status information available for a given channel name. At most one of these sets relates to a current instance of the channel, the rest relate to previously current instances. Multiple instances arise if different transmission queue names or connection names have been used in connection with the same channel. This can happen in the following cases:

- At a sender or server:
 - If the same channel has been connected to by different requesters (servers only),
 - If the transmission queue name has been changed in the definition, or
 - If the connection name has been changed in the definition.
- At a receiver or requester:
 - If the same channel has been connected to by different senders or servers, or
 - If the connection name has been changed in the definition (for requester channels initiating connection).

The number of sets returned for a given channel can be limited by using the *XmitQName*, *ConnectionName* and *ChannelInstanceType* parameters.

Required parameters:

ChannelName

Optional parameters:

ChannelDisposition, *ChannelInstanceAttrs*, *ChannelInstanceType*,
CommandScope, *ConnectionName*, *IntegerFilterCommand*,
StringFilterCommand, *XmitQName*

Required parameters

ChannelName (**MQCFST**)

Channel name (parameter identifier: MQCACH_CHANNEL_NAME).

Generic channel names are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all objects having names that start with the selected character string. An asterisk on its own matches all possible names.

The channel name is always returned, regardless of the instance attributes requested.

The maximum length of the string is MQ_CHANNEL_NAME_LENGTH.

Optional parameters

ChannelDisposition (**MQCFIN**)

Channel disposition (parameter identifier: MQIACH_CHANNEL_DISP). This parameter applies to z/OS only.

Specifies the disposition of the channels for which information is to be returned. The value can be:

MQCHLD_ALL

Returns requested status information for private channels.

In a shared queue environment where the command is being executed on the queue manager where it was issued, or if *ChannelInstanceType* has a value of MQOT_CURRENT_CHANNEL, this option also displays the requested status information for shared channels.

MQCHLD_PRIVATE

Returns requested status information for private channels.

MQCHLD_SHARED

Returns requested status information for shared channels.

The status information that is returned for various combinations of *ChannelDisposition*, *CommandScope*, and status type, is summarized in Table 6, Table 7 on page 208, and Table 8 on page 208.

Table 6. *ChannelDisposition* and *CommandScope* for Inquire Channel Status, Current

<i>ChannelDisposition</i>	<i>CommandScope</i> blank or local queue manager	<i>CommandScope</i> (<i>qmgr-name</i>)	<i>CommandScope</i> (*)
MQCHLD_PRIVATE	Common and current-only status for current private channels on the local queue manager	Common and current-only status for current private channels on the named queue manager	Common and current-only status for current private channels on all queue managers

Table 6. ChannelDisposition and CommandScope for Inquire Channel Status, Current (continued)

<i>ChannelDisposition</i>	<i>CommandScope blank or local queue manager</i>	<i>CommandScope(qmgr-name)</i>	<i>CommandScope(*)</i>
MQCHLD_SHARED	Common and current-only status for current shared channels on the local queue manager	Common and current-only status for current shared channels on the named queue manager	Common and current-only status for current shared channels on all queue managers
MQCHLD_ALL	Common and current-only status for current private and shared channels on the local queue manager	Common and current-only status for current private and shared channels on the named queue manager	Common and current-only status for current private and shared channels on all active queue managers

Table 7. ChannelDisposition and CommandScope for Inquire Channel Status, Short

<i>ChannelDisposition</i>	<i>CommandScope blank or local queue manager</i>	<i>CommandScope(qmgr-name)</i>	<i>CommandScope(*)</i>
MQCHLD_PRIVATE	<i>ChannelStatus</i> and short status for current private channels on the local queue manager	<i>ChannelStatus</i> and short status for current private channels on the named queue manager	<i>ChannelStatus</i> and short status for current private channels on all active queue managers
MQCHLD_SHARED	<i>ChannelStatus</i> and short status for current shared channels on all active queue managers in the queue-sharing group	Not permitted	Not permitted
MQCHLD_ALL	<i>ChannelStatus</i> and short status for current private channels on the local queue manager and current shared channels in the queue-sharing group(1)	<i>ChannelStatus</i> and short status for current private channels on the named queue manager	<i>ChannelStatus</i> and short status for current private, and shared, channels on all active queue managers in the queue-sharing group(1)

Note:

1. In this case you get two separate sets of responses to the command on the queue manager where it was entered; one for MQCHLD_PRIVATE and one for MQCHLD_SHARED.

Table 8. ChannelDisposition and CommandScope for Inquire Channel Status, Saved

<i>ChannelDisposition</i>	<i>CommandScope blank or local queue manager</i>	<i>CommandScope(qmgr-name)</i>	<i>CommandScope(*)</i>
MQCHLD_PRIVATE	Common status for saved private channels on the local queue manager	Common status for saved private channels on the named queue manager	Common status for saved private channels on all active queue managers
MQCHLD_SHARED	Common status for saved shared channels on all active queue managers in the queue-sharing group	Not permitted	Not permitted
MQCHLD_ALL	Common status for saved private channels on the local queue manager and saved shared channels in the queue-sharing group	Common status for saved private channels on the named queue manager	Common status for saved private, and shared, channels on all active queue managers in the queue-sharing group

You cannot use this parameter as a filter keyword.

ChannelInstanceAttrs (MQCFIL)

Channel instance attributes (parameter identifier: MQIACH_CHANNEL_INSTANCE_ATTRS).

If status information is requested which is not relevant for the particular channel type, this is not an error. Similarly, it is not an error to request status information that is applicable only to active channels for saved channel instances. In both of these cases, no structure is returned in the response for the information concerned.

For a saved channel instance, the MQCACH_CURRENT_LUWID, MQIACH_CURRENT_MSGS, and MQIACH_CURRENT_SEQ_NUMBER attributes have meaningful information only if the channel instance is in doubt. However, the attribute values are still returned when requested, even if the channel instance is not in-doubt.

The attribute list might specify the following on its own:

MQIACF_ALL

All attributes.

This is the default value used if the parameter is not specified or it can specify a combination of the following:

Relevant for common status

The following information applies to all sets of channel status, whether or not the set is current.

MQCACH_CHANNEL_NAME

Channel name.

MQCACH_CONNECTION_NAME

Connection name.

MQCACH_CURRENT_LUWID

Logical unit of work identifier for current batch.

MQCACH_LAST_LUWID

Logical unit of work identifier for last committed batch.

MQCACH_XMIT_Q_NAME

Transmission queue name.

MQIACH_CHANNEL_INSTANCE_TYPE

Channel instance type.

MQIACH_CHANNEL_TYPE

Channel type.

MQIACH_CURRENT_MSGS

Number of messages sent or received in current batch.

MQIACH_CURRENT_SEQ_NUMBER

Sequence number of last message sent or received.

MQIACH_INDOUBT_STATUS

Whether the channel is currently in-doubt.

MQIACH_LAST_SEQ_NUMBER

Sequence number of last message in last committed batch.

MQCACH_CURRENT_LUWID, MQCACH_LAST_LUWID, MQIACH_CURRENT_MSGS, MQIACH_CURRENT_SEQ_NUMBER, MQIACH_INDOUBT_STATUS and MQIACH_LAST_SEQ_NUMBER do not apply to server-connection channels, and no values are returned. If specified on the command, they are ignored.

Relevant for current-only status

The following information applies only to current channel instances. The information applies to all channel types, except where stated.

MQCA_Q_MGR_NAME

Name of the queue manager that owns the channel instance. This parameter is valid only on z/OS.

MQCA_REMOTE_Q_MGR_NAME

Queue manager name, or queue-sharing group name of the remote system. The remote queue manager name is always returned regardless of the instance attributes requested.

MQCACH_CHANNEL_START_DATE

Date channel was started.

MQCACH_CHANNEL_START_TIME

Time channel was started.

MQCACH_LAST_MSG_DATE

Date last message was sent, or MQI call was handled.

MQCACH_LAST_MSG_TIME

Time last message was sent, or MQI call was handled.

MQCACH_LOCAL_ADDRESS

Local communications address for the channel.

MQCACH_MCA_JOB_NAME

Name of MCA job.

This parameter is not valid on z/OS.

You cannot use MQCACH_MCA_JOB_NAME as a parameter to filter on.

MQCACH_MCA_USER_ID

The user ID used by the MCA.

MQCACH_REMOTE_APPL_TAG

Remote partner application name. This is the name of the client application at the remote end of the channel. This parameter applies only to server-connection channels.

The maximum length of the string is MQ_APPL_TAG_LENGTH.

MQCACH_SSL_SHORT_PEER_NAME

SSL short peer name.

MQCACH_SSL_CERT_ISSUER_NAME

The full Distinguished Name of the issuer of the remote certificate.

MQCACH_SSL_CERT_USER_ID

User ID associated with the remote certificate. This is valid on z/OS only.

MQIA_MONITORING_CHANNEL

Current level of monitoring data collection.

MQIACF_MONITORING

All channel status monitoring attributes. These are:

MQIA_MONITORING_CHANNEL

Current level of monitoring data collection.

MQIACH_BATCH_SIZE_INDICATOR

Batch size.

MQIACH_COMPRESSION_RATE

The compression rate achieved displayed to the nearest percentage.

MQIACH_COMPRESSION_TIME

The amount of time per message, displayed in microseconds, spent during compression or decompression.

MQIACH_EXIT_TIME_INDICATOR

Exit time.

MQIACH_NETWORK_TIME_INDICATOR

Network time.

MQIACH_XMITQ_MSGS_AVAILABLE

Number of messages available to the channel on the transmission queue.

MQIACH_XMITQ_TIME_INDICATOR

Time on transmission queue.

You cannot use MQIACF_MONITORING as a parameter to filter on.

MQIACH_BATCH_SIZE_INDICATOR

Batch size.

You cannot use MQIACH_BATCH_SIZE_INDICATOR as a parameter to filter on.

MQIACH_BATCHES

Number of completed batches.

MQIACH_BUFFERS_RCVD

Number of buffers received.

MQIACH_BUFFERS_SENT

Number of buffers sent.

MQIACH_BYTES_RCVD

Number of bytes received.

MQIACH_BYTES_SENT

Number of bytes sent.

MQIACH_CHANNEL_SUBSTATE

Current channel substate.

MQIACH_EXIT_TIME_INDICATOR

Exit time.

You cannot use MQIACH_EXIT_TIME_INDICATOR as a parameter to filter on.

MQIACH_COMPRESSION_RATE

The compression rate achieved displayed to the nearest percentage.

You cannot use MQIACH_COMPRESSION_RATE as a parameter to filter on.

MQIACH_COMPRESSION_TIME

The amount of time per message, displayed in microseconds, spent during compression or decompression.

You cannot use MQIACH_COMPRESSION_TIME as a parameter to filter on.

MQIACH_HDR_COMPRESSION

Technique used to compress the header data sent by the channel is compressed.

MQIACH_KEEP_ALIVE_INTERVAL

The KeepAlive interval in use for this session. This parameter is significant only for z/OS.

MQIACH_LONG_RETRIES_LEFT

Number of long retry attempts remaining.

MQIACH_MAX_MSG_LENGTH

Maximum message length. This is valid only on z/OS.

MQIACH_MCA_STATUS

MCA status.

You cannot use MQIACH_MCA_STATUS as a parameter to filter on.

MQIACH_MSG_COMPRESSION

Technique used to compress the message data sent by the channel.

MQIACH_MSGS

Number of messages sent or received, or number of MQI calls handled.

MQIACH_NETWORK_TIME_INDICATOR

Network time.

You cannot use MQIACH_NETWORK_TIME_INDICATOR as a parameter on which to filter.

MQIACH_SHORT_RETRIES_LEFT

Number of short retry attempts remaining.

MQIACH_SSL_KEY_RESETS

Number of successful SSL key resets.

MQIACH_SSL_RESET_DATE

Date of previous successful SSL secret key reset.

MQIACH_SSL_RESET_TIME

Time of previous successful SSL secret key reset.

MQIACH_STOP_REQUESTED

Whether user stop request has been received.

MQIACH_XMITQ_MSGS_AVAILABLE

Number of messages available to the channel on the transmission queue.

MQIACH_XMITQ_TIME_INDICATOR

Time on transmission queue.

You cannot use MQIACH_XMITQ_TIME_INDICATOR as a parameter to filter on.

The following is supported on HP OpenVMS, i5/OS, Compaq NonStop Kernel, UNIX systems, Windows, and z/OS:

MQIACH_BATCH_SIZE

Batch size.

The following is supported on HP OpenVMS, Compaq NonStop Kernel, i5/OS, UNIX systems, Windows and z/OS:

MQIACH_HB_INTERVAL

Heartbeat interval (seconds).

MQIACH_NPM_SPEED

Speed of nonpersistent messages.

The following attributes do not apply to server-connection channels, and no values are returned. If specified on the command they are ignored:

- MQIACH_BATCH_SIZE_INDICATOR
- MQIACH_BATCH_SIZE
- MQIACH_BATCHES
- MQIACH_LONG_RETRIES_LEFT
- MQIACH_NETWORK_TIME
- MQIACH_NPM_SPEED
- MQCA_REMOTE_Q_MGR_NAME
- MQIACH_SHORT_RETRIES_LEFT
- MQIACH_XMITQ_MSGS_AVAILABLE
- MQIACH_XMITQ_TIME_INDICATOR

Relevant for short status

The following parameter applies to current channels on z/OS:

MQCACH_Q_MGR_NAME

Name of the queue manager that owns the channel instance.

ChannelInstanceType (MQCFIN)

Channel instance type (parameter identifier:
MQIACH_CHANNEL_INSTANCE_TYPE).

It is always returned regardless of the channel instance attributes requested.

The value can be:

MQOT_CURRENT_CHANNEL

Current channel status.

This is the default, and indicates that only current status information for active channels is to be returned.

Both common status information and active-only status information can be requested for current channels.

MQOT_SAVED_CHANNEL

Saved channel status.

Specify this to cause saved status information for both active and inactive channels to be returned.

Only common status information can be returned. Active-only status information is not returned for active channels if this keyword is specified.

MQOT_SHORT_CHANNEL

Short channel status (valid on z/OS only).

Specify this to cause short status information for current channels to be returned.

Other common status and current-only status information is not returned for current channels if this keyword is specified.

You cannot use MQIACH_CHANNEL_INSTANCE_TYPE as a parameter to filter on.

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

You cannot use *CommandScope* as a parameter to filter on.

ConnectionName (MQCFST)

Connection name (parameter identifier: MQCACH_CONNECTION_NAME).

If this parameter is present, eligible channel instances are limited to those using this connection name. If it is not specified, eligible channel instances are not limited in this way.

The connection name is always returned, regardless of the instance attributes requested.

The value returned for *ConnectionName* might not be the same as in the channel definition, and might differ between the current channel status and the saved channel status. (Using *ConnectionName* for limiting the number of sets of status is therefore not recommended.)

For example, when using TCP, if *ConnectionName* in the channel definition :

- Is blank or is in “host name” format, the channel status value has the resolved IP address.
- Includes the port number, the current channel status value includes the port number (except on z/OS), but the saved channel status value does not.

The maximum length of the string is MQ_CONN_NAME_LENGTH.

IntegerFilterCommand (**MQCFIF**)

Integer filter command descriptor. The parameter identifier must be any integer type parameter allowed in *ChannelInstanceAttrs* except MQIACF_ALL and others as noted. Use this to restrict the output from the command by specifying a filter condition. See “MQCFIF - PCF integer filter parameter” on page 427 for information about using this filter condition.

If you specify an integer filter, you cannot also specify a string filter using the *StringFilterCommand* parameter.

StringFilterCommand (**MQCFSF**)

String filter command descriptor. The parameter identifier must be any string type parameter allowed in *ChannelInstanceAttrs* except MQCACH_CHANNEL_NAME and others as noted. Use this to restrict the output from the command by specifying a filter condition. See “MQCFSF - PCF string filter parameter” on page 434 for information about using this filter condition.

If you specify a string filter for *ConnectionName* or *XmitQName*, you cannot also specify the *ConnectionName* or *XmitQName* parameter.

If you specify a string filter, you cannot also specify an integer filter using the *IntegerFilterCommand* parameter.

XmitQName (**MQCFST**)

Transmission queue name (parameter identifier: MQCACH_XMIT_Q_NAME).

If this parameter is present, eligible channel instances are limited to those using this transmission queue. If it is not specified, eligible channel instances are not limited in this way.

The transmission queue name is always returned, regardless of the instance attributes requested.

The maximum length of the string is MQ_Q_NAME_LENGTH.

Error codes

This command might return the following in the response format header, in addition to the values shown on page “Error codes applicable to all commands” on page 27.

Reason (**MQLONG**)

The value can be:

MQRCCF_CHANNEL_NAME_ERROR

Channel name error.

MQRCCF_CHANNEL_NOT_FOUND

Channel not found.

MQRCCF_CHL_INST_TYPE_ERROR

Channel instance type not valid.

MQRCCF_CHL_STATUS_NOT_FOUND

Channel status not found.

MQRCCF_XMIT_Q_NAME_ERROR

Transmission queue name error.

Inquire Channel Status (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The response to the Inquire Channel Status (MQCMD_INQUIRE_CHANNEL_STATUS) command consists of the response header followed by

- The *ChannelName* structure,
- The *ChannelDisposition* structure (on z/OS only),
- The *ChannelInstanceType* structure
- The *ChannelStatus* structure (except on z/OS channels whose *ChannelInstanceType* parameter has a value of MQOT_SAVED_CHANNEL).
- The *ChannelType* structure
- The *ConnectionName* structure
- The *RemoteApplTag* structure
- The *RemoteQMgrName* structure
- The *StopRequested* structure
- The *XmitQName* structure

which are followed by the requested combination of status attribute parameter structures. One such message is generated for each channel instance found that matches the criteria specified on the command.

On z/OS, if the value for any of these parameters exceeds 999 999 999, it is returned as 999 999 999:

- *Batches*
- *BuffersReceived*
- *BuffersSent*
- *BytesReceived*
- *BytesSent*
- *CompressionTime*
- *CurrentMsgs*
- *ExitTime*
- *Msgs*
- *NetTime*
- *SSLKeyResets*
- *XQTime*

Always returned:

ChannelDisposition, *ChannelInstanceType*, *ChannelName*, *ChannelStatus*,
ChannelType, *ConnectionName*, *RemoteApplTag*, *RemoteQMgrName*,
StopRequested, *SubState*, *XmitQName*

Returned if requested:

Batches, *BatchSize*, *BatchSizeIndicator*, *BuffersReceived*, *BuffersSent*,
BytesReceived, *BytesSent*, *ChannelMonitoring*, *ChannelStartDate*,
ChannelStartTime, *CompressionRate*, *CompressionTime*, *CurrentLUWID*,

CurrentMsgs, *CurrentSequenceNumber*, *ExitTime*, *HeaderCompression*,
HeartbeatInterval, *InDoubtStatus*, *KeepAliveInterval*, *LastLUWID*,
LastMsgDate, *LastMsgTime*, *LastSequenceNumber*, *LocalAddress*,
LongRetriesLeft, *MaxMsgLength*, *MCAJobName*, *MCAStatus*, *MCAUserIdentifier*,
MessageCompression, *Msgs*, *MsgsAvailable*, *NetTime*, *NonPersistentMsgSpeed*,
QMgrName, , *ShortRetriesLeft*, *SSLCertRemoteIssuerName*, *SSLCertUserId*,
SSLKeyResetDate, *SSLKeyResets*, *SSLKeyResetTime*, *SSLShortPeerName*,
XQTime

Response data

Batches (MQCFIN)

Number of completed batches (parameter identifier: MQIACH_BATCHES).

BatchSize (MQCFIN)

Negotiated batch size (parameter identifier: MQIACH_BATCH_SIZE).

BatchSizeIndicator (MQCFIL)

Indicator of the number of messages in a batch (parameter identifier: MQIACH_BATCH_SIZE_INDICATOR). Two values are returned:

- A value based on recent activity over a short period of time.
- A value based on activity over a longer period of time.

Where no measurement is available, the value MQMON_NOT_AVAILABLE is returned.

BuffersReceived (MQCFIN)

Number of buffers received (parameter identifier: MQIACH_BUFFERS_RCVD).

BuffersSent (MQCFIN)

Number of buffers sent (parameter identifier: MQIACH_BUFFERS_SENT).

BytesReceived (MQCFIN)

Number of bytes received (parameter identifier: MQIACH_BYTES_RCVD).

BytesSent (MQCFIN)

Number of bytes sent (parameter identifier: MQIACH_BYTES_SENT).

ChannelDisposition (MQCFIN)

Channel disposition (parameter identifier: MQIACH_CHANNEL_DISP). This parameter is valid only on z/OS.

The value can be:

MQCHLD_ALL

Status information for private channels.

In a shared queue environment where the command is being executed on the queue manager where it was issued, or if *ChannelInstanceType* has a value of MQOT_CURRENT_CHANNEL, this option also displays the requested status information for shared channels.

MQCHLD_PRIVATE

Status information for private channels.

MQCHLD_SHARED

Status information for shared channels.

ChannelInstanceType (MQCFIN)

Channel instance type (parameter identifier: MQIACH_CHANNEL_INSTANCE_TYPE).

The value can be:

MQOT_CURRENT_CHANNEL

Current channel status.

MQOT_SAVED_CHANNEL

Saved channel status.

MQOT_SHORT_CHANNEL

Short channel status, only on z/OS.

ChannelMonitoring (MQCFIN)

Current level of monitoring data collection for the channel (parameter identifier: MQIACH_MONITORING_CHANNEL).

The value can be:

MQMON_OFF

Monitoring for the channel is switched off.

MQMON_LOW

Low rate of data collection.

MQMON_MEDIUM

Medium rate of data collection.

MQMON_HIGH

High rate of data collection.

ChannelName (MQCFST)

Channel name (parameter identifier: MQCACH_CHANNEL_NAME).

The maximum length of the string is MQ_CHANNEL_NAME_LENGTH.

ChannelStartDate (MQCFST)

Date channel started, in the form yyyy-mm-dd (parameter identifier: MQCACH_CHANNEL_START_DATE).

The maximum length of the string is MQ_CHANNEL_DATE_LENGTH.

ChannelStartTime (MQCFST)

Time channel started, in the form hh.mm.ss (parameter identifier: MQCACH_CHANNEL_START_TIME).

The maximum length of the string is MQ_CHANNEL_TIME_LENGTH.

ChannelStatus (MQCFIN)

Channel status (parameter identifier: MQIACH_CHANNEL_STATUS).

The value can be:

MQCHS_BINDING

Channel is negotiating with the partner.

MQCHS_STARTING

Channel is waiting to become active.

MQCHS_RUNNING

Channel is transferring or waiting for messages.

MQCHS_PAUSED

Channel is paused.

MQCHS_STOPPING

Channel is in process of stopping.

MQCHS_RETRYING

Channel is reattempting to establish connection.

MQCHS_STOPPED

Channel is stopped.

MQCHS_REQUESTING

Requester channel is requesting connection.

MQCHS_INITIALIZING

Channel is initializing.

ChannelType (MQCFIN)

Channel type (parameter identifier: MQIACH_CHANNEL_TYPE).

The value can be:

MQCHT_SENDER

Sender.

MQCHT_SERVER

Server.

MQCHT_RECEIVER

Receiver.

MQCHT_REQUESTER

Requester.

MQCHT_SVRCONN

Server-connection (for use by clients).

MQCHT_CLNTCONN

Client connection.

MQCHT_CLUSRCVR

Cluster-receiver.

MQCHTCLUSSDR

Cluster-sender.

CompressionRate (MQCFIL)

The compression rate achieved displayed to the nearest percentage (parameter identifier: MQIACH_COMPRESSION_RATE). Two values are returned:

- A value based on recent activity over a short period of time.
- A value based on activity over a longer period of time.

Where no measurement is available, the value MQMON_NOT_AVAILABLE is returned.

CompressionTime (MQCFIL)

The amount of time per message, displayed in microseconds, spent during compression or decompression (parameter identifier: MQIACH_COMPRESSION_TIME). Two values are returned:

MQCACH_COMPRESSION_TIME).

- A value based on recent activity over a short period of time.
- A value based on activity over a longer period of time.

Where no measurement is available, the value MQMON_NOT_AVAILABLE is returned.

ConnectionName (MQCFST)

Connection name (parameter identifier: MQCACH_CONNECTION_NAME).

The maximum length of the string is MQ_CONN_NAME_LENGTH.

CurrentLUWID (MQCFST)

Logical unit of work identifier for in-doubt batch (parameter identifier: MQCACH_CURRENT_LUWID).

The logical unit of work identifier associated with the current batch, for a sending or a receiving channel.

For a sending channel, when the channel is in-doubt it is the LUWID of the in-doubt batch.

It is updated with the LUWID of the next batch when this is known.

The maximum length is MQ_LUWID_LENGTH.

CurrentMsgs (MQCFIN)

Number of messages in-doubt (parameter identifier: MQIACH_CURRENT_MSGS).

For a sending channel, this is the number of messages that have been sent in the current batch. It is incremented as each message is sent, and when the channel becomes in-doubt it is the number of messages that are in-doubt.

For a receiving channel, it is the number of messages that have been received in the current batch. It is incremented as each message is received.

The value is reset to zero, for both sending and receiving channels, when the batch is committed.

CurrentSequenceNumber (MQCFIN)

Sequence number of last message in in-doubt batch (parameter identifier: MQIACH_CURRENT_SEQ_NUMBER).

For a sending channel, this is the message sequence number of the last message sent. It is updated as each message is sent, and when the channel becomes in-doubt it is the message sequence number of the last message in the in-doubt batch.

For a receiving channel, it is the message sequence number of the last message that was received. It is updated as each message is received.

ExitTime (MQCFIL)

Indicator of the time taken executing user exits per message (parameter identifier: MQIACH_EXIT_TIME_INDICATOR). Amount of time, in microseconds, spent processing user exits per message. Where more than one exit is executed per message, the value is the sum of all the user exit times for a single message. Two values are returned:

- A value based on recent activity over a short period of time.
- A value based on activity over a longer period of time.

Where no measurement is available, the value MQMON_NOT_AVAILABLE is returned.

HeaderCompression (MQCFIL)

Whether the header data sent by the channel is compressed (parameter identifier: MQIACH_HDR_COMPRESSION). Two values are returned:

- The default header data compression value negotiated for this channel.
- The header data compression value used for the last message sent. The header data compression value can be altered in a sending channels message exit. If no message has been sent, the second value is MQCOMPRESS_NOT_AVAILABLE.

The values can be:

MQCOMPRESS_NONE

No header data compression is performed. This is the default value.

MQCOMPRESS_SYSTEM

Header data compression is performed.

MQCOMPRESS_NOT_AVAILABLE

No message has been sent by the channel.

HeartbeatInterval (MQCFIN)

Heartbeat interval (parameter identifier: MQIACH_HB_INTERVAL).

InDoubtStatus (MQCFIN)

Whether the channel is currently in doubt (parameter identifier: MQIACH_INDOUBT_STATUS).

A sending channel is only in doubt while the sending Message Channel Agent is waiting for an acknowledgment that a batch of messages, which it has sent, has been successfully received. It is not in doubt at all other times, including the period during which messages are being sent, but before an acknowledgment has been requested.

A receiving channel is never in doubt.

The value can be:

MQCHIDS_NOT_INDOUBT

Channel is not in-doubt.

MQCHIDS_INDOUBT

Channel is in-doubt.

KeepAliveInterval (MQCFIN)

KeepAlive interval (parameter identifier: MQIACH_KEEP_ALIVE_INTERVAL).
This parameter is valid only on z/OS.

LastLUWID (MQCFST)

Logical unit of work identifier for last committed batch (parameter identifier: MQCACH_LAST_LUWID).

The maximum length is MQ_LUWID_LENGTH.

LastMsgDate (MQCFST)

Date last message was sent, or MQI call was handled, in the form yyyy-mm-dd
(parameter identifier: MQCACH_LAST_MSG_DATE).

The maximum length of the string is MQ_CHANNEL_DATE_LENGTH.

LastMsgTime (MQCFST)

Time last message was sent, or MQI call was handled, in the form hh.mm.ss
(parameter identifier: MQCACH_LAST_MSG_TIME).

The maximum length of the string is MQ_CHANNEL_TIME_LENGTH.

LastSequenceNumber (MQCFIN)

Sequence number of last message in last committed batch (parameter identifier: MQIACH_LAST_SEQ_NUMBER).

LocalAddress (MQCFST)

Local communications address for the channel (parameter identifier: MQCACH_LOCAL_ADDRESS).

The maximum length of the string is MQ_LOCAL_ADDRESS_LENGTH.

LongRetriesLeft (**MQCFIN**)

Number of long retry attempts remaining (parameter identifier: MQIACH_LONG_RETRIES_LEFT).

MaxMsgLength (**MQCFIN**)

Maximum message length (parameter identifier: MQIACH_MAX_MSG_LENGTH). This parameter is valid only on z/OS.

MCAJobName (**MQCFST**)

Name of MCA job (parameter identifier: MQCACH_MCA_JOB_NAME).

The maximum length of the string is MQ_MCA_JOB_NAME_LENGTH.

MCAStatus (**MQCFIN**)

MCA status (parameter identifier: MQIACH_MCA_STATUS).

The value can be:

MQMCAS_STOPPED

Message channel agent stopped.

MQMCAS_RUNNING

Message channel agent running.

MCAUserIdentifer (**MQCFST**)

The user ID used by the MCA (parameter identifier: MQCACH_MCA_USER_ID).

This parameter applies only to server-connection, receiver, requester, and cluster-receiver channels.

The maximum length of the string is MQ_MCA_USER_ID_LENGTH.

MessageCompression (**MQCFIL**)

Whether the header data sent by the channel is compressed (parameter identifier: MQIACH_MSG_COMPRESSION). Two values are returned:

- The default message data compression value negotiated for this channel.
- The message data compression value used for the last message sent. The message data compression value can be altered in a sending channels message exit. If no message has been sent, the second value is MQCOMPRESS_NOT_AVAILABLE.

The values can be:

MQCOMPRESS_NONE

No message data compression is performed. This is the default value.

MQCOMPRESS_RLE

Message data compression is performed using run-length encoding.

MQCOMPRESS_ZLIBFAST

Message data compression is performed using ZLIB encoding with speed prioritized.

MQCOMPRESS_ZLIBHIGH

Message data compression is performed using ZLIB encoding with compression prioritized.

MQCOMPRESS_NOT_AVAILABLE

No message has been sent by the channel.

Msgs (**MQCFIN**)

Number of messages sent or received, or number of MQI calls handled (parameter identifier: MQIACH_MSGS).

MsgsAvailable (MQCFIN)

Number of messages available (parameter identifier: MQIACH_XMITQ_MSGS_AVAILABLE). Number of messages queued on the transmission queue available to the channel for MQGETs.

Where no measurement is available, the value MQMON_NOT_AVAILABLE is returned.

NetTime (MQCFIL)

Indicator of the time of a network operation (parameter identifier: MQIACH_NETWORK_TIME_INDICATOR). Amount of time, in microseconds, to send a request to the remote end of the channel and receive a response. Two values are returned:

- A value based on recent activity over a short period of time.
- A value based on activity over a longer period of time.

Where no measurement is available, the value MQMON_NOT_AVAILABLE is returned.

NonPersistentMsgSpeed (MQCFIN)

Speed at which nonpersistent messages are to be sent (parameter identifier: MQIACH_NPM_SPEED).

The value can be:

MQNPMS_NORMAL

Normal speed.

MQNPMS_FAST

Fast speed.

QMgrName (MQCFST)

Name of the queue manager that owns the channel instance (parameter identifier: MQCA_Q_MGR_NAME). This parameter is valid only on z/OS.

The maximum length of the string is MQ_Q_MGR_NAME_LENGTH.

RemoteApplTag (MQCFST)

Name of the remote queue manager, or queue-sharing group (parameter identifier: MQCACH_REMOTE_APPL_TAG).

The remote partner application name. This is the name of the client application at the remote end of the channel. This parameter applies only to server-connection channels.

RemoteQMgrName (MQCFST)

Name of the remote queue manager, or queue-sharing group (parameter identifier: MQCA_REMOTE_Q_MGR_NAME).

ShortRetriesLeft (MQCFIN)

Number of short retry attempts remaining (parameter identifier: MQIACH_SHORT_RETRIES_LEFT).

SSLCertRemoteIssuerName (MQCFST)

The full Distinguished Name of the issuer of the remote certificate. The issuer is the Certificate Authority that issued the certificate (parameter identifier: MQCACH_SSL_CERT_ISSUER_NAME).

The maximum length of the string is MQ_SHORT_DNAME_LENGTH.

SSLCertUserId (MQCFST)

The local user ID associated with the remote certificate (parameter identifier: MQCACH_SSL_CERT_USER_ID).

This parameter is valid only on z/OS.

The maximum length of the string is MQ_USER_ID_LENGTH.

***SSLKeyResetDate* (MQCFST)**

Date of the previous successful SSL secret key reset, in the form yyyy-mm-dd (parameter identifier: MQCACH_SSL_KEY_RESET_DATE).

The maximum length of the string is MQ_DATE_LENGTH.

***SSLKeyResets* (MQCFIN)**

SSL secret key resets (parameter identifier: MQIACH_SSL_KEY_RESETS).

The number of successful SSL secret key resets that have occurred for this channel instance since the channel started. If SSL secret key negotiation is enabled, the count is incremented whenever a secret key reset is performed.

***SSLKeyResetTime* (MQCFST)**

Time of the previous successful SSL secret key reset, in the form hh.mm.ss (parameter identifier: MQCACH_SSL_KEY_RESET_TIME).

The maximum length of the string is MQ_TIME_LENGTH.

***SSLShortPeerName* (MQCFST)**

Distinguished Name of the peer queue manager or client at the other end of the channel (parameter identifier: MQCACH_SSL_SHORT_PEER_NAME).

The maximum length is MQ_SHORT_DNAME_LENGTH. This limit might mean that exceptionally long Distinguished Names are truncated.

***StopRequested* (MQCFIN)**

Whether user stop request is outstanding (parameter identifier: MQIACH_STOP_REQUESTED).

The value can be:

MQCHSR_STOP_NOT_REQUESTED

User stop request has not been received.

MQCHSR_STOP_REQUESTED

User stop request has been received.

***SubState* (MQCFIN)**

Current action being performed by the channel (parameter identifier: MQIACH_CHANNEL_SUBSTATE).

The value can be:

MQCHSSTATE_CHADEXIT

Running channel auto-definition exit.

MQCHSSTATE_COMPRESSING

Compressing or decompressing data.

MQCHSSTATE_END_OF_BATCH

End of batch processing.

MQCHSSTATE_HANDSHAKING

SSL handshaking.

MQCHSSTATE_HEARTBEATING

Heartbeating with partner.

MQCHSSTATE_IN_MQGET

Performing MQGET.

MQCHSSTATE_IN_MQI_CALL

Executing an MQ API call, other than an MQPUT or MQGET.

MQCHSSTATE_IN_MQPUT

Performing MQPUT.

MQCHSSTATE_MREXIT

Running retry exit.

MQCHSSTATE_MSGEXIT

Running message exit.

MQCHSSTATE_NAME_SERVER

Nameserver request.

MQCHSSTATE_NET_CONNECTING

Network connect.

MQCHSSTATE_OTHER

Undefined state.

MQCHSSTATE_RCVEXIT

Running receive exit.

MQCHSSTATE_RECEIVING

Network receive.

MQCHSSTATE_RESYNCHING

Resynching with partner.

MQCHSSTATE_SCYEXIT

Running security exit.

MQCHSSTATE_SENDEXIT

Running send exit.

MQCHSSTATE_SENDING

Network send.

MQCHSSTATE_SERIALIZING

Serialized on queue manager access.

XmitQName (MQCFST)

Transmission queue name (parameter identifier: MQCACH_XMIT_Q_NAME).

The maximum length of the string is MQ_Q_NAME_LENGTH.

XQTime (MQCFIL)

Transmission queue time indicator (parameter identifier:

MQIACH_XMITQ_TIME_INDICATOR). The time, in microseconds, that messages remained on the transmission queue before being retrieved. The time is measured from when the message is put onto the transmission queue until it is retrieved to be sent on the channel and, therefore, includes any interval caused by a delay in the putting application.

Two values are returned:

- A value based on recent activity over a short period of time.
- A value based on activity over a longer period of time.

Where no measurement is available, the value MQMON_NOT_AVAILABLE is returned.

Inquire Cluster Queue Manager

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Inquire Cluster Queue Manager (MQCMD_INQUIRE_CLUSTER_Q_MGR) command inquires about the attributes of WebSphere MQ queue managers in a cluster.

Required parameters:

ClusterQMgrName

Optional parameters:

*Channel, ClusterName, ClusterQMgrAttrs, CommandScope,
IntegerFilterCommand, StringFilterCommand,*

Required parameters

ClusterQMgrName (MQCFST)

Queue manager name (parameter identifier:
MQCA_CLUSTER_Q_MGR_NAME).

Generic queue manager names are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all queue managers having names that start with the selected character string. An asterisk on its own matches all possible names.

The queue manager name is always returned, regardless of the attributes requested.

The maximum length of the string is MQ_Q_MGR_NAME_LENGTH.

Optional parameters

Channel (MQCFST)

Channel name (parameter identifier: MQCACH_CHANNEL_NAME).

Specifies that eligible cluster queue managers are limited to those having the specified channel name.

Generic channel names are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all channels having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length of the string is MQ_CHANNEL_NAME_LENGTH.

If you do not specify a value for this parameter, channel information about *all* queue managers in the cluster is returned.

ClusterName (MQCFST)

Cluster name (parameter identifier: MQCA_CLUSTER_NAME).

Specifies that eligible cluster queue managers are limited to those having the specified cluster name.

Generic cluster names are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all clusters having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length of the string is MQ_CLUSTER_NAME_LENGTH.

If you do not specify a value for this parameter, cluster information about *all* queue managers inquired is returned.

ClusterQMgrAttrs (MQCFIL)

Attributes (parameter identifier: MQIACF_CLUSTER_Q_MGR_ATTRS).

Some parameters are relevant only for cluster channels of a particular type or types. Attributes that are not relevant for a particular type of channel cause no output, and do not cause an error. To check which attributes apply to which channel types, refer to *WebSphere MQ Intercommunications*.

The attribute list might specify the following on its own (this is the default value used if the parameter is not specified):

MQIACF_ALL

All attributes.

or a combination of the following:

MQCA_ALTERATION_DATE

The date on which the information was last altered.

MQCA_ALTERATION_TIME

The time at which the information was last altered.

MQCA_CLUSTER_DATE

The date on which the information became available to the local queue manager.

MQCA_CLUSTER_NAME

The name of the cluster to which the channel belongs.

MQCA_CLUSTER_Q_MGR_NAME

The name of the cluster to which the channel belongs.

MQCA_CLUSTER_TIME

The time at which the information became available to the local queue manager.

MQCA_Q_MGR_IDENTIFIER

The unique identifier of the queue manager.

MQCACH_CONNECTION_NAME

Connection name.

MQCACH_DESCRIPTION

Description.

MQCACH_LOCAL_ADDRESS

Local communications address for the channel.

MQCACH_MCA_NAME

Message channel agent name.

You cannot use MQCACH_MCA_NAME as a parameter to filter on.

MQCACH_MCA_USER_ID

MCA user identifier.

MQCACH_MODE_NAME
Mode name.

MQCACH_MR_EXIT_NAME
Message-retry exit name.

MQCACH_MR_EXIT_USER_DATA
Message-retry exit user data.

MQCACH_MSG_EXIT_NAME
Message exit name.

MQCACH_MSG_EXIT_USER_DATA
Message exit user data.

MQCACH_PASSWORD
Password.

This parameter is not valid on z/OS.

MQCACH_RCV_EXIT_NAME
Receive exit name.

MQCACH_RCV_EXIT_USER_DATA
Receive exit user data.

MQCACH_SEC_EXIT_NAME
Security exit name.

MQCACH_SEC_EXIT_USER_DATA
Security exit user data.

MQCACH_SEND_EXIT_NAME
Send exit name.

MQCACH_SEND_EXIT_USER_DATA
Send exit user data.

MQCACH_SSL_CIPHER_SPEC
SSL cipher spec.

|
MQIACH_SSL_CLIENT_AUTH
SSL client authentication.

MQCACH_SSL_PEER_NAME
SSL peer name.

MQCACH_TP_NAME
Transaction program name.

MQCACH_USER_ID
User identifier.

This parameter is not valid on z/OS.

MQIA_MONITORING_CHANNEL
Online monitoring data collection.

MQIACF_Q_MGR_DEFINITION_TYPE
How the cluster queue manager was defined.

MQIACF_Q_MGR_TYPE
The function of the queue manager in the cluster.

MQIACF_SUSPEND
Whether the queue manager is suspended from the cluster.

MQIACH_BATCH_HB

The value being used for batch heartbeating.

MQIACH_BATCH_INTERVAL

Batch wait interval (seconds).

MQIACH_BATCH_SIZE

Batch size.

MQIACH_CHANNEL_STATUS

Channel status.

MQIACH_CLWL_CHANNEL_PRIORITY

Cluster workload channel priority.

MQIACH_CLWL_CHANNEL_RANK

Cluster workload channel rank.

MQIACH_CLWL_CHANNEL_WEIGHT

Cluster workload channel weight.

MQIACH_DATA_CONVERSION

Whether sender must convert application data.

MQIACH_DISC_INTERVAL

Disconnection interval.

MQIACH_HB_INTERVAL

Heartbeat interval (seconds).

MQIACH_HDR_COMPRESSION

The list of header data compression techniques supported by the channel.

MQIACH_KEEP_ALIVE_INTERVAL

KeepAlive interval (valid on z/OS only).

MQIACH_LONG_RETRY

Long retry count.

MQIACH_LONG_TIMER

Long timer.

MQIACH_MAX_MSG_LENGTH

Maximum message length.

MQIACH_MCA_TYPE

MCA type.

MQIACH_MR_COUNT

Message retry count.

MQIACH_MR_INTERVAL

Message retry interval (milliseconds).

MQIACH_MSG_COMPRESSION

List of message data compression techniques supported by the channel.

MQIACH_NETWORK_PRIORITY

Network priority.

MQIACH_NPM_SPEED

Speed of nonpersistent messages.

MQIACH_PUT_AUTHORITY

Put authority.

MQIACH_SEQUENCE_NUMBER_WRAP

Sequence number wrap.

MQIACH_SHORT_RETRY

Short retry count.

MQIACH_SHORT_TIMER

Short timer.

MQIACH_XMIT_PROTOCOL_TYPE

Transmission protocol type.

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

You cannot use *CommandScope* as a parameter to filter on.

IntegerFilterCommand (MQCFIF)

Integer filter command descriptor. The parameter identifier must be any integer type parameter allowed in *ClusterQMgrAttrs* except MQIACF_ALL and others as noted. Use this to restrict the output from the command by specifying a filter condition. See “MQCFIF - PCF integer filter parameter” on page 427 for information about using this filter condition.

If you specify an integer filter, you cannot also specify a string filter using the *StringFilterCommand* parameter.

StringFilterCommand (MQCFSF)

String filter command descriptor. The parameter identifier must be any string type parameter allowed in *ClusterQMgrAttrs* except MQCA_CLUSTER_Q_MGR_NAME and others as noted. Use this to restrict the output from the command by specifying a filter condition. See “MQCFSF - PCF string filter parameter” on page 434 for information about using this filter condition.

If you specify a string filter for *Channel* or *ClusterName*, you cannot also specify the *Channel* or *ClusterName* parameter.

If you specify a string filter, you cannot also specify an integer filter using the *IntegerFilterCommand* parameter.

Inquire Cluster Queue Manager (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The response to the Inquire Cluster Queue Manager (MQCMD_INQUIRE_CLUSTER_Q_MGR) command consists of the response header followed by the *QMgrName* structure and the requested combination of attribute parameter structures.

Always returned:

ChannelName, ClusterName, QMgrName,

Returned if requested:

*AlterationDate, AlterationTime, BatchHeartbeat, BatchInterval,
BatchSize, ChannelDesc, ChannelMonitoring, ChannelStatus, ClusterDate,
ClusterInfo, ClusterTime, CLWLChannelPriority, CLWLChannelRank,
CLWLChannelWeight, ConnectionName, DataConversion, DiscInterval,
HeaderCompression, HeartbeatInterval, KeepAliveInterval, LocalAddress,
LongRetryCount, LongRetryInterval, MaxMsgLength, MCAName, MCAType,
MCAClassIdentifier, MessageCompression, ModeName, MsgExit, MsgRetryCount,
MsgRetryExit, MsgRetryInterval, MsgRetryUserData, MsgUserData,
NetworkPriority, NonPersistentMsgSpeed, Password, PutAuthority,
QMgrDefinitionType, QMgrIdentifier, QMgrType, ReceiveExit,
ReceiveUserData, SecurityExit, SecurityUserData, SendExit, SendUserData,
SeqNumberWrap, ShortRetryCount, ShortRetryInterval, SSLCipherSpec,
SSLClientAuth, SSLPeerName, Suspend, TpName, TransportType,
UserIdentifier*

Response data

AlterationDate (MQCFST)

Alteration date, in the form yyyy-mm-dd (parameter identifier: MQCA_ALTERATION_DATE).

The date at which the information was last altered.

AlterationTime (MQCFST)

Alteration time, in the form hh.mm.ss (parameter identifier: MQCA_ALTERATION_TIME).

The time at which the information was last altered.

BatchHeartbeat (MQCFIN)

The value being used for batch heartbeating (parameter identifier: MQIACH_BATCH_HB).

The value can be between 0 and 999 999. A value of 0 indicates that batch heartbeating is not being used.

BatchInterval (MQCFIN)

Batch interval (parameter identifier: MQIACH_BATCH_INTERVAL).

BatchSize (MQCFIN)

Batch size (parameter identifier: MQIACH_BATCH_SIZE).

ChannelDesc (MQCFST)

Channel description (parameter identifier: MQCACH_DESC).

The maximum length of the string is MQ_CHANNEL_DESC_LENGTH.

ChannelMonitoring (MQCFIN)

Online monitoring data collection (parameter identifier: MQIA_MONITORING_CHANNEL).

The value can be:

MQMON_OFF

Online monitoring data collection is turned off for this channel.

MQMON_Q_MGR

The value of the queue manager's *ChannelMonitoring* parameter is inherited by the channel. This is the default value.

MQMON_LOW

Online monitoring data collection is turned on, with a low rate of data collection, for this channel unless the queue manager's *ChannelMonitoring* parameter is MQMON_NONE.

MQMON_MEDIUM

Online monitoring data collection is turned on, with a moderate rate of data collection, for this channel unless the queue manager's *ChannelMonitoring* parameter is MQMON_NONE.

MQMON_HIGH

Online monitoring data collection is turned on, with a high rate of data collection, for this channel unless the queue manager's *ChannelMonitoring* parameter is MQMON_NONE.

ChannelName (MQCFST)

Channel name (parameter identifier: MQCACH_CHANNEL_NAME).

The maximum length of the string is MQ_CHANNEL_NAME_LENGTH.

ChannelStatus (MQCFIN)

Channel status (parameter identifier: MQIACH_CHANNEL_STATUS).

The value can be:

MQCHS_BINDING

Channel is negotiating with the partner.

MQCHS_INACTIVE

Channel is not active.

MQCHS_STARTING

Channel is waiting to become active.

MQCHS_RUNNING

Channel is transferring or waiting for messages.

MQCHS_PAUSED

Channel is paused.

MQCHS_STOPPING

Channel is in process of stopping.

MQCHS_RETRYING

Channel is reattempting to establish connection.

MQCHS_STOPPED

Channel is stopped.

MQCHS_REQUESTING

Requester channel is requesting connection.

MQCHS_INITIALIZING

Channel is initializing.

This parameter is returned if the channel is a cluster-sender channel (CLUSSDR) only.

ClusterDate (MQCFST)

Cluster date, in the form yyyy-mm-dd (parameter identifier: MQCA_CLUSTER_DATE).

The date at which the information became available to the local queue manager.

ClusterInfo (MQCFIN)

Cluster information (parameter identifier: MQIACF_CLUSTER_INFO).

The cluster information available to the local queue manager.

ClusterName (MQCFST)

Cluster name (parameter identifier: MQCA_CLUSTER_NAME).

ClusterTime (MQCFST)

Cluster time, in the form hh.mm.ss (parameter identifier: MQCA_CLUSTER_TIME).

The time at which the information became available to the local queue manager.

CLWLChannelPriority (MQCFIN)

Channel priority (parameter identifier: MQIACH_CLWL_CHANNEL_PRIORITY).

CLWLChannelRank (MQCFIN)

Channel rank (parameter identifier: MQIACH_CLWL_CHANNEL_RANK).

CLWLChannelWeight (MQCFIN)

Channel weighting (parameter identifier: MQIACH_CLWL_CHANNEL_WEIGHT).

ConnectionName (MQCFST)

Connection name (parameter identifier: MQCACH_CONNECTION_NAME).

The maximum length of the string is MQ_CONN_NAME_LENGTH. On z/OS, it is MQ_LOCAL_ADDRESS_LENGTH.

DataConversion (MQCFIN)

Whether sender must convert application data (parameter identifier: MQIACH_DATA_CONVERSION).

The value can be:

MQCDC_NO_SENDER_CONVERSION

No conversion by sender.

MQCDC_SENDER_CONVERSION

Conversion by sender.

DiscInterval (MQCFIN)

Disconnection interval (parameter identifier: MQIACH_DISC_INTERVAL).

HeaderCompression (MQCFIL)

Header data compression techniques supported by the channel (parameter identifier: MQIACH_HDR_COMPRESSION). The values specified are in order of preference.

The value can be one, or more, of

MQCOMPRESS_NONE

No header data compression is performed.

MQCOMPRESS_SYSTEM

Header data compression is performed.

HeartbeatInterval (MQCFIN)

Heartbeat interval (parameter identifier: MQIACH_HB_INTERVAL).

KeepAliveInterval (MQCFIN)

KeepAlive interval (parameter identifier: MQIACH_KEEP_ALIVE_INTERVAL).

This parameter applies to z/OS only.

LocalAddress (MQCFST)

Local communications address for the channel (parameter identifier: MQCACH_LOCAL_ADDRESS).

The maximum length of the string is MQ_LOCAL_ADDRESS_LENGTH.

LongRetryCount (MQCFIN)

Long retry count (parameter identifier: MQIACH_LONG_RETRY).

LongRetryInterval (MQCFIN)

Long timer (parameter identifier: MQIACH_LONG_TIMER).

MaxMsgLength (MQCFIN)

Maximum message length (parameter identifier: MQIACH_MAX_MSG_LENGTH).

MCAName (MQCFST)

Message channel agent name (parameter identifier: MQCACH_MCA_NAME).

The maximum length of the string is MQ_MCA_NAME_LENGTH.

MCAType (MQCFIN)

Message channel agent type (parameter identifier: MQIACH_MCA_TYPE).

The value can be:

MQMCAT_PROCESS

Process.

MQMCAT_THREAD

Thread (Windows only).

MCAUserIdentifier (MQCFST)

Message channel agent user identifier (parameter identifier: MQCACH_MCA_USER_ID).

The maximum length of the string is MQ_USER_ID_LENGTH.

MessageCompression (MQCFIL)

Message data compression techniques supported by the channel (parameter identifier: MQIACH_MSG_COMPRESSION). The values specified are in order of preference.

The value can be one, or more, of:

MQCOMPRESS_NONE

No message data compression is performed.

MQCOMPRESS_RLE

Message data compression is performed using run-length encoding.

MQCOMPRESS_ZLIBFAST

Message data compression is performed using ZLIB encoding with speed prioritized.

MQCOMPRESS_ZLIBHIGH

Message data compression is performed using ZLIB encoding with compression prioritized.

***ModeName* (MQCFST)**

Mode name (parameter identifier: MQCACH_MODE_NAME).

The maximum length of the string is MQ_MODE_NAME_LENGTH.

***MsgExit* (MQCFST)**

Message exit name (parameter identifier: MQCACH_MSG_EXIT_NAME).

The maximum length of the string is MQ_EXIT_NAME_LENGTH.

In the following environments, if more than one message exit has been defined for the channel, the list of names is returned in an MQCFSL structure instead of an MQCFST structure: AIX, HP-UX, i5/OS, Solaris, Linux, and Windows. An MQCFSL structure is always used on z/OS.

***MsgRetryCount* (MQCFIN)**

Message retry count (parameter identifier: MQIACH_MR_COUNT).

***MsgRetryExit* (MQCFST)**

Message retry exit name (parameter identifier: MQCACH_MR_EXIT_NAME).

The maximum length of the string is MQ_EXIT_NAME_LENGTH.

***MsgRetryInterval* (MQCFIN)**

Message retry interval (parameter identifier: MQIACH_MR_INTERVAL).

***MsgRetryUserData* (MQCFST)**

Message retry exit user data (parameter identifier:
MQCACH_MR_EXIT_USER_DATA).

The maximum length of the string is MQ_EXIT_DATA_LENGTH.

***MsgUserData* (MQCFST)**

Message exit user data (parameter identifier:
MQCACH_MSG_EXIT_USER_DATA).

The maximum length of the string is MQ_EXIT_DATA_LENGTH.

In the following environments, if more than one message exit user data string has been defined for the channel, the list of strings is returned in an MQCFSL structure instead of an MQCFST structure: AIX, HP-UX, i5/OS, Solaris, Linux, and Windows. An MQCFSL structure is always used on z/OS.

***NetworkPriority* (MQCFIN)**

Network priority (parameter identifier: MQIACH_NETWORK_PRIORITY).

***NonPersistentMsgSpeed* (MQCFIN)**

Speed at which non-persistent messages are to be sent (parameter identifier:
MQIACH_NPM_SPEED).

The value can be:

MQNPMS_NORMAL

Normal speed.

MQNPMS_FAST

Fast speed.

Password (**MQCFST**)

Password (parameter identifier: MQCACH_PASSWORD). This parameter is not available on z/OS.

If a nonblank password is defined, it is returned as asterisks. Otherwise, it is returned as blanks.

The maximum length of the string is MQ_PASSWORD_LENGTH. However, only the first 10 characters are used.

PutAuthority (**MQCFIN**)

Put authority (parameter identifier: MQIACH_PUT_AUTHORITY).

The value can be:

MQPA_DEFAULT

Default user identifier is used.

MQPA_CONTEXT

Context user identifier is used.

MQPA_ALTERNATE_OR_MCA

The user identifier from the *UserIdentifier* field of the message descriptor is used. Any user ID received from the network is not used. This value is valid only on z/OS.

MQPA_ONLY_MCA

The default user identifier is used. Any user ID received from the network is not used. This value is valid only on z/OS.

QMgrDefinitionType (**MQCFIN**)

Queue manager definition type (parameter identifier: MQIACF_Q_MGR_DEFINITION_TYPE).

The value can be:

MQQMDS_EXPLICIT_CLUSTER_SENDER

A cluster-sender channel from an explicit definition.

MQQMDS_AUTO_CLUSTER_SENDER

A cluster-sender channel by auto-definition.

MQQMDS_CLUSTER_RECEIVER

A cluster-receiver channel.

MQQMDS_AUTO_EXP_CLUSTER_SENDER

A cluster-sender channel, both from an explicit definition and by auto-definition.

QMgrIdentifier (**MQCFST**)

Queue manager identifier (parameter identifier: MQCA_Q_MGR_IDENTIFIER).

The unique identifier of the queue manager.

QMgrName (**MQCFST**)

Queue manager name (parameter identifier: MQCA_CLUSTER_Q_MGR_NAME).

The maximum length of the string is MQ_Q_MGR_NAME_LENGTH.

QMgrType (**MQCFIN**)

Queue manager type (parameter identifier: MQIACF_Q_MGR_TYPE).

The value can be:

MQQMT_NORMAL

A normal queue manager.

MQQMT_REPOSITORY

A repository queue manager.

ReceiveExit (MQCFST)

Receive exit name (parameter identifier: MQCACH_RCV_EXIT_NAME).

The maximum length of the string is MQ_EXIT_NAME_LENGTH.

In the following environments, if more than one receive exit has been defined for the channel, the list of names is returned in an MQCFSL structure instead of an MQCFST structure: AIX, HP-UX, i5/OS, Solaris, Linux, and Windows. An MQCFSL structure is always used on z/OS.

ReceiveUserData (MQCFST)

Receive exit user data (parameter identifier:

MQCACH_RCV_EXIT_USER_DATA).

The maximum length of the string is MQ_EXIT_DATA_LENGTH.

In the following environments, if more than one receive exit user data string has been defined for the channel, the list of strings is returned in an MQCFSL structure instead of an MQCFST structure: AIX, HP-UX, i5/OS, Solaris, Linux, and Windows. An MQCFSL structure is always used on z/OS.

SecurityExit (MQCFST)

Security exit name (parameter identifier: MQCACH_SEC_EXIT_NAME).

The maximum length of the string is MQ_EXIT_NAME_LENGTH.

SecurityUserData (MQCFST)

Security exit user data (parameter identifier:

MQCACH_SEC_EXIT_USER_DATA).

The maximum length of the string is MQ_EXIT_DATA_LENGTH.

SendExit (MQCFST)

Send exit name (parameter identifier: MQCACH_SEND_EXIT_NAME).

The maximum length of the string is MQ_EXIT_NAME_LENGTH.

In the following environments, if more than one send exit has been defined for the channel, the list of names is returned in an MQCFSL structure instead of an MQCFST structure: AIX, HP-UX, i5/OS, Solaris, Linux, and Windows. An MQCFSL structure is always used on z/OS.

SendUserData (MQCFST)

Send exit user data (parameter identifier:

MQCACH_SEND_EXIT_USER_DATA).

The maximum length of the string is MQ_EXIT_DATA_LENGTH.

In the following environments, if more than one send exit user data string has been defined for the channel, the list of strings is returned in an MQCFSL structure instead of an MQCFST structure: AIX, HP-UX, i5/OS, Solaris, Linux, and Windows. An MQCFSL structure is always used on z/OS.

SqNumberWrap (MQCFIN)

Sequence wrap number (parameter identifier:

MQIACH_SEQUENCE_NUMBER_WRAP).

ShortRetryCount (MQCFIN)

Short retry count (parameter identifier: MQIACH_SHORT_RETRY).

ShortRetryInterval (**MQCFIN**)

Short timer (parameter identifier: MQIACH_SHORT_TIMER).

SSLCipherSpec (**MQCFST**)

CipherSpec (parameter identifier: MQCACH_SSL_CIPHER_SPEC).

The length of the string is MQ_SSL_CIPHER_SPEC_LENGTH.

SSLClientAuth (**MQCFIN**)

Client authentication (parameter identifier: MQIACH_SSL_CLIENT_AUTH).

The value can be:

MQSCA_REQUIRED

Client authentication required

MQSCA_OPTIONAL

Client authentication is optional.

Defines whether WebSphere MQ requires a certificate from the SSL client.

SSLPeerName (**MQCFST**)

Peer name (parameter identifier: MQCACH_SSL_PEER_NAME).

The length of the string is MQ_SSL_PEER_NAME_LENGTH. On z/OS, it is MQ_SHORT_PEER_NAME_LENGTH.

Specifies the filter to use to compare with the Distinguished Name of the certificate from the peer queue manager or client at the other end of the channel. (A Distinguished Name is the identifier of the SSL certificate.) If the Distinguished Name in the certificate received from the peer does not match the SSLPEER filter, the channel does not start.

Suspend (**MQCFIN**)

Whether the queue manager is suspended (parameter identifier: MQIACF_SUSPEND).

The value can be:

MQSUS_NO

The queue manager is not suspended from the cluster.

MQSUS_YES

The queue manager is suspended from the cluster.

TpName (**MQCFST**)

Transaction program name (parameter identifier: MQCACH_TP_NAME).

The maximum length of the string is MQ_TP_NAME_LENGTH.

TransportType (**MQCFIN**)

Transmission protocol type (parameter identifier: MQIACH_XMIT_PROTOCOL_TYPE).

The value can be:

MQXPT_LU62

LU 6.2.

MQXPT_TCP

TCP.

MQXPT_NETBIOS

NetBIOS.

MQXPT_SPX
SPX.

MQXPT_DECNET
DECnet.

UserIdentifier (MQCFST)

Task user identifier (parameter identifier: MQCACH_USER_ID). This parameter is not available on z/OS.

The maximum length of the string is MQ_USER_ID_LENGTH. However, only the first 10 characters are used.

Inquire Connection

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
		X	X	X	X

The Inquire connection (MQCMD_INQUIRE_CONNECTION) command inquires about the applications which are connected to the queue manager, the status of any transactions that those applications are running, and the objects which the application has open.

Required parameters :

ConnectionId, GenericConnectionId

Optional parameters:

CommandScope, ConnectionAttrs, ConnInfoType, IntegerFilterCommand, StringFilterCommand

Required parameters

ConnectionId (MQCFBS)

Connection identifier (parameter identifier: MQBACF_CONNECTION_ID).

This is the unique connection identifier associated with an application that is connected to the queue manager. Specify either this parameter **or** *GenericConnectionId*.

All connections are assigned a unique identifier by the queue manager regardless of how the connection is established.

If you need to specify a generic connection identifier, use the *GenericConnectionId* parameter instead.

The length of the string is MQ_CONNECTION_ID_LENGTH.

GenericConnectionId (MQCFBS)

Generic specification of a connection identifier (parameter identifier: MQBACF_GENERIC_CONNECTION_ID).

Specify either this parameter **or** *ConnectionId*.

If you specify a byte string of zero length, or one which contains only null bytes, information about all connection identifiers is returned. This is the only value permitted for *GenericConnectionId*.

The length of the string is MQ_CONNECTION_ID_LENGTH.

Optional parameters

ByteStringFilterCommand (**MQCFBF**)

Byte string filter command descriptor. The parameter identifier must be MQBACF_EXTERNAL_UOW_ID, MQBACF_ORIGIN_UOW_ID, or MQBACF_Q_MGR_UOW_ID. Use this to restrict the output from the command by specifying a filter condition. See “MQCFBF - PCF byte string filter parameter” on page 422 for information about using this filter condition.

If you specify a byte string filter, you cannot also specify an integer filter using the *IntegerFilterCommand* parameter, or a string filter using the *StringFilterCommand* parameter.

CommandScope (**MQCFST**)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_Q_MGR_NAME_LENGTH.

You cannot use *CommandScope* as a parameter to filter on.

ConnectionAttrs (**MQCFIL**)

Connection attributes (parameter identifier: MQIACF_CONNECTION_ATTRS).

The attribute list can specify the following on its own (this is the default value if the parameter is not specified) :

MQIACF_ALL

All attributes of the selected *ConnInfoType*.

or, if you select a value of MQIACF_CONN_INFO_CONN for *ConnInfoType*, a combination of the following:

MQBACF_CONNECTION_ID

Connection identifier.

MQBACF_EXTERNAL_UOW_ID

External unit of recovery identifier associated with the connection.

MQBACF_ORIGIN_UOW_ID

Unit of recovery identifier assigned by the originator (valid on z/OS only).

MQBACF_Q_MGR_UOW_ID

Unit of recovery identifier assigned by the queue manager.

MQCACF_APPL_TAG

Name of an application that is connected to the queue manager.

MQCACF_ASID

The 4-character address-space identifier of the application identified in MQCACF_APPL_TAG (valid on z/OS only).

MQCACF_ORIGIN_NAME

Originator of the unit of recovery (valid on z/OS only).

MQCACF_PSB_NAME

The 8-character name of the program specification block (PSB) associated with the running IMS transaction (valid on z/OS only).

MQCACF_PST_ID

The 4-character IMS program specification table (PST) region identifier for the connected IMS region (valid on z/OS only).

MQCACF_TASK_NUMBER

A 7-digit CICS task number (valid on z/OS only).

MQCACF_TRANSACTION_ID

A 4-character CICS transaction identifier (valid on z/OS only).

MQCACF_UOW_LOG_EXTENT_NAME

Name of the first extent required to recover the transaction. This is not valid on z/OS.

MQCACF_UOW_LOG_START_DATE

Date on which the transaction associated with the current connection first wrote to the log.

MQCACF_UOW_LOG_START_TIME

Time at which the transaction associated with the current connection first wrote to the log.

MQCACF_UOW_START_DATE

Date on which the transaction associated with the current connection was started.

MQCACF_UOW_START_TIME

Time at which the transaction associated with the current connection was started.

MQCACF_USER_IDENTIFIER

User identifier of the application that is connected to the queue manager.

MQCACH_CHANNEL_NAME

Name of the channel associated with the connected application.

MQCACH_CONNECTION_NAME

Connection name of the channel associated with the application.

MQIA_APP_TYPE

Type of the application that is connected to the queue manager.

MQIACF_CONNECT_OPTIONS

Connect options currently in force for this application connection.

You cannot use the value MQCNO_STANDARD_BINDING as a filter value.

MQIACF_PROCESS_ID

Process identifier of the application that is currently connected to the queue manager.

This parameter is not valid on z/OS.

MQIACF_THREAD_ID

Thread identifier of the application that is currently connected to the queue manager.

This parameter is not valid on z/OS.

MQIACF_UOW_STATE

State of the unit of work.

MQIACF_UOW_TYPE

Type of external unit of recovery identifier as understood by the queue manager.

or, if you select a value of MQIACF_CONN_INFO_HANDLE for *ConnInfoType*, a combination of the following:

MQCACF_OBJECT_NAME

Name of each object that the connection has open.

MQCACH_CONNECTION_NAME

Connection name of the channel associated with the application.

MQIA_QSG_DISP

Disposition of the object (valid on z/OS only).

You cannot use MQIA_QSG_DISP as a parameter to filter on.

MQIACF_HANDLE_STATE

Whether an API call is in progress.

MQIACF_OBJECT_TYPE

Type of each object that the connection has open.

MQIACF_OPEN_OPTIONS

Options used by the connection to open each object.

or, if you select a value of MQIACF_CONN_INFO_ALL for *ConnInfoType*, any of the above.

***ConnInfoType* (MQCFIN)**

Type of connection information to be returned (parameter identifier: MQIACF_CONN_INFO_TYPE).

The value can be:

MQIACF_CONN_INFO_CONN

Connection information. On z/OS, this includes threads which may be logically or actually disassociated from a connection, together with those that are in-doubt and for which external intervention is needed to resolve them. This is the default value used if the parameter is not specified.

MQIACF_CONN_INFO_HANDLE

Information pertaining only to those objects opened by the specified connection.

MQIACF_CONN_INFO_ALL

Connection information and information about those objects that the connection has open.

You cannot use *ConnInfoType* as a parameter to filter on.

***IntegerFilterCommand* (MQCFIF)**

Integer filter command descriptor. The parameter identifier must be any

integer type parameter allowed in *ConnectionAttrs* except as noted and MQIACF_ALL. Use this to restrict the output from the command by specifying a filter condition. You cannot use the value MQCNO_STANDARD_BINDING on the MQIACF_CONNECT_OPTIONS parameter with either the MQCFOP_CONTAINS or MQCFOP_EXCLUDES operator. See “MQCFIF - PCF integer filter parameter” on page 427 for information about using this filter condition.

If you filter on MQIACF_CONNECT_OPTIONS or MQIACF_OPEN_OPTIONS, in each case the filter value must have only one bit set.

If you specify an integer filter, you cannot also specify a byte string filter using the *ByteStringFilterCommand* parameter or a string filter using the *StringFilterCommand* parameter.

StringFilterCommand (MQCFSF)

String filter command descriptor. The parameter identifier must be any string type parameter allowed in *ConnectionAttrs*. Use this to restrict the output from the command by specifying a filter condition. See “MQCFSF - PCF string filter parameter” on page 434 for information about using this filter condition.

If you specify a string filter, you cannot also specify a byte string filter using the *ByteStringFilterCommand* parameter or an integer filter using the *IntegerFilterCommand* parameter.

Error codes

This command might return the following in the response format header, in addition to the values shown on page “Error codes applicable to all commands” on page 27.

Reason (MQLONG)

The value can be:

MQRCCF_CONNECTION_ID_ERROR

Connection identifier not valid.

Inquire Connection (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
		X	X	X	X

The response to the Inquire Connection (MQCMD_INQUIRE_CONNECTION) command consists of the response header followed by the *ConnId* structure and a set of attribute parameter structures determined by the value of *ConnInfoType* in the Inquire command.

If the value of *ConnInfoType* was MQIACF_CONN_INFO_ALL, there is one message for each connection found with MQIACF_CONN_INFO_CONN, and *n* more messages per connection with MQIACF_CONN_INFO_HANDLE (where *n* is the number of objects that the connection has open).

Always returned:

ConnId, *ConnInfoType*

Always returned if ConnInfoType is MQIACF_CONN_INFO_HANDLE:
ObjectName, ObjectType, QSGDisposition

Returned if requested and ConnInfoType is MQIACF_CONN_INFO_CONN:
*ApplTag, ApplType, ASID, ChannelName, ConnectionName, ConnectionOptions,
OriginName, OriginUOWId, ProcessId, PSBName, PSTId, QMgrUOWId,
StartUOWLogExtent, TaskNumber, ThreadId, TransactionId, UOWIdentifier,
UOWLogStartDate, UOWLogStartTime, UOWStartDate, UOWStartTime, UOWState,
UOWType, UserId*

Returned if requested and ConnInfoType is MQIACF_CONN_INFO_HANDLE:
OpenOptions, HandleState

Response data

ApplTag (MQCFST)

Application tag (parameter identifier: MQCACF_APPL_TAG).

The maximum length is MQ_APPL_TAG_LENGTH.

ApplType (MQCFIN)

Application type (parameter identifier: MQIA_APPL_TYPE).

The value can be:

MQAT_QMGR

Queue manager process.

MQAT_CHANNEL_INITIATOR

Channel initiator.

MQAT_USER

User application.

MQAT_BATCH

Application using a batch connection (only on z/OS).

MQAT_RRS_BATCH

RRS-coordinated application using a batch connection (only on z/OS).

MQAT_CICS

CICS transaction (only on z/OS).

MQAT_IMS

IMS transaction (only on z/OS).

ASID (MQCFST)

Address space identifier (parameter identifier: MQCACF_ASID).

The 4-character address-space identifier of the application identified by *ApplTag*. It distinguishes duplicate values of *ApplTag*.

This parameter is valid only on z/OS.

The length of the string is MQ_ASID_LENGTH.

ChannelName (MQCFST)

Channel name (parameter identifier: MQCACH_CHANNEL_NAME).

The maximum length of the string is MQ_CHANNEL_NAME_LENGTH.

ConnectionId (MQCFBS)

Connection identifier (parameter identifier: MQBACF_CONNECTION_ID).

The length of the string is MQ_CONNECTION_ID_LENGTH.

ConnectionName (**MQCFST**)

Connection name (parameter identifier: MQCACH_CONNECTION_NAME).

The maximum length of the string is MQ_CONN_NAME_LENGTH.

ConnectionOptions (**MQCFIL**)

Connect options currently in force for the connection (parameter identifier: MQIACF_CONNECT_OPTIONS).

ConnInfoType (**MQCFIN**)

Type of information returned (parameter identifier: MQIACF_CONN_INFO_TYPE).

The value may be:

MQIACF_CONN_INFO_CONN

Generic information for the specified connection.

MQIACF_CONN_INFO_HANDLE

Information pertinent only to those objects opened by the specified connection.

HandleState (**MQCFIN**)

State of the handle (parameter identifier: MQIACF_HANDLE_STATE).

The value may be:

MQHSTATE_ACTIVE

An API call from this connection is currently in progress for this object. If the object is a queue, this condition can arise when an MQGET WAIT call is in progress.

If there is an MQGET SIGNAL outstanding, then this does not mean, by itself, that the handle is active.

MQHSTATE_INACTIVE

No API call from this connection is currently in progress for this object. If the object is a queue, this condition can arise when no MQGET WAIT call is in progress.

ObjectName (**MQCFST**)

Object name (parameter identifier: MQCACF_OBJECT_NAME).

The maximum length of the string is MQ_OBJECT_NAME_LENGTH.

ObjectType (**MQCFIN**)

Object type (parameter identifier: MQIACF_OBJECT_TYPE).

The value can be:

MQOT_Q

Queue.

MQOT_NAMELIST

Namelist.

MQOT_PROCESS

Process.

MQOT_Q_MGR

Queue manager.

MQOT_CHANNEL

Channel.

MQOT_AUTH_INFO

Authentication information object.

OpenOptions (MQCFIN)

Open options currently in force for the object for connection (parameter identifier: MQIACF_OPEN_OPTIONS).

OriginName (MQCFST)

Origin name (parameter identifier: MQCACF_ORIGIN_NAME).

Identifies the originator of the unit of recovery, except where *ApplType* is MQAT_RRS_BATCH when it is omitted.

This parameter is valid only on z/OS.

The length of the string is MQ_ORIGIN_NAME_LENGTH.

OriginUOWId (MQCFBS)

Origin UOW identifier (parameter identifier: MQBACF_ORIGIN_UOW_ID).

The unit of recovery identifier assigned by the originator. It is an 8-byte value.

This parameter is valid only on z/OS.

The length of the string is MQ_UOW_ID_LENGTH.

ProcessId (MQCFIN)

Process identifier (parameter identifier: MQIACF_PROCESS_ID).

PSBName (MQCFST)

Program specification block name (parameter identifier: MQCACF_PSB_NAME).

The 8-character name of the program specification block (PSB) associated with the running IMS transaction.

This parameter is valid only on z/OS.

The length of the string is MQ_PSB_NAME_LENGTH.

PSTId (MQCFST)

Program specification table identifier (parameter identifier: MQCACF_PST_ID).

The 4-character IMS program specification table (PST) region identifier for the connected IMS region.

This parameter is valid only on z/OS.

The length of the string is MQ_PST_ID_LENGTH.

QMgrUOWId (MQCFBS)

Unit of recovery identifier assigned by the queue manager (parameter identifier: MQBACF_Q_MGR_UOW_ID).

On z/OS platforms, this is returned as a 6-byte RBA. On platforms other than z/OS, this is an 8-byte transaction identifier.

The maximum length of the string is MQ_UOW_ID_LENGTH.

QSGDisposition (MQCFIN)

QSG disposition (parameter identifier: MQIA_QSG_DISP).

Specifies the disposition of the object (that is, where it is defined and how it behaves). This is valid only on z/OS. The value can be:

MQQSGD_COPY

The object is defined as MQQSGD_COPY.

MQQSGD_Q_MGR

The object is defined as MQQSGD_Q_MGR.

MQQSGD_SHARED

The object is defined as MQQSGD_SHARED.

StartUOWLogExtent (MQCFST)

Name of the first extent needed to recover the transaction (parameter identifier: MQCACF_UOW_LOG_EXTENT_NAME).

The 8-character name of the program specification block (PSB) associated with the running IMS transaction.

This parameter is not valid on z/OS.

The maximum length of the string is MQ_LOG_EXTENT_NAME_LENGTH.

TaskNumber (MQCFST)

Task number (parameter identifier: MQCACF_TASK_NUMBER).

The 7-digit CICS task number.

This parameter is valid only on z/OS.

The maximum length of the string is MQ_TASK_NUMBER_LENGTH.

ThreadId (MQCFIN)

Thread identifier (parameter identifier: MQIACF_THREAD_ID).

TransactionId (MQCFST)

Transaction identifier (parameter identifier: MQCACF_TRANSACTION_ID).

The 4-character CICS transaction identifier.

This parameter is valid only on z/OS.

The maximum length of the string is MQ_TRANSACTION_ID_LENGTH.

UOWIdentifier (MQCFBS)

External unit of recovery identifier associated with the connection (parameter identifier: MQBACF_EXTERNAL_UOW_ID).

This is the recovery identifier for the unit of recovery. The value of *UOWType* determines its format.

The maximum length of the byte string is MQ_UOW_ID_LENGTH.

UOWLogStartDate (MQCFST)

Logged unit of work start date, in the form yyyy-mm-dd (parameter identifier: MQCACF_UOW_LOG_START_DATE).

The maximum length of the string is MQ_DATE_LENGTH.

UOWLogStartTime (MQCFST)

Logged unit of work start time, in the form hh.mm.ss (parameter identifier: MQCACF_UOW_LOG_START_TIME).

The maximum length of the string is MQ_TIME_LENGTH.

UOWStartDate (MQCFST)

Unit of work creation date (parameter identifier: MQCACF_UOW_START_DATE).

The maximum length of the string is MQ_DATE_LENGTH.

UOWStartTime (MQCFST)

Unit of work creation time (parameter identifier: MQCACF_UOW_START_TIME).

The maximum length of the string is MQ_TIME_LENGTH.

UOWState (**MQCFIN**)

State of the unit of work (parameter identifier: MQIACF_UOW_STATE).

The value can be:

MQUOWST_NONE

There is no unit of work.

MQUOWST_ACTIVE

The unit of work is active.

MQUOWST_PREPARED

The unit of work is in the process of being committed.

MQUOWST_UNRESOLVED

The unit of work is in the second phase of a two-phase commit operation. WebSphere MQ holds resources on its behalf and external intervention is required to resolve it. This might be as simple as starting the recovery coordinator (such as CICS, IMS, or RRS) or it might involve a more complex operation such as using the RESOLVE INDOUBT command. This value can occur only on z/OS.

UOWType (**MQCFIN**)

Type of external unit of recovery identifier as perceived by the queue manager (parameter identifier: MQIACF_UOW_TYPE).

The value can be:

MQUOWT_Q_MGR

MQUOWT_CICS

MQUOWT_RRS

MQUOWT_IMS

MQUOWT_XA

UserId (**MQCFST**)

User identifier (parameter identifier: MQCACF_USER_IDENTIFIER).

The maximum length of the string is MQ_MAX_USER_ID_LENGTH.

Inquire Entity Authority

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
		X	X	X	

The Inquire Entity Authority (MQCMD_INQUIRE_ENTITY_AUTH) command inquires about an entity's authorizations to a specified object.

Required parameters:

Options, ObjectType, EntityType, EntityName, ObjectName

Optional parameters:

ProfileAttrs, ServiceComponent

Required parameters

EntityName (MQCFST)

Entity name (parameter identifier: MQCACF_ENTITY_NAME).

Depending on the value of *EntityType*, this is either:

- A principal name. This is the name of a user for whom to retrieve authorizations to the specified object. On WebSphere MQ for Windows, the name of the principal can optionally include a domain name, specified in this format: user@domain.
- A group name. This is the name of the user group on which to make the inquiry. You can specify one name only and this must be the name of an existing user group. On WebSphere MQ for Windows, you can only use local groups.

The maximum length of the string is MQ_ENTITY_NAME_LENGTH.

EntityType (MQCFIN)

Entity type (parameter identifier: MQIACF_ENTITY_TYPE).

The value can be:

MQZAET_GROUP

The value of the *EntityName* parameter refers to a group name.

MQZAET_PRINCIPAL

The value of the *EntityName* parameter refers to a principal name.

ObjectName (MQCFST)

Object name (parameter identifier: MQCACF_OBJECT_NAME).

The name of the queue manager, queue, process definition or generic profile on which to make the inquiry.

You must include this parameter unless the *ObjectType* is MQOT_Q_MGR, in which case, you must omit it. If you do not include this parameter, it is assumed that you are making an inquiry on the queue manager.

You cannot specify a generic object name although you can specify the name of a generic profile.

The maximum length of the string is MQ_OBJECT_NAME_LENGTH.

ObjectType (MQCFIN)

The type of object referred to by the profile (parameter identifier: MQIACF_OBJECT_TYPE).

The value can be:

MQOT_AUTH_INFO

Authentication information.

MQOT_CHANNEL

Channel object.

MQOT_CLNTCONN_CHANNEL

Client-connection channel object.

MQOT_LISTENER

Listener object.

MQOT_NAMELIST

Namelist.

MQOT_PROCESS

Process.

MQOT_Q

Queue, or queues, that match the object name parameter.

MQOT_Q_MGR

Queue manager.

MQOT_SERVICE

Service object.

Options (MQCFIN)

Options to control the set of authority records that is returned (parameter identifier: MQIACF_AUTH_OPTIONS).

This parameter is required and you should set it to the value MQAUTHOPT_CUMULATIVE. It returns a set of authorities representing the cumulative authority that an entity has to a specified object.

If a user ID is a member of more than one group, this command displays the combined authorizations of all groups.

Optional parameters

ProfileAttrs (MQCFIL)

Profile attributes (parameter identifier: MQIACF_AUTH_PROFILE_ATTRS).

The attribute list might specify the following on its own (this is the default value if the parameter is not specified):

MQIACF_ALL

All attributes.

or a combination of the following:

MQCACF_ENTITY_NAME

Entity name.

MQIACF_AUTHORIZATION_LIST

Authorization list.

MQIACF_ENTITY_TYPE

Entity type.

MQIACF_OBJECT_TYPE

Object type.

ServiceComponent (MQCFST)

Service component (parameter identifier: MQCACF_SERVICE_COMPONENT).

If installable authorization services are supported, this specifies the name of the authorization service to which the authorizations apply.

If you omit this parameter, the authorization inquiry is made to the first installable component for the service.

The maximum length of the string is MQ_SERVICE_COMPONENT_LENGTH.

Error codes

This command might return the following in the response format header, in addition to the values shown on page "Error codes applicable to all commands" on page 27.

Reason (MQLONG)

The value can be:

MQRC_UNKNOWN_ENTITY

User ID not authorized, or unknown.

MQRCCF_OBJECT_TYPE_MISSING

Object type missing.

Inquire Entity Authority (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
		X	X	X	

Each response to the Inquire Entity Authority (MQCMD_INQUIRE_AUTH_RECS) command consists of the response header followed by the *QMgrName*, *Options*, and *ObjectName* structures and the requested combination of attribute parameter structures.

Always returned:

ObjectName, *Options*, *QMgrName*

Returned if requested:

AuthorizationList, *EntityName*, *EntityType*, *ObjectType*

Response data

AuthorizationList (MQCFIL)

Authorization list(parameter identifier: MQIACF_AUTHORIZATION_LIST).

This list can contain zero or more authorization values. Each returned authorization value means that any user ID in the specified group or principal has the authority to perform the operation defined by that value. The value can be:

MQAUTH_ALT_USER_AUTHORITY

Specify an alternate user ID on an MQI call.

MQAUTH_BROWSE

Retrieve a message from a queue by issuing an MQGET call with the BROWSE option.

MQAUTH_CHANGE

Change the attributes of the specified object, using the appropriate command set.

MQAUTH_CLEAR

Clear a queue.

MQAUTH_CONNECT

Connect the application to the specified queue manager by issuing an MQCONN call.

MQAUTH_CREATE

Create objects of the specified type using the appropriate command set.

MQAUTH_DELETE

Delete the specified object using the appropriate command set.

MQAUTH_DISPLAY

Display the attributes of the specified object using the appropriate command set.

MQAUTH_INPUT

Retrieve a message from a queue by issuing an MQGET call.

MQAUTH_INQUIRE

Make an inquiry on a specific queue by issuing an MQINQ call.

MQAUTH_OUTPUT

Put a message on a specific queue by issuing an MQPUT call.

MQAUTH_PASS_ALL_CONTEXT

Pass all context.

MQAUTH_PASS_IDENTITY_CONTEXT

Pass the identity context.

MQAUTH_SET

Set attributes on a queue from the MQI by issuing an MQSET call.

MQAUTH_SET_ALL_CONTEXT

Set all context on a queue.

MQAUTH_SET_IDENTITY_CONTEXT

Set the identity context on a queue.

Use the *Count* field in the MQCFIL structure to determine how many values are returned.

***EntityName* (MQCFST)**

Entity name (parameter identifier: MQCACF_ENTITY_NAME).

This can either be a principal name or a group name.

The maximum length of the string is MQ_ENTITY_NAME_LENGTH.

***EntityType* (MQCFIN)**

Entity type (parameter identifier: MQIACF_ENTITY_TYPE).

The value can be:

MQZAET_GROUP

The value of the *EntityName* parameter refers to a group name.

MQZAET_PRINCIPAL

The value of the *EntityName* parameter refers to a principal name.

MQZAET_UNKNOWN

On Windows, an authority record still exists from a previous queue manager which did not originally contain entity type information.

***ObjectName* (MQCFST)**

Object name (parameter identifier: MQCACF_OBJECT_NAME).

The name of the queue manager, queue, process definition or generic profile on which the inquiry is made.

The maximum length of the string is MQ_OBJECT_NAME_LENGTH.

***ObjectType* (MQCFIN)**

Object type (parameter identifier: MQIACF_OBJECT_TYPE).

The value can be:

MQOT_AUTH_INFO

Authentication information.

MQOT_CHANNEL

Channel object.

MQOT_CLNTCONN_CHANNEL

Client-connection channel object.

MQOT_LISTENER

Listener object.

MQOT_NAMELIST

Namelist.

MQOT_PROCESS

Process.

MQOT_Q

Queue, or queues, that match the object name parameter.

MQOT_Q_MGR

Queue manager.

MQOT_SERVICE

Service object.

***QMgrName* (MQCFST)**

Name of the queue manager on which the Inquire command is issued (parameter identifier: MQCA_Q_MGR_NAME).

The maximum length of the string is MQ_Q_MGR_NAME_LENGTH.

Inquire Group

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The Inquire Group (MQCMD_INQUIRE_QSG) command inquires about the queue-sharing group to which the queue manager is connected.

Note: This command is supported only on z/OS when the queue manager is a member of a queue-sharing group.

Required parameters:

None

Optional parameters:

None

Inquire Group (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The response to the Inquire Group (MQCMD_INQUIRE_QSG) command consists of the response header followed by the *QMgrName* structure and a number of other parameter structures. One such message is generated for each queue manager in the queue-sharing group. If there are any obsolete DB2 messages, one message, identified by a value of MQCMDI_DB2_OBSOLETE_MSGS in the *CommandInformation* parameter, is returned for each such message.

Always returned for the queue manager:

CommandLevel, *DB2ConnectStatus*, *DB2Name*, *QmgrCPF*, *QMgrName*, *QmgrNumber*,
QMgrStatus, *QSGName*

Always returned for obsolete DB2 messages:

CommandInformation, *CFMsgIdentifier*

Response data relating to the queue manager

***CommandLevel* (MQCFIN)**

Command level supported by the queue manager (parameter identifier: MQIA_COMMAND_LEVEL). The value can be:

MQCMDL_LEVEL_520

Level 520 of system control commands.

MQCMDL_LEVEL_530

Level 530 of system control commands.

MQCMDL_LEVEL_531

Level 531 of system control commands.

MQCMDL_LEVEL_600

Level 600 of system control commands.

***DB2ConnectStatus* (MQCFIN)**

The current status of the connection to DB2 (parameter identifier: MQIACF_DB2_CONN_STATUS).

The current status of the queue manager. The value can be:

MQQSGS_ACTIVE

The queue manager is running and is connected to DB2.

MQQSGS_INACTIVE

The queue manager is not running and is not connected to DB2.

MQQSGS_FAILED

The queue manager is running but not connected because DB2 has terminated abnormally.

MQQSGS_PENDING

The queue manager is running but not connected because DB2 has terminated normally.

MQQSGS_UNKNOWN

The status cannot be determined.

***DB2Name* (MQCFST)**

The name of the DB2 subsystem or group to which the queue manager is to connect (parameter identifier: MQCACF_DB2_NAME).

The maximum length is MQ_Q_MGR_CPF_LENGTH.

***QMgrCPF* (MQCFST)**

The command prefix of the queue manager (parameter identifier: MQCA_Q_MGR_CPF).

The maximum length is MQ_Q_MGR_CPF_LENGTH.

QMgrName (**MQCFST**)

Name of the queue manager (parameter identifier: MQCA_Q_MGR_NAME).

The maximum length is MQ_Q_MGR_NAME_LENGTH.

QmgrNumber (**MQCFIN**)

The number, generated internally, of the queue manager in the group.(parameter identifier: MQIACF_Q_MGR_NUMBER).

QMgrStatus (**MQCFIN**)

Recovery (parameter identifier: MQIACF_Q_MGR_STATUS).

The current status of the queue manager. The value can be:

MQQSGS_ACTIVE

The queue manager is running.

MQQSGS_INACTIVE

The queue manager is not running, having terminated normally.

MQQSGS_FAILED

The queue manager is not running, having terminated abnormally.

MQQSGS_CREATED

The queue manager has been defined to the group, but has not yet been started.

MQQSGS_UNKNOWN

The status cannot be determined.

QSGName (**MQCFST**)

The name of the queue sharing group (parameter identifier: MQCA_QSG_NAME).

The maximum length is MQ_QSG_NAME_LENGTH.

Response data relating to obsolete DB2 messages

CFMsgIdentifier (**MQCFBS**)

CF list entry identifier (parameter identifier: MQBACF_CF_LEID).

The maximum length is MQ_CF_LEID_LENGTH.

CommandInformation (**MQCFIN**)

Command information (parameter identifier: MQIACF_COMMAND_INFO).

This indicates whether queue managers in the group contain obsolete messages. The value is MQCMDI_DB2_OBSOLETE_MSGS.

Inquire Log

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The Inquire Log (MQCMD_INQUIRE_LOG) command returns log system parameters and information.

Required parameters:

None

Optional parameters:
CommandScope

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

Inquire Log (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The response to the Inquire Log (MQCMD_INQUIRE_LOG) command consists of the response header followed by the *ParameterType* structure and the combination of attribute parameter structures determined by the value of *ParameterType*.

Always returned:

ParameterType. Specifies the type of archive information being returned.
The value can be:

MQSYSP_TYPE_INITIAL

The initial settings of the log parameters.

MQSYSP_TYPE_SET

The settings of the log parameters if they have been altered since their initial setting.

MQSYSP_TYPE_LOG_COPY

Information relating to the active log copy.

MQSYSP_TYPE_LOG_STATUS

Information relating to the status of the logs.

Returned if *ParameterType* is MQSYSP_TYPE_INITIAL (one message is returned):

DeallocateInterval, DualArchive, DualActive, DualBSDS, InputBufferSize, LogArchive, MaxArchiveLog, MaxReadTapeUnits, OutputBufferCount, OutputBufferSize

Returned if ParameterType is MQSYSP_TYPE_SET and any value is set (one message is returned):

*DeallocateInterval, DualArchive, DualActive, DualBSDS, InputBufferSize,
LogArchive, MaxArchiveLog, MaxReadTapeUnits, OutputBufferCount,
OutputBufferSize*

Returned if ParameterType is MQSYSP_TYPE_LOG_COPY (one message is returned for each log copy):

DataSetName, LogCopyNumber, LogUsed

Returned if ParameterType is MQSYSP_TYPE_LOG_STATUS (one message is returned):

*FullLogs, LogRBA, LogSuspend, OffloadStatus, QMgrStartDate, QMgrStartRBA,
QMgrStartTime, TotalLogs*

Response data - log parameter information

DeallocateInterval (MQCFIN)

Deallocation interval (parameter identifier:
MQIACF_SYSP_DEALLOC_INTERVAL).

Specifies the length of time, in minutes, that an allocated archive read tape unit is allowed to remain unused before it is deallocated. The value can be in the range zero through 1440. If it is zero, the tape unit is deallocated immediately. If it is 1440, the tape unit is never deallocated.

DualActive (MQCFIN)

Specifies whether dual logging is being used (parameter identifier:
MQIACF_SYSP_DUAL_ACTIVE).

The value can be:

MQSYSP_YES

Dual logging is being used.

MQSYSP_NO

Dual logging is not being used.

DualArchive (MQCFIN)

Specifies whether dual archive logging is being used (parameter identifier:
MQIACF_SYSP_DUAL_ARCHIVE).

The value can be:

MQSYSP_YES

Dual archive logging is being used.

MQSYSP_NO

Dual archive logging is not being used.

DualBSDS (MQCFIN)

Specifies whether dual BSDS is being used (parameter identifier:
MQIACF_SYSP_DUAL_BSDS).

The value can be:

MQSYSP_YES

Dual BSDS is being used.

MQSYSP_NO

Dual BSDS is not being used.

InputBufferSize (**MQCFIN**)

Specifies the size of input buffer storage for active and archive log data sets (parameter identifier: MQIACF_SYSP_IN_BUFFER_SIZE).

LogArchive (**MQCFIN**)

Specifies whether archiving is on or off (parameter identifier: MQIACF_SYSP_ARCHIVE).

The value can be:

MQSYSP_YES

Archiving is on.

MQSYSP_NO

Archiving is off.

MaxArchiveLog (**MQCFIN**)

Specifies the maximum number of archive log volumes that can be recorded in the BSDS (parameter identifier: MQIACF_SYSP_MAX_ARCHIVE).

MaxReadTapeUnits (**MQCFIN**)

Specifies the maximum number of dedicated tape units that can be allocated to read archive log tape volumes (parameter identifier: MQIACF_SYSP_MAX_READ_TAPES).

OutputBufferCount (**MQCFIN**)

Specifies the number of output buffers to be filled before they are written to the active log data sets (parameter identifier: MQIACF_SYSP_OUT_BUFFER_COUNT).

OutputBufferSize (**MQCFIN**)

Specifies the size of output buffer storage for active and archive log data sets (parameter identifier: MQIACF_SYSP_OUT_BUFFER_SIZE).

Response data - to log status information

DataSetName (**MQCFST**)

The data set name of the active log data set (parameter identifier: MQCACF_DATA_SET_NAME).

If the copy is not currently active, this is returned as blank.

The maximum length of the string is

MQ_DATA_DATA_SET_NAME_LENGTH.

FullLogs (**MQCFIN**)

The total number of full active log data sets that have not yet been archived (parameter identifier: MQIACF_SYSP_FULL_LOGS).

LogCopyNumber (**MQCFIN**)

Copy number (parameter identifier: MQIACF_SYSP_LOG_COPY).

LogRBA (**MQCFST**)

The RBA of the most recently written log record (parameter identifier: MQCACF_SYSP_LOG_RBA).

The maximum length of the string is MQ_RBA_LENGTH.

LogSuspend (**MQCFIN**)

Specifies whether logging is suspended (parameter identifier: MQIACF_SYSP_LOG_SUSPEND).

The value can be:

MQSYSP_YES

Logging is suspended.

MQSYSP_NO

Logging is not suspended.

LogUsed (MQCFIN)

The percentage of the active log data set that has been used (parameter identifier: MQIACF_SYSP_LOG_USED).

OffloadStatus (MQCFIN)

Specifies the status of the offload task (parameter identifier: MQIACF_SYSP_OFFLOAD_STATUS).

The value can be:

MQSYSP_STATUS_ALLOCATING_ARCHIVE

The offload task is busy, allocating the archive data set. This could indicate that a tape mount request is pending.

MQSYSP_STATUS_COPYING_BSDS

The offload task is busy, copying the BSDS data set.

MQSYSP_STATUS_COPYING_LOG

The offload task is busy, copying the active log data set.

MQSYSP_STATUS_BUSY

The offload task is busy with other processing.

MQSYSP_STATUS_AVAILABLE

The offload task is waiting for work.

QMgrStartDate (MQCFST)

The date on which the queue manager was started, in the form yyyy-mm-dd (parameter identifier: MQCACF_SYSP_Q_MGR_DATE).

The maximum length of the string is MQ_DATE_LENGTH.

QMgrStartRBA (MQCFST)

The RBA from which logging began when the queue manager was started (parameter identifier: MQCACF_SYSP_Q_MGR_RBA).

The maximum length of the string is MQ_RBA_LENGTH.

QMgrStartTime (MQCFST)

The time that the queue manager was started, in the form hh.mm.ss (parameter identifier: MQCACF_SYSP_Q_MGR_TIME).

The maximum length of the string is MQ_TIME_LENGTH.

TotalLogs (MQCFIN)

The total number of active log data sets (parameter identifier: MQIACF_SYSP_TOTAL_LOGS).

Inquire Namelist

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Inquire Namelist (MQCMD_INQUIRE_NAMELIST) command inquires about the attributes of existing WebSphere MQ namelists.

Required parameters:

NamelistName

Optional parameters:

CommandScope, *IntegerFilterCommand*, *NamelistAttrs*, *QSGDisposition*,

StringFilterCommand

Required parameters

NamelistName (MQCFST)

Namelist name (parameter identifier: MQCA_NAMELIST_NAME).

This is the name of the namelist whose attributes are required. Generic namelist names are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all namelists having names that start with the selected character string. An asterisk on its own matches all possible names.

The namelist name is always returned regardless of the attributes requested.

The maximum length of the string is MQ_NAMELIST_NAME_LENGTH.

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

You cannot use *CommandScope* as a parameter to filter on.

IntegerFilterCommand (MQCFIF)

Integer filter command descriptor. The parameter identifier must be any integer type parameter allowed in *NamelistAttrs* except MQIACF_ALL. Use this to restrict the output from the command by specifying a filter condition. See “MQCFIF - PCF integer filter parameter” on page 427 for information about using this filter condition.

If you specify an integer filter for *NamelistType* (MQIA_NAMELIST_TYPE), you cannot also specify the *NamelistType* parameter.

If you specify an integer filter, you cannot also specify a string filter using the *StringFilterCommand* parameter.

NamelistAttrs (MQCFIL)

Namelist attributes (parameter identifier: MQIACF_NAMELIST_ATTRS).

The attribute list might specify the following on its own (this is the default value if the parameter is not specified):

MQIACF_ALL

All attributes.

or a combination of the following:

MQCA_NAMELIST_NAME

Name of namelist object.

MQCA_NAMELIST_DESC

Namelist description.

MQCA NAMES

Names in the namelist.

MQCA_ALTERATION_DATE

The date on which the information was last altered.

MQCA_ALTERATION_TIME

The time at which the information was last altered.

MQIA_NAME_COUNT

Number of names in the namelist.

MQIA_NAMELIST_TYPE

Namelist type (valid only on z/OS)

NamelistType (MQCFIN)

Namelist attributes (parameter identifier: MQIA_NAMELIST_TYPE). This parameter applies to z/OS only.

Specifies the type of names in the namelist. The value can be:

MQNT_NONE

The names are of no particular type.

MQNT_Q

A namelist that holds a list of queue names.

MQNT_CLUSTER

A namelist that is associated with clustering, containing a list of the cluster names.

MQNT_AUTH_INFO

The namelist is associated with SSL, and contains a list of authentication information object names.

QSGDisposition (MQCFIN)

Disposition of the object within the group (parameter identifier: MQIA_QSG_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object for which information is to be returned (that is, where it is defined and how it behaves). The value can be:

MQQSGD_LIVE

The object is defined as MQQSGD_Q_MGR or MQQSGD_COPY. This is the default value if the parameter is not specified.

MQQSGD_ALL

The object is defined as MQQSGD_Q_MGR or MQQSGD_COPY.

If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this option also displays information for objects defined with MQQSGD_GROUP.

If MQQSGD_LIVE is specified or defaulted, or if MQQSGD_ALL is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

MQQSGD_COPY

The object is defined as MQQSGD_COPY.

MQQSGD_GROUP

The object is defined as MQQSGD_GROUP. This is permitted only in a shared queue environment.

MQQSGD_Q_MGR

The object is defined as MQQSGD_Q_MGR.

MQQSGD_PRIVATE

The object is defined as either MQQSGD_Q_MGR or MQQSGD_COPY. Note that MQQSGD_PRIVATE returns the same information as MQQSGD_LIVE.

You cannot use *QSGDisposition* as a parameter to filter on.

StringFilterCommand (MQCFSF)

String filter command descriptor. The parameter identifier must be any string type parameter allowed in *NamelistAttrs* except MQCA_NAMELIST_NAME. Use this to restrict the output from the command by specifying a filter condition. See “MQCFSF - PCF string filter parameter” on page 434 for information about using this filter condition.

If you specify a string filter, you cannot also specify an integer filter using the *IntegerFilterCommand* parameter.

Inquire Namelist (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The response to the Inquire Namelist (MQCMD_INQUIRE_NAMELIST) command consists of the response header followed by the *NamelistName* structure and the requested combination of attribute parameter structures. If a generic namelist name was specified, one such message is generated for each namelist found.

Always returned:

NamelistName, QSGDisposition

Returned if requested:

AlterationDate, AlterationTime, NameCount , NamelistDesc, NamelistType, Names

Response data

AlterationDate (MQCFST)

Alteration date (parameter identifier: MQCA_ALTERATION_DATE).

The date when the information was last altered, in the form yyyy-mm-dd.

AlterationTime (**MQCFST**)

Alteration time (parameter identifier: MQCA_ALTERATION_TIME).

The time when the information was last altered, in the form hh.mm.ss .

NameCount (**MQCFIN**)

Number of names in the namelist (parameter identifier:
MQIA_NAME_COUNT).

The number of names contained in the namelist.

NamelistDesc (**MQCFST**)

Description of namelist definition (parameter identifier:
MQCA_NAMELIST_DESC).

The maximum length of the string is MQ_NAMELIST_DESC_LENGTH.

NamelistName (**MQCFST**)

The name of the namelist definition (parameter identifier:
MQCA_NAMELIST_NAME).

The maximum length of the string is MQ_NAMELIST_NAME_LENGTH.

NamelistType (**MQCFIN**)

Type of names in the namelist (parameter identifier: MQIA_NAMELIST_TYPE).
This parameter applies to z/OS only.

Specifies the type of names in the namelist . The value can be:

MQNT_NONE

The names are of no particular type.

MQNT_Q

A namelist that holds a list of queue names.

MQNT_CLUSTER

A namelist that is associated with clustering, containing a list of the cluster names.

MQNT_AUTH_INFO

The namelist is associated with SSL, and contains a list of authentication information object names.

Names (**MQCFSL**)

A list of the names contained in the namelist (parameter identifier:
MQCA_NAMES).

The number of names in the list is given by the *Count* field in the MQCFSL structure. The length of each name is given by the *StringLength* field in that structure. The maximum length of a name is MQ_OBJECT_NAME_LENGTH.

QSGDisposition (**MQCFIN**)

QSG disposition (parameter identifier: MQIA_QSG_DISP).

Specifies the disposition of the object (that is, where it is defined and how it behaves). This parameter applies only to z/OS. The value can be:

MQQSGD_COPY

The object is defined as MQQSGD_COPY.

MQQSGD_GROUP

The object is defined as MQQSGD_GROUP.

MQQSGD_Q_MGR

The object is defined as MQQSGD_Q_MGR.

Inquire Namelist Names

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Inquire Namelist Names (MQCMD_INQUIRE_NAMELIST_NAMES) command inquires for a list of namelist names that match the generic namelist name specified.

Required parameters:

NamelistName

Optional parameters:

CommandScope, QSGDisposition

Required parameters

NamelistName (MQCFST)

Name of namelist (parameter identifier: MQCA_NAMELIST_NAME).

Generic namelist names are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all objects having names that start with the selected character string. An asterisk on its own matches all possible names.

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

QSGDisposition (MQCFIN)

Disposition of the object within the group (parameter identifier: MQIA_QSG_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object for which information is to be returned (that is, where it is defined and how it behaves). The value can be:

MQQSGD_LIVE

The object is defined as MQQSGD_Q_MGR or MQQSGD_COPY. This is the default value if the parameter is not specified.

MQQSGD_ALL

The object is defined as MQQSGD_Q_MGR or MQQSGD_COPY.

If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this option also displays information for objects defined with MQQSGD_GROUP.

If MQQSGD_LIVE is specified or defaulted, or if MQQSGD_ALL is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

MQQSGD_COPY

The object is defined as MQQSGD_COPY.

MQQSGD_GROUP

The object is defined as MQQSGD_GROUP. This is permitted only in a shared queue environment.

MQQSGD_Q_MGR

The object is defined as MQQSGD_Q_MGR.

MQQSGD_PRIVATE

The object is defined with either MQQSGD_Q_MGR or MQQSGD_COPY. Note that MQQSGD_PRIVATE returns the same information as MQQSGD_LIVE.

Inquire Namelist Names (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The response to the Inquire Namelist Names (MQCMD_INQUIRE_NAMELIST_NAMES) command consists of the response header followed by a single parameter structure giving zero or more names that match the specified namelist name.

In addition to this, on z/OS only, the *QSGDispositions* structure (with the same number of entries as the *NamelistNames* structure) is returned. Each entry in this structure indicates the disposition of the object with the corresponding entry in the *NamelistNames* structure.

Always returned:

NamelistNames, *QSGDispositions*

Returned if requested:

None

Response data

NamelistNames (MQCFSL)

List of namelist names (parameter identifier: MQCACF_NAMELIST_NAMES).

QSGDispositions (MQCFIL)

List of QSG dispositions (parameter identifier: MQIACF_QSG_DISPS). This parameter is valid only on z/OS. Possible values for fields in this structure are:

MQQSGD_COPY

The object is defined as MQQSGD_COPY.

MQQSGD_GROUP

The object is defined as MQQSGD_GROUP. This is permitted only in a shared queue environment.

MQQSGD_Q_MGR

The object is defined as MQQSGD_Q_MGR.

Inquire Process

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Inquire Process (MQCMD_INQUIRE_PROCESS) command inquires about the attributes of existing WebSphere MQ processes.

Required parameters:

ProcessName

Optional parameters:

*CommandScope, IntegerFilterCommand, ProcessAttrs, QSGDisposition,
StringFilterCommand*

Required parameters

ProcessName (MQCFST)

Process name (parameter identifier: MQCA_PROCESS_NAME).

Generic process names are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all processes having names that start with the selected character string. An asterisk on its own matches all possible names.

The process name is always returned regardless of the attributes requested.

The maximum length of the string is MQ_PROCESS_NAME_LENGTH.

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

You cannot use *CommandScope* as a parameter to filter on.

IntegerFilterCommand (MQCFIF)

Integer filter command descriptor. The parameter identifier must be any integer type parameter allowed in *ProcessAttrs* except MQIACF_ALL. Use this to restrict the output from the command by specifying a filter condition. See “MQCFIF - PCF integer filter parameter” on page 427 for information about using this filter condition.

If you specify an integer filter, you cannot also specify a string filter using the *StringFilterCommand* parameter.

ProcessAttrs (MQCFIL)

Process attributes (parameter identifier: MQIACF_PROCESS_ATTRS).

The attribute list might specify the following on its own (this is the default value used if the parameter is not specified):

MQIACF_ALL

All attributes.

or a combination of the following:

MQCA_ALTERATION_DATE

The date at which the information was last altered.

MQCA_ALTERATION_TIME

The time at which the information was last altered.

MQCA_APPL_ID

Application identifier.

MQCA_ENV_DATA

Environment data.

MQCA_PROCESS_DESC

Description of process definition.

MQCA_PROCESS_NAME

Name of process definition.

MQCA_USER_DATA

User data.

MQIA_APPL_TYPE

Application type.

QSGDisposition (MQCFIN)

Disposition of the object within the group (parameter identifier: MQIA_QSG_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object for which information is to be returned (that is, where it is defined and how it behaves). The value can be:

MQQSGD_LIVE

The object is defined as MQQSGD_Q_MGR or MQQSGD_COPY. This is the default value if the parameter is not specified.

MQQSGD_ALL

The object is defined as MQQSGD_Q_MGR or MQQSGD_COPY.

If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this option also displays information for objects defined with MQQSGD_GROUP.

If MQQSGD_LIVE is specified or defaulted, or if MQQSGD_ALL is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

MQQSGD_COPY

The object is defined as MQQSGD_COPY.

MQQSGD_GROUP

The object is defined as MQQSGD_GROUP. This is permitted only in a shared queue environment.

MQQSGD_Q_MGR

The object is defined as MQQSGD_Q_MGR.

MQQSGD_PRIVATE

The object is defined as either MQQSGD_Q_MGR or MQQSGD_COPY. Note that MQQSGD_PRIVATE returns the same information as MQQSGD_LIVE.

You cannot use *QSGDisposition* as a parameter to filter on.

StringFilterCommand (MQCFSF)

String filter command descriptor. The parameter identifier must be any string type parameter allowed in *ProcessAttrs* except MQCA_PROCESS_NAME. Use this to restrict the output from the command by specifying a filter condition. See “MQCFSF - PCF string filter parameter” on page 434 for information about using this filter condition.

If you specify a string filter, you cannot also specify an integer filter using the *IntegerFilterCommand* parameter.

Inquire Process (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The response to the Inquire Process (MQCMD_INQUIRE_PROCESS) command consists of the response header followed by the *ProcessName* structure and the requested combination of attribute parameter structures. If a generic process name was specified, one such message is generated for each process found.

Always returned:

ProcessName, QSGDisposition

Returned if requested:

AlterationDate, AlterationTime, ApplId, ApplType, EnvData, ProcessDesc, UserData

Response data

AlterationDate (MQCFST)

Alteration date (parameter identifier: MQCA_ALTERATION_DATE).

The date when the information was last altered, in the form yyyy-mm-dd.

AlterationTime (MQCFST)

Alteration time (parameter identifier: MQCA_ALTERATION_TIME).

The time when the information was last altered, in the form hh.mm.ss.

ApplId (**MQCFST**)

Application identifier (parameter identifier: MQCA_APPL_ID).

The maximum length of the string is MQ_PROCESS_APPL_ID_LENGTH.

ApplType (**MQCFIN**)

Application type (parameter identifier: MQIA_APPL_TYPE).

The value can be:

MQAT_OS400

i5/OS application.

MQAT_OS2

OS/2 or Presentation Manager® application.

MQAT_DOS

DOS client application.

MQAT_WINDOWS

Windows client or Windows 3.1 application.

MQAT_WINDOWS_NT

Windows or Windows 95, Windows 98 application.

MQAT_UNIX

UNIX application.

MQAT_AIX

AIX application (same value as MQAT_UNIX).

MQAT_CICS

CICS transaction.

MQAT_MVS

z/OS application.

integer: System-defined application type in the range zero through 65 535 or a user-defined application type in the range 65 536 through 999 999 999.

EnvData (**MQCFST**)

Environment data (parameter identifier: MQCA_ENV_DATA).

The maximum length of the string is MQ_PROCESS_ENV_DATA_LENGTH.

ProcessDesc (**MQCFST**)

Description of process definition (parameter identifier: MQCA_PROCESS_DESC).

The maximum length of the string is MQ_PROCESS_DESC_LENGTH.

ProcessName (**MQCFST**)

The name of the process definition (parameter identifier: MQCA_PROCESS_NAME).

The maximum length of the string is MQ_PROCESS_NAME_LENGTH.

QSGDisposition (**MQCFIN**)

QSG disposition (parameter identifier: MQIA_QSG_DISP).

Specifies the disposition of the object (that is, where it is defined and how it behaves). This parameter is valid on z/OS only. The value can be:

MQQSGD_COPY

The object is defined as MQQSGD_COPY.

MQQSGD_GROUP

The object is defined as MQQSGD_GROUP.

MQQSGD_Q_MGR

The object is defined as MQQSGD_Q_MGR.

***UserData* (MQCFST)**

User data (parameter identifier: MQCA_USER_DATA).

The maximum length of the string is MQ_PROCESS_USER_DATA_LENGTH.

Inquire Process Names

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Inquire Process Names (MQCMD_INQUIRE_PROCESS_NAMES) command inquires for a list of process names that match the generic process name specified.

Required parameters:

ProcessName

Optional parameters:

CommandScope, *QSGDisposition*

Required parameters

***ProcessName* (MQCFST)**

Name of process-definition for queue (parameter identifier: MQCA_PROCESS_NAME).

Generic process names are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all objects having names that start with the selected character string. An asterisk on its own matches all possible names.

Optional parameters

***CommandScope* (MQCFST)**

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

QSGDisposition (**MQCFIN**)

Disposition of the object within the group (parameter identifier: MQIA_QSG_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object for which information is to be returned (that is, where it is defined and how it behaves). The value can be:

MQQSGD_LIVE

The object is defined as MQQSGD_Q_MGR or MQQSGD_COPY. This is the default value if the parameter is not specified.

MQQSGD_ALL

The object is defined as MQQSGD_Q_MGR or MQQSGD_COPY.

If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this option also displays information for objects defined with MQQSGD_GROUP.

If MQQSGD_LIVE is specified or defaulted, or if MQQSGD_ALL is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

MQQSGD_COPY

The object is defined as MQQSGD_COPY.

MQQSGD_GROUP

The object is defined as MQQSGD_GROUP. This is permitted only in a shared queue environment.

MQQSGD_Q_MGR

The object is defined as MQQSGD_Q_MGR.

MQQSGD_PRIVATE

The object is defined with either MQQSGD_Q_MGR or MQQSGD_COPY. Note that MQQSGD_PRIVATE returns the same information as MQQSGD_LIVE.

Inquire Process Names (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The response to the Inquire Process Names (MQCMD_INQUIRE_PROCESS_NAMES) command consists of the response header followed by a single parameter structure giving zero or more names that match the specified process name.

In addition to this, on z/OS only, a parameter structure, *QSGDispositions* (with the same number of entries as the *ProcessNames* structure) is returned. Each entry in this structure indicates the disposition of the object with the corresponding entry in the *ProcessNames* structure.

This response is not supported on Windows.

Always returned:

ProcessNames, *QSGDispositions*

Returned if requested:
None

Response data

ProcessNames (MQCFSL)

List of process names (parameter identifier: MQCACF_PROCESS_NAMES).

QSGDispositions (MQCFIL)

List of QSG dispositions (parameter identifier: MQIACF_QSG_DISPS). This parameter applies only to z/OS. Possible values for fields in this structure are:

MQQSGD_COPY

The object is defined as MQQSGD_COPY.

MQQSGD_GROUP

The object is defined as MQQSGD_GROUP.

MQQSGD_Q_MGR

The object is defined as MQQSGD_Q_MGR.

Inquire Queue

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Inquire Queue (MQCMD_INQUIRE_Q) command inquires about the attributes of WebSphere MQ queues.

Required parameters:

QName

Optional parameters:

ClusterInfo, ClusterName, ClusterNamelist, CommandScope,, IntegerFilterCommand, PageSetID, QAttrs, QSGDisposition, QType, StringFilterCommand

Required parameters

QName (MQCFST)

Queue name (parameter identifier: MQCA_Q_NAME).

Generic queue names are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all queues having names that start with the selected character string. An asterisk on its own matches all possible names.

The queue name is always returned, regardless of the attributes requested.

The maximum length of the string is MQ_Q_NAME_LENGTH.

Optional parameters

CFStructure (MQCFST)

Storage class (parameter identifier: MQCA_CF_STRUC_NAME). Specifies the name of the storage class. This parameter is valid only on z/OS.

This specifies that eligible queues are limited to those having the specified *CFStructure* value. If this is not specified, then all queues are eligible.

Generic CF structure names are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all CF structures having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length of the string is MQ_CF_STRUC_NAME_LENGTH.

ClusterInfo (MQCFIN)

Cluster information (parameter identifier: MQIACF_CLUSTER_INFO).

This parameter requests that, in addition to information about attributes of queues defined on this queue manager, cluster information about these and other queues in the repository that match the selection criteria will be displayed.

In this case, there might be multiple queues with the same name displayed. The cluster information is shown with a queue type of MQQT_CLUSTER.

You can set this parameter to any integer value, the value used does not affect the response to the command.

The cluster information is obtained locally from the queue manager.

ClusterName (MQCFST)

Cluster name (parameter identifier: MQCA_CLUSTER_NAME).

This specifies that eligible queues are limited to those having the specified *ClusterName* value. If this is not specified, then all queues are eligible.

Generic cluster names are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all clusters having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length of the string is MQ_CLUSTER_NAME_LENGTH.

ClusterNamelist (MQCFST)

Cluster namelist (parameter identifier: MQCA_CLUSTER_NAMELIST).

This specifies that eligible queues are limited to those having the specified *ClusterNameList* value. If this is not specified, then all queues are eligible.

Generic cluster namelists are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all cluster namelists having names that start with the selected character string. An asterisk on its own matches all possible names.

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.

- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

You cannot use *CommandScope* as a parameter to filter on.

IntegerFilterCommand (MQCFIF)

Integer filter command descriptor. The parameter identifier must be any integer type parameter allowed in *QAttrs* except MQIACF_ALL. Use this to restrict the output from the command by specifying a filter condition. See “MQCFIF - PCF integer filter parameter” on page 427 for information about using this filter condition.

If you specify an integer filter for *Qtype* or *PageSetID*, you cannot also specify the *Qtype* or *PageSetID* parameter.

If you specify an integer filter, you cannot also specify a string filter using the *StringFilterCommand* parameter.

PageSetID (MQCFIN)

Page set identifier (parameter identifier: MQIA_PAGESET_ID). This parameter applies to z/OS only.

This specifies that eligible queues are limited to those having the specified *PageSetID* value. If this is not specified, then all queues are eligible.

QAttrs (MQCFIL)

Queue attributes (parameter identifier: MQIACF_Q_ATTRS).

The attribute list might specify the following on its own (this is the default value used if the parameter is not specified):

MQIACF_ALL

All attributes.

or a combination of the parameters in the following table:

Table 9. Inquire Queue command, queue attributes

	Local queue	Model queue	Alias queue	Remote queue	Cluster queue
MQCA_ALTERATION_DATE The date on which the information was last altered	X	X	X	X	X
MQCA_ALTERATION_TIME The time at which the information was last altered	X	X	X	X	X
MQCA_BACKOUT_REQ_Q_NAME Excessive backout requeue name	X	X			
MQCA_BASE_NAME Name of queue that alias resolves to			X		
MQCA_CF_STRUC_NAME Coupling facility structure name. This attribute is valid on z/OS only	X	X			

Table 9. Inquire Queue command, queue attributes (continued)

	Local queue	Model queue	Alias queue	Remote queue	Cluster queue
MQCA_CLUSTER_DATE Date when the definition became available to the local queue manager					X
MQCA_CLUSTER_NAME Cluster name	X		X	X	X
MQCA_CLUSTER_NAMELIST Cluster namelist	X		X	X	
MQCA_CLUSTER_Q_MGR_NAME Queue manager name that hosts the queue					X
MQCA_CLUSTER_TIME Time when the definition became available to the local queue manager					X
MQCA_CREATION_DATE Queue creation date	X	X			
MQCA_CREATION_TIME Queue creation time	X	X			
MQCA_INITIATION_Q_NAME Initiation queue name	X	X			
MQCA_PROCESS_NAME Name of process definition	X	X			
MQCA_Q_DESC Queue description	X	X	X	X	X
MQCA_Q_MGR_IDENTIFIER Internally generated queue manager name					X
MQCA_Q_NAME Queue name	X	X	X	X	X
MQCA_REMOTE_Q_MGR_NAME Name of remote queue manager				X	
MQCA_REMOTE_Q_NAME Name of remote queue as known locally on the remote queue manager				X	
MQCA_STORAGE_CLASS Storage class. This is valid on z/OS only	X	X			

Table 9. Inquire Queue command, queue attributes (continued)

	Local queue	Model queue	Alias queue	Remote queue	Cluster queue
MQCA_TPIPE_NAME The TPIPE name used for communication with OTMA using the WebSphere MQ IMS Bridge	X				
MQCA_TRIGGER_DATA Trigger data	X	X			
MQCA_XMIT_Q_NAME Transmission queue name				X	
MQIA_ACCOUNTING_Q Accounting data collection	X	X			
MQIA_BACKOUT_THRESHOLD Backout threshold	X	X			
MQIA_CLUSTER_Q_TYPE Cluster queue type					X
MQIA_CLWL_Q_PRIORITY Cluster workload queue priority	X		X	X	X
MQIA_CLWL_Q_RANK Cluster workload queue rank	X		X	X	X
MQIA_CLWL_USEQ Cluster workload use remote setting	X				
MQIA_CURRENT_Q_DEPTH Number of messages on queue	X				
MQIA_DEF_BIND Default binding	X		X	X	X
MQIA_DEF_INPUT_OPEN_OPTION Default open-for-input option	X	X			
MQIA_DEF_PERSISTENCE Default message persistence	X	X	X	X	X
MQIA_DEF_PRIORITY Default message priority	X	X	X	X	X
MQIA_DEFINITION_TYPE Queue definition type	X	X			
MQIA_DIST_LISTS Distribution list support. This is not valid on z/OS	X	X			

Table 9. Inquire Queue command, queue attributes (continued)

	Local queue	Model queue	Alias queue	Remote queue	Cluster queue
MQIA_HARDEN_GET_BACKOUT Whether to harden backout count	X	X			
MQIA_INDEX_TYPE Index type. This attribute is valid on z/OS only.	X	X			
MQIA_INHIBIT_GET Whether get operations are allowed	X	X	X		
MQIA_INHIBIT_PUT Whether put operations are allowed	X	X	X	X	X
MQIA_MAX_MSG_LENGTH Maximum message length	X	X			
MQIA_MAX_Q_DEPTH Maximum number of messages allowed on queue	X	X			
MQIA_MONITORING_Q Online monitoring data collection	X	X			
MQIA_MSG_DELIVERY_SEQUENCE Whether message priority is relevant	X	X			
MQIA_NPM_CLASS Level of reliability assigned to non-persistent messages that are put to the queue	X	X			
MQIA_OPEN_INPUT_COUNT Number of MQOPEN calls that have the queue open for input	X				
MQIA_OPEN_OUTPUT_COUNT Number of MQOPEN calls that have the queue open for output	X				
MQIA_PAGESET_ID Page set identifier	X				
MQIA_Q_DEPTH_HIGH_EVENT Control attribute for queue depth high events. You cannot use this as a filter attribute.	X	X			
MQIA_Q_DEPTH_HIGH_LIMIT High limit for queue depth	X	X			

Table 9. Inquire Queue command, queue attributes (continued)

	Local queue	Model queue	Alias queue	Remote queue	Cluster queue
MQIA_Q_DEPTH_LOW_EVENT Control attribute for queue depth low events. You cannot use this as a filter attribute.	X	X			
MQIA_Q_DEPTH_LOW_LIMIT Low limit for queue depth	X	X			
MQIA_Q_DEPTH_MAX_EVENT Control attribute for queue depth max events	X	X			
MQIA_Q_SERVICE_INTERVAL Limit for queue service interval	X	X			
MQIA_Q_SERVICE_INTERVAL_EVENT Control attribute for queue service interval events	X	X			
MQIA_Q_TYPE Queue type	X	X	X	X	X
MQIA_RETENTION_INTERVAL Queue retention interval	X	X			
MQIA_SCOPE Queue definition scope. This is not valid on z/OS or i5/OS	X		X	X	
MQIA_SHAREABILITY Whether queue can be shared	X	X			
MQIA_STATISTICS_Q Statistics data collection. This is valid only on AIX, HP-UX, Linux, i5/OS, Solaris, and Windows.	X	X			
MQIA_TRIGGER_CONTROL Trigger control	X	X			
MQIA_TRIGGER_DEPTH Trigger depth	X	X			
MQIA_TRIGGER_MSG_PRIORITY Threshold message priority for triggers	X	X			
MQIA_TRIGGER_MTYPE Trigger type	X	X			

Table 9. Inquire Queue command, queue attributes (continued)

	Local queue	Model queue	Alias queue	Remote queue	Cluster queue
MQIA_USAGE Usage	X	X			

QSGDisposition (MQCFIN)

Disposition of the object within the group (parameter identifier: MQIA_QSG_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object for which information is to be returned (that is, where it is defined and how it behaves). The value can be:

MQQSGD_LIVE

The object is defined as MQQSGD_Q_MGR or MQQSGD_COPY. If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this also returns information for objects defined with MQQSGD_SHARED. This is the default value if the parameter is not specified.

MQQSGD_ALL

The object is defined as MQQSGD_Q_MGR or MQQSGD_COPY.

If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this option also displays information for objects defined with MQQSGD_GROUP or MQQSGD_SHARED.

If MQQSGD_LIVE is specified or defaulted, or if MQQSGD_ALL is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

MQQSGD_COPY

The object is defined as MQQSGD_COPY.

MQQSGD_GROUP

The object is defined as MQQSGD_GROUP. This is permitted only in a shared queue environment.

MQQSGD_Q_MGR

The object is defined as MQQSGD_Q_MGR.

MQQSGD_PRIVATE

The object is defined with either MQQSGD_Q_MGR or MQQSGD_COPY.

MQQSGD_SHARED

The object is defined as MQQSGD_SHARED. This is permitted only in a shared queue environment.

You cannot use *QSGDisposition* as a parameter to filter on.

QType (MQCFIN)

Queue type (parameter identifier: MQIA_Q_TYPE).

If this parameter is present, eligible queues are limited to those of the specified type. Any attribute selector specified in the *QAttrs* list which is valid only for queues of a different type or types is ignored; no error is raised.

If this parameter is not present (or if MQQT_ALL is specified), queues of all types are eligible. Each attribute specified must be a valid queue attribute

selector (that is, it must be one of those in the following list), but it need not be applicable to all (or any) of the queues actually returned. Queue attribute selectors that are valid but not applicable to the queue are ignored, no error messages occur and no attribute is returned. The value can be:

MQQT_ALL

All queue types.

MQQT_LOCAL

Local queue.

MQQT_ALIAS

Alias queue definition.

MQQT_REMOTE

Local definition of a remote queue.

MQQT_CLUSTER

Cluster queue.

MQQT_MODEL

Model queue definition.

Note: On platforms other than z/OS, if this parameter is present, it must occur immediately after the *QName* parameter.

StorageClass (**MQCFST**)

Storage class (parameter identifier: MQCA_STORAGE_CLASS). Specifies the name of the storage class. This parameter is valid only on z/OS.

This specifies that eligible queues are limited to those having the specified *StorageClass* value. If this is not specified, then all queues are eligible.

Generic names are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all storage classes having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length of the string is MQ_STORAGE_CLASS_LENGTH.

StringFilterCommand (**MQCFSF**)

String filter command descriptor. The parameter identifier must be any string type parameter allowed in *QAttrs* except MQCA_Q_NAME. Use this to restrict the output from the command by specifying a filter condition. See “MQCFSF - PCF string filter parameter” on page 434 for information about using this filter condition.

If you specify a string filter for *ClusterName*, *ClusterNameList*, *StorageClass*, or *CStructure*, you cannot also specify that as a parameter.

If you specify a string filter, you cannot also specify an integer filter using the *IntegerFilterCommand* parameter.

Error codes

This command might return the following in the response format header, in addition to the values shown on page “Error codes applicable to all commands” on page 27.

Reason (**MQLONG**)

The value can be:

MQRCCF_Q_TYPE_ERROR
Queue type not valid.

Inquire Queue (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The response to the Inquire Queue (MQCMD_INQUIRE_Q) command consists of the response header followed by the *QName* structure, and, on z/OS only, the *QSGDisposition* structure, and the requested combination of attribute parameter structures. If a generic queue name was specified, or cluster queues requested (either by using MQQT_CLUSTER or MQIACF_CLUSTER_INFO), one such message is generated for each queue found.

Always returned:

QName, QSGDisposition, QType

Returned if requested:

AlterationDate, AlterationTime, BackoutRequeueName, BackoutThreshold, BaseQName, CFStructure, ClusterDate, ClusterName, ClusterNamelist, ClusterQType, ClusterTime, CLWLQueuePriority, CLWLQueueRank, CLWLUseQ, CreationDate, CreationTime, CurrentQDepth, DefBind, DefinitionType, DefInputOpenOption, DefPersistence, DefPriority, DistLists, HardenGetBackout, IndexType, InhibitGet, InhibitPut, InitiationQName, MaxMsgLength, MaxQDepth, MsgDeliverySequence, NonPersistentMessageClass, OpenInputCount, OpenOutputCount, PageSetID, ProcessName, QDepthHighEvent, QDepthHighLimit, QDepthLowEvent, QDepthLowLimit, QDepthMaxEvent, QDesc, QMgrIdentifier, QMgrName, QServiceInterval, QServiceIntervalEvent, QueueAccounting, QueueMonitoring, QueueStatistics, RemoteQMgrName, RemoteQName, RetentionInterval, Scope, Shareability, StorageClass, TpipeNames, TriggerControl, TriggerData, TriggerDepth, TriggerMsgPriority, TriggerType, Usage, XmitQName

Response data

AlterationDate (MQCFST)

Alteration date (parameter identifier: MQCA_ALTERATION_DATE).

The date when the information was last altered, in the form yyyy-mm-dd.

AlterationTime (MQCFST)

Alteration time (parameter identifier: MQCA_ALTERATION_TIME).

The time when the information was last altered, in the form hh.mm.ss.

BackoutRequeueName (MQCFST)

Excessive backout requeue name (parameter identifier: MQCA_BACKOUT_REQ_Q_NAME).

The maximum length of the string is MQ_Q_NAME_LENGTH.

BackoutThreshold (MQCFIN)

Backout threshold (parameter identifier: MQIA_BACKOUT_THRESHOLD).

***BaseQName* (MQCFST)**

Queue name to which the alias resolves (parameter identifier: MQCA_BASE_Q_NAME).

This is the name of a queue that is defined to the local queue manager.

The maximum length of the string is MQ_Q_NAME_LENGTH.

***CFStructure* (MQCFST)**

Coupling facility structure name (parameter identifier: MQCA_CF_STRUC_NAME). This parameter applies to z/OS only.

Specifies the name of the coupling facility structure where you want to store messages when you use shared queues.

The maximum length of the string is MQ_CF_STRUC_NAME_LENGTH.

***ClusterDate* (MQCFST)**

Cluster date (parameter identifier: MQCA_CLUSTER_DATE).

The date on which the information became available to the local queue manager, in the form yyyy-mm-dd.

***ClusterName* (MQCFST)**

Cluster name (parameter identifier: MQCA_CLUSTER_NAME).

***ClusterNamelist* (MQCFST)**

Cluster namelist (parameter identifier: MQCA_CLUSTER_NAMELIST).

***ClusterQType* (MQCFIN)**

Cluster queue type (parameter identifier: MQIA_CLUSTER_Q_TYPE).

The value can be:

MQCQT_LOCAL_Q

The cluster queue represents a local queue.

MQCQT_ALIAS_Q

The cluster queue represents an alias queue.

MQCQT_REMOTE_Q

The cluster queue represents a remote queue.

MQCQT_Q_MGR_ALIAS

The cluster queue represents a queue manager alias.

***ClusterTime* (MQCFST)**

Cluster time (parameter identifier: MQCA_CLUSTER_TIME).

The time at which the information became available to the local queue manager, in the form hh.mm.ss.

***CLWLQueuePriority* (MQCFIN)**

Cluster workload queue priority (parameter identifier: MQIA_CLWL_Q_PRIORITY).

Priority of the queue in cluster workload management. The value is in the range zero through 9, where zero is the lowest priority and 9 is the highest.

***CLWLQueueRank* (MQCFIN)**

Cluster workload queue rank (parameter identifier: MQIA_CLWL_Q_RANK).

Rank of the queue in cluster workload management. The value is in the range zero through 9, where zero is the lowest rank and 9 is the highest.

***CLWLUseQ* (MQCFIN)**

Cluster workload queue rank (parameter identifier: MQIA_CLWL_USEQ).

The value can be:

MQCLWL_USEQ_AS_Q_MGR

Use the value of the *CLWLUseQ* parameter on the queue manager's definition.

MQCLWL_USEQ_ANY

Use remote and local queues.

MQCLWL_USEQ_LOCAL

Do not use remote queues.

CreationDate (**MQCFST**)

Queue creation date, in the form yyyy-mm-dd (parameter identifier: MQCA_CREATION_DATE).

The maximum length of the string is MQ_CREATION_DATE_LENGTH.

CreationTime (**MQCFST**)

Creation time, in the form hh.mm.ss (parameter identifier: MQCA_CREATION_TIME).

The maximum length of the string is MQ_CREATION_TIME_LENGTH.

CurrentQDepth (**MQCFIN**)

Current queue depth (parameter identifier: MQIA_CURRENT_Q_DEPTH).

DefBind (**MQCFIN**)

Default binding (parameter identifier: MQIA_DEF_BIND).

The value can be:

MQBND_BIND_ON_OPEN

Binding fixed by MQOPEN call.

MQBND_BIND_NOT_FIXED

Binding not fixed.

DefinitionType (**MQCFIN**)

Queue definition type (parameter identifier: MQIA_DEFINITION_TYPE).

The value can be:

MQQDT_PREDEFINED

Predefined permanent queue.

MQQDT_PERMANENT_DYNAMIC

Dynamically defined permanent queue.

MQQDT_SHARED_DYNAMIC

Dynamically defined shared queue. This option is available on z/OS only.

MQQDT_TEMPORARY_DYNAMIC

Dynamically defined temporary queue.

DefInputOpenOption (**MQCFIN**)

Default input open option for defining whether queues can be shared (parameter identifier: MQIA_DEF_INPUT_OPEN_OPTION).

The value can be:

MQOO_INPUT_EXCLUSIVE

Open queue to get messages with exclusive access.

MQOO_INPUT_SHARED

Open queue to get messages with shared access.

DefPersistence (MQCFIN)

Default persistence (parameter identifier: MQIA_DEF_PERSISTENCE).

The value can be:

MQPER_PERSISTENT

Message is persistent.

MQPER_NOT_PERSISTENT

Message is not persistent.

DefPriority (MQCFIN)

Default priority (parameter identifier: MQIA_DEF_PRIORITY).

DistLists (MQCFIN)

Distribution list support (parameter identifier: MQIA_DIST_LISTS).

The value can be:

MQDL_SUPPORTED

Distribution lists supported.

MQDL_NOT_SUPPORTED

Distribution lists not supported.

This parameter is supported in the following environments: AIX, HP-UX, i5/OS, Solaris, Windows and Linux.

HardenGetBackout (MQCFIN)

Whether to harden backout (parameter identifier:

MQIA_HARDEN_GET_BACKOUT).

The value can be:

MQQA_BACKOUT_HARDENED

Backout count remembered.

MQQA_BACKOUT_NOT_HARDENED

Backout count may not be remembered.

IndexType (MQCFIN)

Index type (parameter identifier: MQIA_INDEX_TYPE). This parameter applies to z/OS only.

Specifies the type of index maintained by the queue manager to expedite MQGET operations on the queue. The value can be:

MQIT_NONE

No index.

MQIT_MSG_ID

The queue is indexed using message identifiers.

MQIT_CORREL_ID

The queue is indexed using correlation identifiers.

MQIT_MSG_TOKEN

The queue is indexed using message tokens.

MQIT_GROUP_ID

The queue is indexed using group identifiers.

InhibitGet (MQCFIN)

Whether get operations are allowed (parameter identifier:

MQIA_INHIBIT_GET).

The value can be:

MQQA_GET_ALLOWED

Get operations are allowed.

MQQA_GET_INHIBITED

Get operations are inhibited.

InhibitPut (MQCFIN)

Whether put operations are allowed (parameter identifier: MQIA_INHIBIT_PUT).

The value can be:

MQQA_PUT_ALLOWED

Put operations are allowed.

MQQA_PUT_INHIBITED

Put operations are inhibited.

InitiationQName (MQCFST)

Initiation queue name (parameter identifier: MQCA_INITIATION_Q_NAME).

The maximum length of the string is MQ_Q_NAME_LENGTH.

MaxMsgLength (MQCFIN)

Maximum message length (parameter identifier: MQIA_MAX_MSG_LENGTH).

MaxQDepth (MQCFIN)

Maximum queue depth (parameter identifier: MQIA_MAX_Q_DEPTH).

MsgDeliverySequence (MQCFIN)

Whether priority is relevant (parameter identifier: MQIA_MSG_DELIVERY_SEQUENCE).

The value can be:

MQMDS_PRIORITY

Messages are returned in priority order.

MQMDS_FIFO

Messages are returned in FIFO order (first in, first out).

NonPersistentMessageClass (MQCFIN)

The level of reliability assigned to non-persistent messages that are put to the queue (parameter identifier: MQIA_NPM_CLASS).

Specifies the circumstances under which non-persistent messages put to the queue may be lost. The value can be:

MQNPM_CLASS_NORMAL

Non-persistent messages are limited to the lifetime of the queue manager session. They are discarded in the event of a queue manager restart. This is the default value.

MQNPM_CLASS_HIGH

The queue manager attempts to retain non-persistent messages for the lifetime of the queue. Non-persistent messages may still be lost in the event of a failure.

OpenInputCount (MQCFIN)

Number of MQOPEN calls that have the queue open for input (parameter identifier: MQIA_OPEN_INPUT_COUNT).

OpenOutputCount (**MQCFIN**)

Number of MQOPEN calls that have the queue open for output (parameter identifier: MQIA_OPEN_OUTPUT_COUNT).

PageSetID (**MQCFIN**)

Page set identifier (parameter identifier: MQIA_PAGESET_ID).

Specifies the identifier of the page set on which the queue resides.

This parameter applies to z/OS only when the queue is actively associated with a page set.

ProcessName (**MQCFST**)

Name of process definition for queue (parameter identifier: MQCA_PROCESS_NAME).

The maximum length of the string is MQ_PROCESS_NAME_LENGTH.

QDepthHighEvent (**MQCFIN**)

Controls whether Queue Depth High events are generated (parameter identifier: MQIA_Q_DEPTH_HIGH_EVENT).

The value can be:

MQEVR_DISABLED

Event reporting disabled.

MQEVR_ENABLED

Event reporting enabled.

QDepthHighLimit (**MQCFIN**)

High limit for queue depth (parameter identifier: MQIA_Q_DEPTH_HIGH_LIMIT).

The threshold against which the queue depth is compared to generate a Queue Depth High event.

QDepthLowEvent (**MQCFIN**)

Controls whether Queue Depth Low events are generated (parameter identifier: MQIA_Q_DEPTH_LOW_EVENT).

The value can be:

MQEVR_DISABLED

Event reporting disabled.

MQEVR_ENABLED

Event reporting enabled.

QDepthLowLimit (**MQCFIN**)

Low limit for queue depth (parameter identifier: MQIA_Q_DEPTH_LOW_LIMIT).

The threshold against which the queue depth is compared to generate a Queue Depth Low event.

QDepthMaxEvent (**MQCFIN**)

Controls whether Queue Full events are generated (parameter identifier: MQIA_Q_DEPTH_MAX_EVENT).

The value can be:

MQEVR_DISABLED

Event reporting disabled.

MQEVR_ENABLED

Event reporting enabled.

***QDesc* (MQCFST)**

Queue description (parameter identifier: MQCA_Q_DESC).

The maximum length of the string is MQ_Q_DESC_LENGTH.

***QMgrIdentifier* (MQCFST)**

Queue manager identifier (parameter identifier: MQCA_Q_MGR_IDENTIFIER).

The unique identifier of the queue manager.

***QMgrName* (MQCFST)**

Name of local queue manager (parameter identifier: MQCA_CLUSTER_Q_MGR_NAME).

The maximum length of the string is MQ_Q_MGR_NAME_LENGTH.

***QName* (MQCFST)**

Queue name (parameter identifier: MQCA_Q_NAME).

The maximum length of the string is MQ_Q_NAME_LENGTH.

***QServiceInterval* (MQCFIN)**

Target for queue service interval (parameter identifier: MQIA_Q_SERVICE_INTERVAL).

The service interval used for comparison to generate Queue Service Interval High and Queue Service Interval OK events.

***QServiceIntervalEvent* (MQCFIN)**

Controls whether Service Interval High or Service Interval OK events are generated (parameter identifier: MQIA_Q_SERVICE_INTERVAL_EVENT).

The value can be:

MQQSIE_HIGH

Queue Service Interval High events enabled.

MQQSIE_OK

Queue Service Interval OK events enabled.

MQQSIE_NONE

No queue service interval events enabled.

***QSGDisposition* (MQCFIN)**

QSG disposition (parameter identifier: MQIA_QSG_DISP).

Specifies the disposition of the object (that is, where it is defined and how it behaves). This is valid only on z/OS. The value can be:

MQQSGD_COPY

The object is defined as MQQSGD_COPY.

MQQSGD_GROUP

The object is defined as MQQSGD_GROUP.

MQQSGD_Q_MGR

The object is defined as MQQSGD_Q_MGR.

MQQSGD_SHARED

The object is defined as MQQSGD_SHARED.

***QType* (MQCFIN)**

Queue type (parameter identifier: MQIA_Q_TYPE).

The value can be:

MQQT_ALIAS

Alias queue definition.

MQQT_CLUSTER

Cluster queue definition.

MQQT_LOCAL

Local queue.

MQQT_REMOTE

Local definition of a remote queue.

MQQT_MODEL

Model queue definition.

QueueAccounting (MQCFIN)

Controls the collection of accounting (thread-level and queue-level accounting) data (parameter identifier: MQIA_ACCOUNTING_Q).

The value can be:

MQMON_Q_MGR

The collection of accounting data for the queue is performed based upon the setting of the *QueueAccounting* parameter on the queue manager.

MQMON_OFF

Do not collect accounting data for the queue.

MQMON_ON

Collect accounting data for the queue.

QueueMonitoring (MQCFIN)

Online monitoring data collection (parameter identifier: MQIA_MONITORING_Q).

The value can be:

MQMON_OFF

Online monitoring data collection is turned off for this queue.

MQMON_Q_MGR

The value of the queue manager's *QueueMonitoring* parameter is inherited by the queue.

MQMON_LOW

Online monitoring data collection is turned on, with a low rate of data collection, for this queue unless *QueueMonitoring* for the queue manager is MQMON_NONE.

MQMON_MEDIUM

Online monitoring data collection is turned on, with a moderate rate of data collection, for this queue unless *QueueMonitoring* for the queue manager is MQMON_NONE.

MQMON_HIGH

Online monitoring data collection is turned on, with a high rate of data collection, for this queue unless *QueueMonitoring* for the queue manager is MQMON_NONE.

QueueStatistics (MQCFIN)

Controls the collection of statistics data (parameter identifier: MQIA_STATISTICS_Q).

The value can be:

MQMON_Q_MGR

The collection of statistics data for the queue is performed based upon the setting of the *QueueStatistics* parameter on the queue manager.

MQMON_OFF

Do not collect statistics data for the queue.

MQMON_ON

Collect statistics data for the queue unless *QueueStatistics* for the queue manager is MQMON_NONE.

This parameter is valid only on i5/OS, UNIX systems, and Windows.

RemoteQMgrName (**MQCFST**)

Name of remote queue manager (parameter identifier: MQCA_REMOTE_Q_MGR_NAME).

The maximum length of the string is MQ_Q_MGR_NAME_LENGTH.

RemoteQName (**MQCFST**)

Name of remote queue as known locally on the remote queue manager (parameter identifier: MQCA_REMOTE_Q_NAME).

The maximum length of the string is MQ_Q_NAME_LENGTH.

RetentionInterval (**MQCFIN**)

Retention interval (parameter identifier: MQIA_RETENTION_INTERVAL).

Scope (**MQCFIN**)

Scope of the queue definition (parameter identifier: MQIA_SCOPE).

The value can be:

MQSCO_Q_MGR

Queue-manager scope.

MQSCO_CELL

Cell scope.

This parameter is not valid on i5/OS or z/OS.

Shareability (**MQCFIN**)

Whether queue can be shared (parameter identifier: MQIA_SHAREABILITY).

The value can be:

MQQA_SHAREABLE

Queue is shareable.

MQQA_NOT_SHAREABLE

Queue is not shareable.

StorageClass (**MQCFST**)

Storage class (parameter identifier: MQCA_STORAGE_CLASS). This parameter applies to z/OS only.

Specifies the name of the storage class.

The maximum length of the string is MQ_STORAGE_CLASS_LENGTH.

TpipeNames (**MQCFSL**)

TPIPE names (parameter identifier: MQCA_TPIPE_NAME). This parameter applies to local queues on z/OS only.

Specifies the TPIPE names used for communication with OTMA via the WebSphere MQ IMS bridge, if the bridge is active.

The maximum length of the string is MQ_TPIPE_NAME_LENGTH.

TriggerControl (MQCFIN)

Trigger control (parameter identifier: MQIA_TRIGGER_CONTROL).

The value can be:

MQTC_OFF

Trigger messages not required.

MQTC_ON

Trigger messages required.

TriggerData (MQCFST)

Trigger data (parameter identifier: MQCA_TRIGGER_DATA).

The maximum length of the string is MQ_TRIGGER_DATA_LENGTH.

TriggerDepth (MQCFIN)

Trigger depth (parameter identifier: MQIA_TRIGGER_DEPTH).

TriggerMsgPriority (MQCFIN)

Threshold message priority for triggers (parameter identifier: MQIA_TRIGGER_MSG_PRIORITY).

TriggerType (MQCFIN)

Trigger type (parameter identifier: MQIA_TRIGGER_TYPE).

The value can be:

MQTT_NONE

No trigger messages.

MQTT_FIRST

Trigger message when queue depth goes from 0 to 1.

MQTT_EVERY

Trigger message for every message.

MQTT_DEPTH

Trigger message when depth threshold exceeded.

Usage (MQCFIN)

Usage (parameter identifier: MQIA_USAGE).

The value can be:

MQUS_NORMAL

Normal usage.

MQUS_TRANSMISSION

Transmission queue.

XmitQName (MQCFST)

Transmission queue name (parameter identifier: MQCA_XMIT_Q_NAME).

The maximum length of the string is MQ_Q_NAME_LENGTH.

Inquire Queue Manager

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Inquire Queue Manager (MQCMD_INQUIRE_Q_MGR) command inquires about the attributes of a queue manager.

Required parameters:

None

Optional parameters:

CommandScope, QMgrAttrs

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

QMgrAttrs (MQCFIL)

Queue manager attributes (parameter identifier: MQIACF_Q_MGR_ATTRS).

The attribute list might specify the following on its own (this is the default value used if the parameter is not specified):

MQIACF_ALL

All attributes.

or a combination of the following:

MQCA_ALTERATION_DATE

Date at which the definition was last altered.

MQCA_ALTERATION_TIME

Time at which the definition was last altered.

MQCA_CHANNEL_AUTO_DEF_EXIT

Automatic channel definition exit name. This is not valid on z/OS.

MQCA_CLUSTER_WORKLOAD_DATA

Data passed to the cluster workload exit.

MQCA_CLUSTER_WORKLOAD_EXIT	Name of the cluster workload exit.
MQCA_COMMAND_INPUT_Q_NAME	System command input queue name.
MQCA_DEAD_LETTER_Q_NAME	Name of dead-letter queue.
MQCA_DEF_XMIT_Q_NAME	Default transmission queue name.
MQCA_DNS_GROUP	The name of the group that the TCP listener handling inbound transmissions for the queue-sharing group should join when using Workload Manager for Dynamic Domain Name Services support (DDNS). This is valid on z/OS only.
MQCA_IGQ_USER_ID	Intra-group queuing user identifier. This parameter is valid on z/OS only.
MQCA_LU_GROUP_NAME	Generic LU name for the LU 6.2 listener. This is valid on z/OS only.
MQCA_LU_NAME	LU name to use for outbound LU 6.2 transmissions. This is valid on z/OS only.
MQCA_LU62_ARM_SUFFIX	APPCPM suffix. This is valid on z/OS only.
MQCA_Q_MGR_DESC	Queue manager description.
MQCA_Q_MGR_IDENTIFIER	Internally generated unique queue manager name.
MQCA_Q_MGR_NAME	Name of local queue manager.
MQCA_QSG_NAME	Queue sharing group name. This parameter attribute is valid on z/OS only.
MQCA_REPOSITORY_NAME	Cluster name for the queue manager repository.
MQCA_REPOSITORY_NAMELIST	Name of the list of clusters for which the queue manager is providing a repository manager service.
MQCA_SSL_CRL_NAMELIST	SSL Certification Revocation List (CRL) namelist.
MQCA_SSL_CRYPTO_HARDWARE	Parameters to configure the SSL cryptographic hardware. This parameter is supported on UNIX and Windows platforms only.
MQCA_SSL_KEY_REPOSITORY	Location and name of the SSL key repository.
MQCA_TCP_NAME	Name of the TCP/IP system that you are using. This is valid on z/OS only.

MQIA_ACCOUNTING_CONN_OVERRIDE

Whether the settings of the *MQIAccounting* and *QueueAccounting* queue manager parameters may be overridden. This is valid only on AIX, HP-UX, Linux, i5/OS, Solaris, and Windows.

MQIA_ACCOUNTING_INTERVAL

Intermediate accounting data collection interval. This is valid only on AIX, HP-UX, Linux, i5/OS, Solaris, and Windows.

MQIA_ACCOUNTING_MQI

Whether accounting information is to be collected for MQI data. This is valid only on AIX, HP-UX, Linux, i5/OS, Solaris, and Windows.

MQIA_ACCOUNTING_Q

Accounting data collection for queues.

MQIA_ACTIVE_CHANNELS

Maximum number of channels that can be active at any time. This is valid on z/OS only.

MQIA_ACTIVITY_RECORDING

Whether activity reports can be generated.

MQIA_ADOPTNEWMCA_CHECK

Elements checked to determine whether an MCA should be adopted when a new inbound channel is detected with the same name as an MCA that is already active. This is valid on z/OS only.

MQIA_ADOPTNEWMCA_TYPE

Whether an orphaned instance of an MCA should be restarted automatically when a new inbound channel request matching the *AdoptNewMCACheck* parameter is detected. This is valid on z/OS only.

MQIA_AUTHORITY_EVENT

Control attribute for authority events.

MQIA_BRIDGE_EVENT

Control attribute for IMS Bridge events. This is valid only on z/OS.

MQIA_CHANNEL_AUTO_DEF

Control attribute for automatic channel definition. This is not valid on z/OS.

MQIA_CHANNEL_AUTO_DEF_EVENT

Control attribute for automatic channel definition events. This is not valid on z/OS.

MQIA_CHANNEL_EVENT

Control attribute for channel events.

MQIA_CHINIT_ADAPTERS

Number of adapter subtasks to use for processing WebSphere MQ calls. This is valid on z/OS only.

MQIA_CHINIT_CONTROL

Start channel initiator automatically when queue manager starts.

MQIA_CHINIT_DISPATCHERS

Number of dispatchers to use for the channel initiator. This is valid on z/OS only.

MQIA_CHINIT_SERVICE_PARM

Reserved for use by IBM. This is valid only on z/OS.

MQIA_CHINIT_TRACE_AUTO_START

Whether the channel initiator trace should start automatically. This is valid on z/OS only.

MQIA_CHINIT_TRACE_TABLE_SIZE

Size, in megabytes, of the channel initiator's trace data space. This is valid on z/OS only.

MQIA_CLUSTER_WORKLOAD_LENGTH

Maximum length of the message passed to the cluster workload exit.

MQIA_CLWL_MRU_CHANNELS

Cluster workload most recently used channels.

MQIA_CLWL_USEQ

Cluster workload remote queue use.

MQIA_CMD_SERVER_CONTROL

Start command server automatically when queue manager starts.

MQIA_CODED_CHAR_SET_ID

Coded character set identifier.

MQIA_COMMAND_EVENT

Control attribute for command events. This parameter is valid on z/OS only.

MQIA_COMMAND_LEVEL

Command level supported by queue manager.

MQIA_CONFIGURATION_EVENT

Control attribute for configuration events. This parameter is valid on z/OS only.

MQIA_CPI_LEVEL

Reserved for use by IBM.

MQIA_DIST_LISTS

Distribution list support. This parameter is not valid on z/OS.

MQIA_DNS_WLM

Whether the TCP listener that handles inbound transmissions for the queue-sharing group should register with Workload Manager (WLM) for DDNS. This is valid on z/OS only.

MQIA_EXPIRY_INTERVAL

Expiry interval. This parameter is valid on z/OS only.

MQIA_IGQ_PUT_AUTHORITY

Intra-group queuing put authority. This parameter is valid on z/OS only.

MQIA_INHIBIT_EVENT

Control attribute for inhibit events.

MQIA_INTRA_GROUP_QUEUING

Intra-group queuing support. This parameter is valid on z/OS only.

MQIA_IP_ADDRESS_VERSION

IP address version selector.

MQIA_LISTENER_TIMER

Listener restart interval. This is valid on z/OS only.

MQIA_LOCAL_EVENT	Control attribute for local events.
MQIA_LOGGER_EVENT	Control attribute for recovery log events.
MQIA_LU62_CHANNELS	Maximum number of LU 6.2 channels. This is valid on z/OS only.
MQIA_MAX_CHANNELS	Maximum number of channels that can be current. This is valid on z/OS only.
MQIA_MAX_HANDLES	Maximum number of handles.
MQIA_MAX_MSG_LENGTH	Maximum message length.
MQIA_MAX_PRIORITY	Maximum priority.
MQIA_MAX_UNCOMMITTED_MSGS	Maximum number of uncommitted messages within a unit of work.
MQIA_MONITORING_AUTO_CLUSSDR	Default value of the <i>ChannelMonitoring</i> attribute of automatically defined cluster-sender channels.
MQIA_MONITORING_CHANNEL	Whether channel monitoring is enabled.
MQIA_MONITORING_Q	Whether queue monitoring is enabled.
MQIA_OUTBOUND_PORT_MAX	Maximum value in the range for the binding of outgoing channels. This is valid on z/OS only.
MQIA_OUTBOUND_PORT_MIN	Minimum value in the range for the binding of outgoing channels. This is valid on z/OS only.
MQIA_PERFORMANCE_EVENT	Control attribute for performance events.
MQIA_PLATFORM	Platform on which the queue manager resides.
MQIA_RECEIVE_TIMEOUT	How long a TCP/IP channel waits to receive data from its partner. This is valid on z/OS only.
MQIA_RECEIVE_TIMEOUT_MIN	Minimum length of time that a TCP/IP channel waits to receive data from its partner. This is valid on z/OS only.
MQIA_RECEIVE_TIMEOUT_TYPE	Qualifier to apply to the <i>ReceiveTimeout</i> parameter. This is valid on z/OS only.
MQIA_REMOTE_EVENT	Control attribute for remote events.

MQIA_SHARED_Q_Q_MGR_NAME

When a queue manager makes an MQOPEN call for a shared queue and the queue manager that is specified in the *ObjectQmgrName* parameter of the MQOPEN call is in the same queue-sharing group as the processing queue manager, the SQQMNAME attribute specifies whether the *ObjectQmgrName* is used or whether the processing queue manager opens the shared queue directly. This is valid on z/OS only.

MQIA_SSL_EVENT

Control attribute for SSL events.

MQIA_SSL_FIPS_REQUIRED

Whether only FIPS-certified algorithms are to be used if cryptography is executed in WebSphere MQ itself. This is not valid on z/OS.

MQIA_SSL_RESET_COUNT

SSL key reset count.

MQIA_SSL_TASKS

SSL tasks. This parameter is valid on z/OS only.

MQIA_START_STOP_EVENT

Control attribute for start stop events.

MQIA_STATISTICS_AUTOCLUSSDR

Whether statistics data is to be collected for auto-defined cluster-sender channels and, if so, the rate of data collection. This is valid only on AIX, HP-UX, Linux, i5/OS, Solaris, and Windows.

MQIA_STATISTICS_CHANNEL

Whether statistics monitoring data is to be collected for channels and, if so, the rate of data collection. This is valid only on AIX, HP-UX, Linux, i5/OS, Solaris, and Windows.

MQIA_STATISTICS_INTERVAL

Statistics data collection interval. This is valid only on AIX, HP-UX, Linux, i5/OS, Solaris, and Windows.

MQIA_STATISTICS_MQI

Whether statistics monitoring data is to be collected for the queue manager. This is valid only on AIX, HP-UX, Linux, i5/OS, Solaris, and Windows.

MQIA_STATISTICS_Q

Whether statistics monitoring data is to be collected for queues. This is valid only on AIX, HP-UX, Linux, i5/OS, Solaris, and Windows.

MQIA_SYNCPOINT

Syncpoint availability.

MQIA_TCP_CHANNELS

Maximum number of channels that can be current, or clients that can be connected, that use the TCP/IP transmission protocol. This is valid on z/OS only.

MQIA_TCP_KEEP_ALIVE

Whether the TCP KEEPALIVE facility is to be used to check whether the other end of a connection is still available. This is valid on z/OS only.

MQIA_TCP_STACK_TYPE

Whether the channel initiator may use only the TCP/IP address space

specified in the *TCPName* parameter, or may optionally bind to any selected TCP/IP address. This is valid on z/OS only.

MQIA_TRACE_ROUTE_RECORDING

Whether trace-route information can be recorded and reply messages generated.

MQIA_TRIGGER_INTERVAL

Trigger interval.

MQIACF_Q_MGR_CLUSTER

All clustering attributes. These are:

- MQCA_CLUSTER_WORKLOAD_DATA
- MQCA_CLUSTER_WORKLOAD_EXIT
- MQCA_CHANNEL_AUTO_DEF_EXIT
- MQCA_REPOSITORY_NAME
- MQCA_REPOSITORY_NAMELIST
- MQIA_CLUSTER_WORKLOAD_LENGTH
- MQIA_CLWL_MRU_CHANNELS
- MQIA_CLWL_USEQ
- MQIA_MONITORING_AUTOCLUSSDR
- MQCA_Q_MGR_IDENTIFIER

MQIACF_Q_MGR_DQM

All distributed queuing attributes. These are:

- MQCA_CHANNEL_AUTO_DEF_EXIT
- MQCA_DEAD_LETTER_Q_NAME
- MQCA_DEF_XMIT_Q_NAME
- MQCA_DNS_GROUP
- MQCA_IGQ_USER_ID
- MQCA LU GROUP NAME
- MQCA LU NAME
- MQCA LU62 ARM SUFFIX
- MQCA_Q_MGR_IDENTIFIER
- MQCA_SSL_CRL_NAMELIST
- MQCA_SSL_CRYPTO_HARDWARE
- MQCA_SSL_KEY_REPOSITORY
- MQCA_TCP_NAME
- MQIA_ACTIVE_CHANNELS
- MQIA_ADOPTNEWMCA_CHECK
- MQIA_ADOPTNEWMCA_TYPE
- MQIA_CHANNEL_AUTO_DEF
- MQIA_CHANNEL_AUTO_DEF_EVENT
- MQIA_CHANNEL_EVENT
- MQIA_CHINIT_ADAPTERS
- MQIA_CHINIT_CONTROL
- MQIA_CHINIT_DISPATCHERS
- MQIA_CHINIT_SERVICE_PARM
- MQIA_CHINIT_TRACE_AUTO_START

- MQIA_CHINIT_TRACE_TABLE_SIZE
- MQIA_INTRA_GROUP_QUEUING
- MQIA_IGQ_PUT_AUTHORITY
- MQIA_IP_ADDRESS_VERSION
- MQIA_LISTENER_TIMER
- MQIA_LU62_CHANNELS
- MQIA_MAX_CHANNELS
- MQIA_MONITORING_CHANNEL
- MQIA_OUTBOUND_PORT_MAX
- MQIA_OUTBOUND_PORT_MIN
- MQIA_RECEIVE_TIMEOUT
- MQIA_RECEIVE_TIMEOUT_MIN
- MQIA_RECEIVE_TIMEOUT_TYPE
- MQIA_SSL_EVENT
- MQIA_SSL_FIPS_REQUIRED
- MQIA_SSL_RESET_COUNT
- MQIA_SSL_TASKS
- MQIA_STATISTICS_AUTO_CLUSSDR
- MQIA_TCP_CHANNELS
- MQIA_TCP_KEEP_ALIVE
- MQIA_TCP_STACK_TYPE

MQIACF_Q_MGR_EVENT

All event control attributes. These are:

- MQIA_AUTHORITY_EVENT
- MQIA_BRIDGE_EVENT
- MQIA_CHANNEL_EVENT
- MQIA_COMMAND_EVENT
- MQIA_CONFIGURATION_EVENT
- MQIA_INHIBIT_EVENT
- MQIA_LOCAL_EVENT
- MQIA_LOGGER_EVENT
- MQIA_PERFORMANCE_EVENT
- MQIA_REMOTE_EVENT
- MQIA_SSL_EVENT
- MQIA_START_STOP_EVENT

MQIACF_Q_MGR_SYSTEM

All queue manager system attributes. These are:

- MQCA_COMMAND_INPUT_Q_NAME
- MQCA_DEAD_LETTER_Q_NAME
- MQCA_Q_MGR_NAME
- MQCA_QSG_NAME
- MQIA_ACCOUNTING_CONN_OVERRIDE
- MQIA_ACCOUNTING_INTERVAL
- MQIA_ACCOUNTING_Q
- MQIA_ACTIVITY_RECORDING

- MQCA_ALTERATION_DATE
- MQCA_ALTERATION_TIME
- MQIA_CMD_SERVER_CONTROL
- MQIA_CODED_CHAR_SET_ID
- MQIA_COMMAND_LEVEL
- MQIA_CPI_LEVEL
- MQIA_DIST_LISTS
- MQIA_EXPIRY_INTERVAL
- MQIA_MAX_HANDLES
- MQIA_MAX_MSG_LENGTH
- MQIA_MAX_PRIORITY
- MQIA_MAX_UNCOMMITTED_MSGS
- MQIA_MONITORING_Q
- MQIA_PLATFORM
- MQIA_SHARED_Q_Q_MGR_NAME
- MQIA_STATISTICS_INTERVAL
- MQIA_STATISTICS_MQI
- MQIA_STATISTICS_Q
- MQIA_SYNCPOINT
- MQIA_TRACE_ROUTE_RECORDING
- MQIA_TRIGGER_INTERVAL

Inquire Queue Manager (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The response to the Inquire Queue Manager (MQCMD_INQUIRE_Q_MGR) command consists of the response header followed by the *QMgrName* structure and the requested combination of attribute parameter structures.

Always returned:

QMgrName

Returned if requested:

*AccountingConnOverride, AccountingInterval, ActivityRecording,
AdoptNewMCACheck, AdoptNewMCAType, AlterationDate, AlterationTime,
AuthorityEvent, BridgeEvent, ChannelAutoDef, ChannelAutoDefEvent,
ChannelAutoDefExit, ChannelEvent, ChannelInitiatorControl,
ChannelMonitoring, ChannelStatistics, ChinitAdapters,
ChinitDispatchers, ChinitServiceParm, ChinitTraceAutoStart,
ChinitTraceTableSize, ClusterSenderMonitoringDefault,
ClusterSenderStatistics, ClusterWorkloadData, ClusterWorkloadExit,
ClusterWorkloadLength, CLWLMRUCHannels, CLWLUseQ, CodedCharSetId,
CommandEvent, CommandInputQName, CommandLevel, CommandServerControl,
ConfigurationEvent, DeadLetterQName, DefXmitQName, DistLists, DNSGroup,
DNSWLM, ExpiryInterval, IGQPPutAuthority, IGQUserId), InhibitEvent,
IntraGroupQueuing, IPAddressVersion, ListenerTimer, LocalEvent,*

*LoggerEvent, LUGroupName, LUName, LU62ARMSuffix, LU62Channels,
MaxChannels, MaxActiveChannels, MaxHandles, MaxMsgLength, MaxPriority,
MaxUncommittedMsgs, MQIAccounting, MQIStatisticsOutboundPortMax,
OutboundPortMin, PerformanceEvent, Platform, QmgrDesc, QMgrIdentifier, ,
QSGName, QueueAccounting, QueueMonitoring, QueueStatistics,
ReceiveTimeout), ReceiveTimeoutMin, ReceiveTimeoutType, RemoteEvent,
RepositoryName, RepositoryNamelist, SharedQQmgrName, SSLCRLNamelist,
SSLCryptoHardware, SSLEvent, SSLFIPSRequired, SSLKeyRepository,
SSLKeyResetCount, SSLTasks, StartStopEvent, StatisticsInterval,
SyncPoint, TCPChannels, TCPKeepAlive, TCPName, TCPStackType,
TraceRouteRecording, TriggerInterval*

Response data

AccountingConnOverride (MQCFIN)

Specifies whether applications can override the settings of the *QueueAccounting* and *MQIAccounting* queue manager parameters (parameter identifier: **MQIA_ACCOUNTING_CONN_OVERRIDE**).

The value can be:

MQMON_DISABLED

Applications cannot override the settings of the *QueueAccounting* and *MQIAccounting* parameters.

MQMON_ENABLED

Applications can override the settings of the *QueueAccounting* and *MQIAccounting* parameters by using the options field of the MQCNO structure of the MQCONNX API call.

This parameter applies only to AIX, HP-UX, Linux, i5/OS, Solaris, and Windows.

AccountingInterval (MQCFIN)

The time interval, in seconds, at which intermediate accounting records are written (parameter identifier: **MQIA_ACCOUNTING_INTERVAL**).

It is a value in the range 1 through 604 000.

This parameter applies only to AIX, HP-UX, Linux, i5/OS, Solaris, and Windows.

ActivityRecording (MQCFIN)

Whether activity reports can be generated (parameter identifier: **MQIA_ACTIVITY_RECORDING**).

The value can be:

MQRECORDING_DISABLED

Activity reports cannot be generated.

MQRECORDING_MSG

Activity reports can be generated and sent to the destination specified by the originator of the message causing the report.

MQRECORDING_Q

Activity reports can be generated and sent to SYSTEM.ADMIN.ACTIVITY.QUEUE.

AdoptNewMCACheck (MQCFIN)

The elements checked to determine whether an MCA should be adopted

(restarted) when a new inbound channel is detected that has the same name as a currently active MCA (parameter identifier: MQIA_ADOPTNEWMCA_CHECK).

The value can be:

MQADOPT_CHECK_Q_MGR_NAME

Check the queue manager name.

MQADOPT_CHECK_NET_ADDR

Check the network address.

MQADOPT_CHECK_ALL

Check the queue manager name and network address.

MQADOPT_CHECK_NONE

Do not check any elements.

This parameter applies to z/OS only.

AdoptNewMCAType (MQCFIL)

Adoption of orphaned channel instances (parameter identifier: MQIA_ADOPTNEWMCA_TYPE).

The value can be:

MQADOPT_TYPE_NO

Do not adopt orphaned channel instances.

MQADOPT_TYPE_ALL

Adopt all channel types.

This parameter applies to z/OS only.

AlterationDate (MQCFST)

Alteration date (parameter identifier: MQCA_ALTERATION_DATE).

The date, in the form yyyy-mm-dd, on which the information was last altered.

AlterationTime (MQCFST)

Alteration time (parameter identifier: MQCA_ALTERATION_TIME).

The time, in the form hh.mm.ss, at which the information was last altered.

AuthorityEvent (MQCFIN)

Controls whether authorization (Not Authorized) events are generated (parameter identifier: MQIA_AUTHORITY_EVENT).

The value can be:

MQEVR_DISABLED

Event reporting disabled.

MQEVR_ENABLED

Event reporting enabled.

BridgeEvent (MQCFIN)

Controls whether IMS Bridge events are generated (parameter identifier: MQIA_BRIDGE_EVENT). This parameter applies to z/OS only.

The value can be:

MQEVR_DISABLED

Event reporting disabled.

MQEVR_ENABLED

Event reporting enabled.

ChannelAutoDef (MQCFIN)

Controls whether receiver and server-connection channels can be auto-defined (parameter identifier: MQIA_CHANNEL_AUTO_DEF).

The value can be:

MQCHAD_DISABLED

Channel auto-definition disabled.

MQCHAD_ENABLED

Channel auto-definition enabled.

ChannelAutoDefEvent (MQCFIN)

Controls whether channel auto-definition events are generated (parameter identifier: MQIA_CHANNEL_AUTO_DEF_EVENT), when a receiver, server-connection, or cluster-sender channel is auto-defined.

The value can be:

MQEVR_DISABLED

Event reporting disabled.

MQEVR_ENABLED

Event reporting enabled.

ChannelAutoDefExit (MQCFST)

Channel auto-definition exit name (parameter identifier: MQCA_CHANNEL_AUTO_DEF_EXIT).

The maximum length of the exit name depends on the environment in which the exit is running. MQ_EXIT_NAME_LENGTH gives the maximum length for the environment in which your application is running.

MQ_MAX_EXIT_NAME_LENGTH gives the maximum for all supported environments.

ChannelEvent (MQCFIN)

Controls whether channel events are generated (parameter identifier: MQIA_CHANNEL_EVENT).

The value can be:

MQEVR_DISABLED

Event reporting disabled.

MQEVR_ENABLED

Event reporting enabled.

MQEVR_EXCEPTION

Reporting of exception channel events enabled.

ChannelInitiatorControl (MQCFIN)

Start the channel initiator during queue manager start (parameter identifier: MQIA_CHINIT_CONTROL). This parameter is not available on z/OS.

The value can be:

MQSVC_CONTROL_MANUAL

The channel initiator is not to be started automatically when the queue manager starts.

MQSVC_CONTROL_Q_MGR

The channel initiator is to be started automatically when the queue manager starts.

ChannelMonitoring (MQCFIN)

Default setting for online monitoring for channels (parameter identifier: MQIA_MONITORING_CHANNEL).

If the *ChannelMonitoring* channel attribute is set to MQMON_Q_MGR, this attribute specifies the value which is assumed by the channel. The value can be:

MQMON_OFF

Online monitoring data collection is turned off.

MQMON_NONE

Online monitoring data collection is turned off for channels regardless of the setting of their *ChannelMonitoring* attribute.

MQMON_LOW

Online monitoring data collection is turned on, with a low ratio of data collection.

MQMON_MEDIUM

Online monitoring data collection is turned on, with a moderate ratio of data collection.

MQMON_HIGH

Online monitoring data collection is turned on, with a high ratio of data collection.

ChannelStatistics (MQCFIN)

Whether statistics data is to be collected for channels (parameter identifier: MQIA_STATISTICS_CHANNEL).

The value can be:

MQMON_NONE

Statistics data collection is turned off for channels regardless of the setting of their *ChannelStatistics* parameter. This is the queue manager's initial default value.

MQMON_OFF

Statistics data collection is turned off for channels specifying a value of MQMON_Q_MGR in their *ChannelStatistics* parameter.

MQMON_LOW

Statistics data collection is turned on, with a low ratio of data collection, for channels specifying a value of MQMON_Q_MGR in their *ChannelStatistics* parameter.

MQMON_MEDIUM

Statistics data collection is turned on, with a moderate ratio of data collection, for channels specifying a value of MQMON_Q_MGR in their *ChannelStatistics* parameter.

MQMON_HIGH

Statistics data collection is turned on, with a high ratio of data collection, for channels specifying a value of MQMON_Q_MGR in their *ChannelStatistics* parameter.

This parameter is valid only on AIX, HP-UX, Linux, i5/OS, Solaris, and Windows.

ChinitAdapters (MQCFIN)

Number of adapter subtasks (parameter identifier: MQIA_CHINIT_ADAPTERS).

The number of adapter subtasks to use for processing WebSphere MQ calls.
This parameter applies to z/OS only.

ChinitDispatchers (MQCFIN)

Number of dispatchers (parameter identifier: MQIA_CHINIT_DISPATCHERS).

The number of dispatchers to use for the channel initiator. This parameter applies to z/OS only.

ChinitServiceParm (MQCFST)

Reserved for use by IBM (parameter identifier: MQCA_CHINIT_SERVICE_PARM).

ChinitTraceAutoStart (MQCFIN)

Whether the channel initiator trace should start automatically (parameter identifier: MQIA_CHINIT_TRACE_AUTO_START).

The value can be:

MQTRAXSTR_YES

Channel initiator trace is to start automatically.

MQTRAXSTR_NO

Channel initiator trace is not to start automatically.

This parameter applies to z/OS only.

ChinitTraceTableSize (MQCFIN)

The size, in megabytes, of the channel initiator's trace data space (parameter identifier: MQIA_CHINIT_TRACE_TABLE_SIZE).

This parameter applies to z/OS only.

ClusterSenderMonitoringDefault (MQCFIN)

Setting for online monitoring for automatically defined cluster-sender channels (parameter identifier: MQIA_MONITORING_AUTO_CLUSSDR).

The value can be:

MQMON_Q_MGR

Collection of online monitoring data is inherited from the setting of the queue manager's *ChannelMonitoring* parameter.

MQMON_OFF

Monitoring for the channel is switched off.

MQMON_LOW

Specifies a low rate of data collection with a minimal impact on system performance unless *ChannelMonitoring* for the queue manager is MQMON_NONE. The data collected is not likely to be the most current.

MQMON_MEDIUM

Specifies a moderate rate of data collection with limited impact on system performance unless *ChannelMonitoring* for the queue manager is MQMON_NONE..

MQMON_HIGH

Specifies a high rate of data collection with a likely impact on system

performance unless *ChannelMonitoring* for the queue manager is MQMON_NONE.. The data collected is the most current available.

ClusterSenderStatistics (MQCFIN)

Whether statistics data is to be collected for auto-defined cluster-sender channels (parameter identifier: MQIA_STATISTICS_AUTO_CLUSSDR).

The value can be:

MQMON_Q_MGR

Collection of statistics data is inherited from the setting of the queue manager's *ChannelStatistics* parameter.

MQMON_OFF

Statistics data collection for the channel is switched off.

MQMON_LOW

Specifies a low rate of data collection with a minimal impact on system performance.

MQMON_MEDIUM

Specifies a moderate rate of data collection.

MQMON_HIGH

Specifies a high rate of data collection.

This parameter is valid only on AIX, HP-UX, Linux, i5/OS, Solaris, and Windows.

ClusterWorkLoadData (MQCFST)

Data passed to the cluster workload exit (parameter identifier: MQCA_CLUSTER_WORKLOAD_DATA).

ClusterWorkLoadExit (MQCFST)

Name of the cluster workload exit (parameter identifier: MQCA_CLUSTER_WORKLOAD_EXIT).

The maximum length of the exit name depends on the environment in which the exit is running. MQ_EXIT_NAME_LENGTH gives the maximum length for the environment in which your application is running.

MQ_MAX_EXIT_NAME_LENGTH gives the maximum for all supported environments.

ClusterWorkLoadLength (MQCFIN)

Cluster workload length (parameter identifier: MQIA_CLUSTER_WORKLOAD_LENGTH).

The maximum length of the message passed to the cluster workload exit.

CLWLMRUCHannels (MQCFIN)

Cluster workload most recently used (MRU) channels (parameter identifier: MQIA_CLWL_MRU_CHANNELS).

The maximum number of active most recently used outbound channels.

CLWLUseQ (MQCFIN)

Use of remote queue (parameter identifier: MQIA_CLWL_USEQ).

Specifies whether a cluster queue manager is to use remote puts to other queues defined in other queue managers within the cluster during workload management.

The value can be:

MQCLWL_USEQ_ANY

Use remote queues.

MQCLWL_USEQ_LOCAL

Do not use remote queues.

CodedCharSetId (MQCFIN)

Coded character set identifier (parameter identifier: MQIA_CODED_CHAR_SET_ID).

CommandEvent (MQCFIN)

Controls whether command events are generated (parameter identifier: MQIA_COMMAND_EVENT). This parameter applies to z/OS only.

The value can be:

MQEVR_DISABLED

Event reporting disabled.

MQEVR_ENABLED

Event reporting enabled.

MQEVR_NODISPLAY

Event reporting enabled for all successful commands except Inquire commands.

CommandInputQName (MQCFST)

Command input queue name (parameter identifier: MQCA_COMMAND_INPUT_Q_NAME).

The maximum length of the string is MQ_Q_NAME_LENGTH.

CommandLevel (MQCFIN)

Command level supported by queue manager (parameter identifier: MQIA_COMMAND_LEVEL).

The value can be:

MQCMDL_LEVEL_1

Level 1 of system control commands.

This value is returned by the following:

- MQSeries for AIX V2.2
- MQSeries for OS/2® V2.0
- MQSeries for OS/400®:
 - V2R3
 - V3R1
 - V3R6
- MQSeries for Windows V2.0

MQCMDL_LEVEL_101

MQSeries for Windows V2.0.1

MQCMDL_LEVEL_110

MQSeries for Windows V2.1

MQCMDL_LEVEL_200

MQSeries for Windows NT® V2.0

MQCMDL_LEVEL_201

MQSeries for OS/2 V2.0.1

MQCMDL_LEVEL_220

Level 220 of system control commands.

This value is returned by the following:

- MQSeries for AT&T GIS UNIX V2.2
- MQSeries for SINIX and DC/OSx V2.2
- MQSeries for Compaq NonStop Kernel V2.2.0.1

MQCMDL_LEVEL_221

Level 221 of system control commands.

This value is returned by the following:

- MQSeries for AIX Version 2.2.1
- MQSeries for DIGITAL UNIX (Compaq Tru64 UNIX) V2.2.1

MQCMDL_LEVEL_320

MQSeries for OS/400 V3R2 and V3R7

MQCMDL_LEVEL_420

MQSeries for AS/400® V4R2 and R2.1

MQCMDL_LEVEL_500

Level 500 of system control commands.

This value is returned by the following:

- MQSeries for AIX V5.0
- MQSeries for HP-UX V5.0
- MQSeries for OS/2 Warp V5.0
- MQSeries for Solaris V5.0
- MQSeries for Windows NT V5.0

MQCMDL_LEVEL_510

Level 510 of system control commands.

This value is returned by the following:

- MQSeries for AIX V5.1
- MQSeries for AS/400 V5.1
- MQSeries for HP-UX V5.1
- MQSeries for OS/2 Warp V5.1
- MQSeries for Compaq Tru64 UNIX, V5.1
- MQSeries for Compaq OpenVMS Alpha, Version 5.1
- MQSeries for Compaq NonStop Kernel, V5.1
- MQSeries for Solaris V5.1
- MQSeries for Windows NT V5.1

MQCMDL_LEVEL_520

Level 520 of system control commands.

This value is returned by the following:

- MQSeries for AIX V5.2
- MQSeries for AS/400 V5.2
- MQSeries for HP-UX V5.2
- MQSeries for Linux® V5.2
- MQSeries for Solaris V5.2
- MQSeries for Windows NT V5.2

- MQSeries for Windows 2000 V5.2

MQCMDL_LEVEL_530

Level 530 of system control commands.

This value is returned by the following:

- WebSphere MQ for AIX, V5.3
- WebSphere MQ for iSeries, V5.3
- WebSphere MQ for HP-UX, V5.3
- WebSphere MQ for Linux, V5.3
- WebSphere MQ for Sun Solaris, Version 5.3
- WebSphere MQ for Windows NT and Windows 2000, Version 5.3

MQCMDL_LEVEL_531

Level 531 of system control commands.

MQCMDL_LEVEL_600

Level 600 of system control commands.

The set of system control commands that corresponds to a particular value of the *CommandLevel* attribute varies according to the value of the *Platform* attribute; both must be used to decide which system control commands are supported.

CommandServerControl (MQCFIN)

Start the command server during queue manager start (parameter identifier: MQIA_CMD_SERVER_CONTROL). This parameter is not available on z/OS.

The value can be:

MQSVC_CONTROL_MANUAL

The command server is not to be started automatically when the queue manager starts.

MQSVC_CONTROL_Q_MGR

The command server is to be started automatically when the queue manager starts.

ConfigurationEvent (MQCFIN)

Queue sharing group name (parameter identifier: MQIA_CONFIGURATION_EVENT). This parameter is valid only on z/OS.

The value can be:

MQEVR_DISABLED

Event reporting disabled.

MQEVR_ENABLED

Event reporting enabled.

DeadLetterQName (MQCFST)

Dead letter (undelivered message) queue name (parameter identifier: MQCA_DEAD_LETTER_Q_NAME).

Specifies the name of the local queue that is to be used for undelivered messages. Messages are put on this queue if they cannot be routed to their correct destination.

The maximum length of the string is MQ_Q_NAME_LENGTH.

DefXmitQName (**MQCFST**)

Default transmission queue name (parameter identifier: MQCA_DEF_XMIT_Q_NAME).

This is the name of the default transmission queue that is used for the transmission of messages to remote queue managers, if there is no other indication of which transmission queue to use.

The maximum length of the string is MQ_Q_NAME_LENGTH.

DistLists (**MQCFIN**)

Distribution list support (parameter identifier: MQIA_DIST_LISTS).

The value can be:

MQDL_SUPPORTED

Distribution lists supported.

MQDL_NOT_SUPPORTED

Distribution lists not supported.

DNSGroup (**MQCFST**)

DNS group name (parameter identifier: MQCA_DNS_GROUP).

The name of the group that the TCP listener handling inbound transmissions for the queue-sharing group should join when using Workload Manager for Dynamic Domain Name Services support (DDNS). This parameter applies to z/OS only.

DNSWLM (**MQCFIN**)

Controls whether the TCP listener that handles inbound transmissions for the queue-sharing group should register with Workload Manager (WLM) for DDNS: (parameter identifier: MQIA_DNS_WLM).

The value can be:

MQDNSWLM_YES

The listener should register with WLM.

MQDNSWLM_NO

The listener is not to register with WLM. This is the queue manager's initial default value.

This parameter applies to z/OS only.

ExpiryInterval (**MQCFIN**)

Interval between scans for expired messages (parameter identifier: MQIA_EXPIRY_INTERVAL). This parameter is valid only on z/OS.

Specifies the frequency with which the queue manager scans the queues looking for expired messages. This is a time interval in seconds in the range 1 through 99 999 999, or the following special value:

MQEXPI_OFF

No scans for expired messages.

IGQPutAuthority (**MQCFIN**)

Type of authority checking used by the intra-group queuing agent (parameter identifier: MQIA_IGQ_PUT_AUTHORITY). This parameter is valid only on z/OS.

The attribute indicates the type of authority checking that is performed when the local intra-group queuing agent (IGQ agent) removes a message from the shared transmission queue and places the message on a local queue. The value can be:

MQIGQPA_DEFAULT

Default user identifier is used.

MQIGQPA_CONTEXT

Context user identifier is used.

MQIGQPA_ONLY_IGQ

Only the IGQ user identifier is used.

MQIGQPA_ALTERNATE_OR_IGQ

Alternate user identifier or IGQ-agent user identifier is used.

IGQUserId (**MQCFST**)

Use identifier used the intra-group queuing agent (parameter identifier: MQCA_IGQ_USER_ID). This parameter is valid only on z/OS.

The maximum length of the string is MQ_USER_ID_LENGTH.

InhibitEvent (**MQCFIN**)

Controls whether inhibit (Inhibit Get and Inhibit Put) events are generated (parameter identifier: MQIA_INHIBIT_EVENT).

The value can be:

MQEVR_DISABLED

Event reporting disabled.

MQEVR_ENABLED

Event reporting enabled.

IntraGroupQueuing (**MQCFIN**)

Specifies whether intra-group queuing is used (parameter identifier: MQIA_INTRA_GROUP_QUEUING). This parameter is valid only on z/OS.

The value can be:

MQIGQ_DISABLED

Intra-group queuing is disabled. All messages destined for other queue managers in the queue-sharing group are transmitted using conventional channels.

MQIGQ_ENABLED

Intra-group queuing is enabled.

IPAddressVersion (**MQCFIN**)

IP address version selector (parameter identifier: MQIA_IP_ADDRESS_VERSION).

Specifies which IP address version, either IPv4 or IPv6, is used. The value can be:

MQIPADDR_IPV4

IPv4 is used.

MQIPADDR_IPV6

IPv6 is used.

ListenerTimer (**MQCFIN**)

Listener restart interval (parameter identifier: MQIA_LISTENER_TIMER).

The time interval, in seconds, between attempts by WebSphere MQ to restart the listener after an APPC or TCP/IP failure. This parameter applies to z/OS only.

LocalEvent (**MQCFIN**)

Controls whether local error events are generated (parameter identifier: MQIA_LOCAL_EVENT).

The value can be:

MQEVR_DISABLED

Event reporting disabled.

MQEVR_ENABLED

Event reporting enabled.

LoggerEvent (**MQCFIN**)

Controls whether recovery log events are generated (parameter identifier: MQIA_LOGGER_EVENT).

The value can be:

MQEVR_DISABLED

Event reporting disabled.

MQEVR_ENABLED

Event reporting enabled.

This is valid only on AIX, HP-UX, i5/OS, Solaris, Linux, and Windows.

LUGroupName (**MQCFST**)

Generic LU name for the LU 6.2 listener (parameter identifier: MQCA_LU_GROUP_NAME).

The generic LU name to be used by the LU 6.2 listener that handles inbound transmissions for the queue-sharing group. This parameter applies to z/OS only.

LUName (**MQCFST**)

LU name to use for outbound LU 6.2 transmissions (parameter identifier: MQCA_LU_NAME).

The name of the LU to use for outbound LU 6.2 transmissions. This parameter applies to z/OS only.

LU62ARMSuffix (**MQCFST**)

APPCPM suffix (parameter identifier: MQCA_LU62_ARM_SUFFIX).

The suffix of the APPCPM member of SYS1.PARMLIB. This suffix nominates the LUADD for this channel initiator. This parameter applies to z/OS only.

LU62Channels (**MQCFIN**)

Maximum number of LU 6.2 channels (parameter identifier: MQIA_LU62_CHANNELS).

The maximum number of channels that can be current, or clients that can be connected, that use the LU 6.2 transmission protocol. This parameter applies to z/OS only.

MaxActiveChannels (**MQCFIN**)

Maximum number of channels (parameter identifier: MQIA_ACTIVE_CHANNELS).

The maximum number of channels that can be active at any time. This parameter applies to z/OS only.

MaxChannels (MQCFIN)

Maximum number of current channels (parameter identifier: MQIA_MAX_CHANNELS).

The maximum number of channels that can be current (including server-connection channels with connected clients). This parameter applies to z/OS only.

MaxHandles (MQCFIN)

Maximum number of handles (parameter identifier: MQIA_MAX_HANDLES).

Specifies the maximum number of handles that any one connection can have open at the same time.

MaxMsgLength (MQCFIN)

Maximum message length (parameter identifier: MQIA_MAX_MSG_LENGTH).

MaxPriority (MQCFIN)

Maximum priority (parameter identifier: MQIA_MAX_PRIORITY).

MaxUncommittedMsgs (MQCFIN)

Maximum number of uncommitted messages within a unit of work (parameter identifier: MQIA_MAX_UNCOMMITTED_MSGS).

That is:

- The number of messages that can be retrieved, plus
- The number of messages that can be put on a queue, plus
- Any trigger messages generated within this unit of work

under any one syncpoint. This limit does not apply to messages that are retrieved or put outside syncpoint.

MQIAccounting (MQCFIN)

Whether accounting information for MQI data is to be collected (parameter identifier: MQIA_ACCOUNTING_MQI).

The value can be:

MQMON_OFF

MQI accounting data collection is disabled.

MQMON_ON

MQI accounting data collection is enabled.

This parameter is valid only on AIX, HP-UX, Linux, i5/OS, Solaris, and Windows.

MQIStatistics (MQCFIN)

Whether statistics monitoring data is to be collected for the queue manager (parameter identifier: MQIA_STATISTICS_MQI).

The value can be:

MQMON_OFF

Data collection for MQI statistics is disabled. This is the queue manager's initial default value.

MQMON_ON

Data collection for MQI statistics is enabled.

This parameter is valid only on AIX, HP-UX, Linux, i5/OS, Solaris, and Windows.

OutboundPortMax (**MQCFIN**)

The maximum value in the range for the binding of outgoing channels (parameter identifier: MQIA_OUTBOUND_PORT_MAX).

The maximum value in the range of port numbers to be used when binding outgoing channels. This parameter applies to z/OS only.

OutboundPortMin (**MQCFIN**)

The minimum value in the range for the binding of outgoing channels (parameter identifier: MQIA_OUTBOUND_PORT_MIN).

The minimum value in the range of port numbers to be used when binding outgoing channels. This parameter applies to z/OS only.

PerformanceEvent (**MQCFIN**)

Controls whether performance-related events are generated (parameter identifier: MQIA_PERFORMANCE_EVENT).

The value can be:

MQEVR_DISABLED

Event reporting disabled.

MQEVR_ENABLED

Event reporting enabled.

Platform (**MQCFIN**)

Platform on which the queue manager resides (parameter identifier: MQIA_PLATFORM).

The value can be:

MQPL_AIX

AIX (same value as MQPL_UNIX).

MQPL NSK

Compaq NonStop Kernel.

MQPL_OS400

i5/OS.

MQPL_UNIX

UNIX systems.

MQPL_VMS

HP OpenVMS.

MQPL_WINDOWS_NT

Windows.

MQPL_ZOS

z/OS

QmgrDesc (**MQCFST**)

Queue manager description (parameter identifier: MQCA_Q_MGR_DESC).

The maximum length of the string is MQ_Q_MGR_DESC_LENGTH.

QMgrIdentifier (**MQCFST**)

Queue manager identifier (parameter identifier: MQCA_Q_MGR_IDENTIFIER).

The unique identifier of the queue manager.

QMngrName (**MQCFST**)

Name of local queue manager (parameter identifier: MQCA_Q_MGR_NAME).

The maximum length of the string is MQ_Q_MGR_NAME_LENGTH.

QSGName (**MQCFST**)

Queue sharing group name (parameter identifier: MQCA_QSG_NAME). This parameter is valid only on z/OS.

The maximum length of the string is MQ_QSG_NAME_LENGTH.

QueueAccounting (**MQCFIN**)

Collection of accounting (thread-level and queue-level accounting) data for queues (parameter identifier: MQIA_ACCOUNTING_Q).

The value can be:

MQMON_NONE

Accounting data collection for queues is disabled.

MQMON_OFF

Accounting data collection is disabled for queues specifying a value of MQMON_Q_MGR in the *QueueAccounting* parameter.

MQMON_ON

Accounting data collection is enabled for queues specifying a value of MQMON_Q_MGR in the *QueueAccounting* parameter.

QueueMonitoring (**MQCFIN**)

Default setting for online monitoring for queues (parameter identifier: MQIA_MONITORING_Q).

If the *QueueMonitoring* queue attribute is set to MQMON_Q_MGR, this attribute specifies the value which is assumed by the channel. The value can be:

MQMON_OFF

Online monitoring data collection is turned off.

MQMON_NONE

Online monitoring data collection is turned off for queues regardless of the setting of their *QueueMonitoring* attribute.

MQMON_LOW

Online monitoring data collection is turned on, with a low ratio of data collection.

MQMON_MEDIUM

Online monitoring data collection is turned on, with a moderate ratio of data collection.

MQMON_HIGH

Online monitoring data collection is turned on, with a high ratio of data collection.

QueueStatistics (**MQCFIN**)

Whether statistics data is to be collected for queues (parameter identifier: MQIA_STATISTICS_Q).

The value can be:

MQMON_NONE

Statistics data collection is turned off for queues regardless of the setting of their *QueueStatistics* parameter.

MQMON_OFF

Statistics data collection is turned off for queues specifying a value of MQMON_Q_MGR in their *QueueStatistics* parameter.

MQMON_ON

Statistics data collection is turned on for queues specifying a value of MQMON_Q_MGR in their *QueueStatistics* parameter.

This parameter is valid only on AIX, HP-UX, Linux, i5/OS, Solaris, and Windows.

ReceiveTimeout (MQCFIN)

How long a TCP/IP channel waits to receive data from its partner (parameter identifier: MQIA_RECEIVE_TIMEOUT).

The length of time that a TCP/IP channel waits to receive data, including heartbeats, from its partner before returning to the inactive state.

This parameter applies to z/OS only.

ReceiveTimeoutMin (MQCFIN)

The minimum length of time that a TCP/IP channel waits to receive data from its partner (parameter identifier: MQIA_RECEIVE_TIMEOUT_MIN).

The minimum length of time that a TCP/IP channel waits to receive data, including heartbeats, from its partner before returning to the inactive state.

This parameter applies to z/OS only.

ReceiveTimeoutType (MQCFIN)

The qualifier to apply to *ReceiveTimeout* (parameter identifier: MQIA_RECEIVE_TIMEOUT_TYPE).

The qualifier to apply to *ReceiveTimeoutType* to calculate how long a TCP/IP channel waits to receive data, including heartbeats, from its partner before returning to the inactive state. This parameter applies to z/OS only.

The value can be:

MQRCVTIME_MULTIPLY

The *ReceiveTimeout* value is a multiplier to be applied to the negotiated value of *HeartbeatInterval* to determine how long a channel will wait.

MQRCVTIME_ADD

ReceiveTimeout is a value, in seconds, to be added to the negotiated value of *HeartbeatInterval* to determine how long a channel will wait.

MQRCVTIME_EQUAL

ReceiveTimeout is a value, in seconds, representing how long a channel will wait.

RemoteEvent (MQCFIN)

Controls whether remote error events are generated (parameter identifier: MQIA_REMOTE_EVENT).

The value can be:

MQEVR_DISABLED

Event reporting disabled.

MQEVR_ENABLED

Event reporting enabled.

RepositoryName (MQCFST)

Repository name (parameter identifier: MQCA_REPOSITORY_NAME).

The name of a cluster for which this queue manager is to provide a repository service.

***RepositoryNamelist* (MQCFST)**

Repository name list (parameter identifier: MQCA_REPOSITORY_NAMELIST).

The name of a list of clusters for which this queue manager is to provide a repository service.

***SharedQQmgrName* (MQCFIN)**

Shared-queue queue manager name (parameter identifier: MQIA_SHARED_Q_Q_MGR_NAME).

When a queue manager makes an MQOPEN call for a shared queue and the queue manager that is specified in the *ObjectQmgrName* parameter of the MQOPEN call is in the same queue-sharing group as the processing queue manager, the SQQMNAME attribute specifies whether the *ObjectQmgrName* is used or whether the processing queue manager opens the shared queue directly. This parameter is valid only on z/OS.

The value can be:

MQSQQM_USE

ObjectQmgrName is used and the appropriate transmission queue is opened.

MQSQQM_IGNORE

The processing queue manager opens the shared queue directly.

***SSLCRLNamelist* (MQCFST)**

The SSL Certification Revocation List (CRL) namelist (parameter identifier: MQCA_SSL_CRL_NAMELIST).

The length of the string is MQ_NAMELIST_NAME_LENGTH.

Indicates the name of a namelist of authentication information objects to be used for CRL checking by the queue manager.

***SSLCryptoHardware* (MQCFST)**

Parameters to configure the SSL cryptographic hardware (parameter identifier: MQCA_SSL_CRYPTO_HARDWARE).

The length of the string is MQ_SSL_CRYPTO_HARDWARE_LENGTH.

Sets the name of the parameter string required to configure the cryptographic hardware present on the system.

This parameter is supported on AIX, HP-UX, Solaris, Linux, and Windows only.

***SSLEvent* (MQCFIN)**

Controls whether SSL events are generated (parameter identifier: MQIA_SSL_EVENT).

The value can be:

MQEVR_DISABLED

Event reporting disabled.

MQEVR_ENABLED

Event reporting enabled.

***SSLFipsRequired* (MQCFIN)**

Controls whether only FIPS-certified algorithms are to be used if cryptography is executed in WebSphere MQ itself (parameter identifier: MQIA_SSL_FIPS_REQUIRED). This parameter is valid only on Windows and UNIX platforms.

The value can be:

MQSSL_FIPS_NO

Any supported CipherSpec can be used.

MQSSL_FIPS_YES

Only FIPS-certified cryptographic algorithms are to be used if cryptography is executed in WebSphere MQ itself.

SSLKeyRepository (MQCFST)

Location and name of the SSL key repository (parameter identifier: MQCA_SSL_KEY_REPOSITORY).

The length of the string is MQ_SSL_KEY_REPOSITORY_LENGTH.

Indicates the name of the Secure Sockets Layer key repository.

The format of the name depends on the environment.

SSLKeyResetCount (MQCFIN)

SSL key reset count (parameter identifier: MQIA_SSL_RESET_COUNT).

The number of unencrypted bytes that initiating SSL channel MCAs send or receive before renegotiating the secret key.

SSLTasks (MQCFIN)

Number of server subtasks used for processing SSL calls (parameter identifier: MQIA_SSL_TASKS). This parameter is valid only on z/OS.

The number of server subtasks used for processing SSL calls.

StartStopEvent (MQCFIN)

Controls whether start and stop events are generated (parameter identifier: MQIA_START_STOP_EVENT).

The value can be:

MQEVR_DISABLED

Event reporting disabled.

MQEVR_ENABLED

Event reporting enabled.

StatisticsInterval (MQCFIN)

The time interval, in seconds, at which statistics monitoring data is written to the monitoring queue (parameter identifier: MQIA_STATISTICS_INTERVAL).

This parameter is valid only on AIX, HP-UX, Linux, i5/OS, Solaris, and Windows.

SyncPoint (MQCFIN)

Syncpoint availability (parameter identifier: MQIA_SYNCPOINT).

The value can be:

MQSP_AVAILABLE

Units of work and syncpointing available.

MQSP_NOT_AVAILABLE

Units of work and syncpointing not available.

TCPChannels (MQCFIN)

The maximum number of channels that can be current, or clients that can be connected, that use the TCP/IP transmission protocol (parameter identifier: MQIA_TCP_CHANNELS).

This parameter applies to z/OS only.

TCPKeepAlive (**MQCFIN**)

Whether the TCP KEEPALIVE facility is to be used to check whether the other end of the connection is still available (parameter identifier: MQIA_TCP_KEEP_ALIVE).

The value can be:

MQTCPKEEP_YES

The TCP KEEPALIVE facility is to be used as specified in the TCP profile configuration data set. The interval is specified in the *KeepAliveInterval* channel attribute.

MQTCPKEEP_NO

The TCP KEEPALIVE facility is not to be used.

This parameter applies to z/OS only.

TCPName (**MQCFST**)

The name of the TCP/IP system that you are using (parameter identifier: MQIA_TCP_NAME).

This parameter applies to z/OS only.

TCPStackType (**MQCFIN**)

Whether the channel initiator may use only the TCP/IP address space specified in *TCPName*, or may optionally bind to any selected TCP/IP address (parameter identifier: MQIA_TCP_STACK_TYPE).

The value can be:

MQTCPSTACK_SINGLE

The channel initiator may only use the TCP/IP address space specified in *TCPName*.

MQTCPSTACK_MULTIPLE

The channel initiator may use any TCP/IP address space available to it.

This parameter applies to z/OS only.

TraceRouteRecording (**MQCFIN**)

Whether trace-route information can be recorded and a reply message generated (parameter identifier: MQIA_TRACE_ROUTE_RECORDING).

The value can be:

MQRECORDING_DISABLED

Trace-route information cannot be recorded.

MQRECORDING_MSG

Trace-route information can be recorded and sent to the destination specified by the originator of the message causing the trace route record.

MQRECORDING_Q

Trace-route information can be recorded and sent to SYSTEM.ADMIN TRACE.ROUTE.QUEUE.

TriggerInterval (**MQCFIN**)

Trigger interval (parameter identifier: MQIA_TRIGGER_INTERVAL).

Specifies the trigger time interval, expressed in milliseconds, for use only with queues where *TriggerType* has a value of MQTT_FIRST.

Inquire Queue Manager Status

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
		X	X	X	

The Inquire Queue Manager Status (MQCMD_INQUIRE_Q_MGR_STATUS) command inquires about the status of the local queue manager.

Required parameters:

None

Optional parameters:

QMStatusAttrs

Optional parameters

QMStatusAttrs (MQCFIL)

Queue manager status attributes (parameter identifier: MQIACF_Q_MGR_STATUS_ATTRS).

The attribute list might specify the following on its own (this is the default value used if the parameter is not specified):

MQIACF_ALL

All attributes.

or a combination of the following:

MQCA_Q_MGR_NAME

Name of the local queue manager.

MQCACF_CURRENT_LOG_EXTENT_NAME

Name of the log extent currently being written to by the logger. This is available only on queue managers using linear logging. On other queue managers, this is blank.

MQCACF_LOG_PATH

Location of the recovery log extents.

MQCACF_MEDIA_LOG_EXTENT_NAME

Name of the earliest log extent required to perform media recovery. This is available only on queue managers using linear logging. On other queue managers, this is blank.

MQCACF_RESTART_LOG_EXTENT_NAME

Name of the earliest log extent required to perform restart recovery. This is available only on queue managers using linear logging. On other queue managers, this is blank.

MQIACF_CHINIT_STATUS

Current status of the channel initiator.

MQIACF_CMD_SERVER_STATUS

Current status of the command server.

MQIACF_CONNECTION_COUNT

Current number of connections to the queue manager.

MQIACF_Q_MGR_STATUS

Current status of the queue manager.

Inquire Queue Manager Status (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
		X	X	X	

The response to the Inquire Queue Manager Status (MQCMD_INQUIRE_Q_MGR_STATUS) command consists of the response header followed by the *QMgrName* and *QMgrStatus* structures and the requested combination of attribute parameter structures.

Always returned:

QMgrName, *QMgrStatus*

Returned if requested:

ChannelInitiatorStatus, *CommandServerStatus*, *ConnectionCount*,
CurrentLog, *LogPath*, *MediaRecoveryLog*, *RestartRecoveryLog*

Response data

ChannelInitiatorStatus (**MQCFIN**)

Status of the channel initiator reading SYSTEM.DEFAULT.INITIATION.QUEUE (parameter identifier: MQIACF_CHINIT_STATUS).

The value can be:

MQSVC_STATUS_STOPPED

The channel initiator is not running.

MQSVC_STATUS_STARTING

The channel initiator is in the process of initializing.

MQSVC_STATUS_RUNNING

The channel initiator is fully initialized and is running.

MQSVC_STATUS_STOPPING

The channel initiator is stopping.

CommandServerStatus (**MQCFIN**)

Status of the command server (parameter identifier: MQIACF_CCMD_SERVER_STATUS).

The value can be:

MQSVC_STATUS_STOPPED

The command server is not running.

MQSVC_STATUS_STARTING

The command server is in the process of initializing.

MQSVC_STATUS_RUNNING

The command server is fully initialized and is running.

MQSVC_STATUS_STOPPING

The command server is stopping.

ConnectionCount (**MQCFIN**)

Connection count (parameter identifier: MQIACF_CONNECTION_COUNT).

The current number of connections to the queue manager.

CurrentLog (**MQCFST**)

Log extent name (parameter identifier:
MQCACF_CURRENT_LOG_EXTENT_NAME).

The name of the log extent that was being written to at the time of the Inquire command. If the queue manager is using circular logging, this is blank.

The maximum length of the string is MQ_LOG_EXTENT_NAME_LENGTH.

LogPath (**MQCFST**)

Location of the recovery log extents (parameter identifier:
MQCACF_LOG_PATH).

This identifies the directory where log files are created by the queue manager.

The maximum length of the string is MQ_LOG_PATH_LENGTH.

MediaRecoveryLog (**MQCFST**)

Name of the oldest log extent required by the queue manager to perform media recovery (parameter identifier:
MQCACF_MEDIA_LOG_EXTENT_NAME). This is available only on queue managers using linear logging. If the queue manager is using circular logging, this is blank.

The maximum length of the string is MQ_LOG_EXTENT_NAME_LENGTH.

QMgrName (**MQCFST**)

Name of the local queue manager (parameter identifier:
MQCA_Q_MGR_NAME).

The maximum length of the string is MQ_Q_MGR_NAME_LENGTH.

QMgrStatus (**MQCFIN**)

Current execution status of the queue manager (parameter identifier:
MQIACF_Q_MGR_STATUS).

The value can be:

MQQMSTA_STARTING

The queue manager is initializing.

MQQMSTA_RUNNING

The queue manager is fully initialized and is running.

MQQMSTA QUIESCING

The queue manager is quiescing.

RestartRecoveryLog (**MQCFST**)

Name of the oldest log extent required by the queue manager to perform restart recovery (parameter identifier:
MQCACF_RESTART_LOG_EXTENT_NAME).

This is available only on queue managers using linear logging. If the queue manager is using circular logging, this is blank.

The maximum length of the string is MQ_LOG_EXTENT_NAME_LENGTH.

Inquire Queue Names

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Inquire Queue Names (MQCMD_INQUIRE_Q_NAMES) command inquires a list of queue names that match the generic queue name, and the optional queue type specified.

Required parameters:

QName

Optional parameters:

CommandScope, QSGDisposition, QType

Required parameters

QName (MQCFST)

Queue name (parameter identifier: MQCA_Q_NAME).

Generic queue names are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all objects having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length of the string is MQ_Q_LENGTH.

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

QSGDisposition (MQCFIN)

Disposition of the object within the group (parameter identifier: MQIA_QSG_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object for which information is to be returned (that is, where it is defined and how it behaves). The value can be:

MQQSGD_LIVE

The object is defined as MQQSGD_Q_MGR or MQQSGD_COPY. This is the default value if the parameter is not specified.

MQQSGD_ALL

The object is defined as MQQSGD_Q_MGR or MQQSGD_COPY.

If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this option also displays information for objects defined with MQQSGD_GROUP.

If MQQSGD_LIVE is specified or defaulted, or if MQQSGD_ALL is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

MQQSGD_COPY

The object is defined as MQQSGD_COPY.

MQQSGD_GROUP

The object is defined as MQQSGD_GROUP. This is permitted only in a shared queue environment.

MQQSGD_Q_MGR

The object is defined as MQQSGD_Q_MGR.

MQQSGD_PRIVATE

The object is defined as MQQSGD_Q_MGR or MQQSGD_COPY. Note that MQQSGD_PRIVATE returns the same information as MQQSGD_LIVE.

MQQSGD_SHARED

The object is defined as MQQSGD_SHARED. This is permitted only in a shared queue environment.

***QType* (MQCFIN)**

Queue type (parameter identifier: MQIA_Q_TYPE).

If present, this parameter limits the queue names returned to queues of the specified type. If this parameter is not present, queues of all types are eligible. The value can be:

MQQT_ALL

All queue types.

MQQT_LOCAL

Local queue.

MQQT_ALIAS

Alias queue definition.

MQQT_REMOTE

Local definition of a remote queue.

MQQT_MODEL

Model queue definition.

The default value if this parameter is not specified is MQQT_ALL.

Inquire Queue Names (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The response to the Inquire Queue Names (MQCMD_INQUIRE_Q_NAMES) command consists of the response header followed by a single parameter structure giving zero or more names that match the specified queue name. This is followed by the *QTypes* structure, with the same number of entries as the *QNames* structure. Each entry gives the type of the queue with the corresponding entry in the *QNames* structure.

In addition to this, on z/OS only, the *QSGDispositions* parameter structure (with the same number of entries as the *QNames* structure) is returned. Each entry in this structure indicates the disposition of the object with the corresponding entry in the *QNames* structure.

Always returned:

QNames, *QSGDispositions*, *QTypes*

Returned if requested:

None

Response data

***QNames* (MQCFSL)**

List of queue names (parameter identifier: MQCACF_Q_NAMES).

***QSGDispositions* (MQCFIL)**

List of QSG dispositions (parameter identifier: MQIACF_QSG_DISPS). This is valid on z/OS only. Possible values for fields in this structure are:

MQQSGD_COPY

The object is defined as MQQSGD_COPY.

MQQSGD_GROUP

The object is defined as MQQSGD_GROUP.

MQQSGD_Q_MGR

The object is defined as MQQSGD_Q_MGR.

MQQSGD_SHARED

The object is defined as MQQSGD_SHARED.

***QTypes* (MQCFIL)**

List of queue types parameter identifier: MQIACF_Q_TYPES). Possible values for fields in this structure are:

MQQT_ALIAS

Alias queue definition.

MQQT_LOCAL

Local queue.

MQQT_REMOTE

Local definition of a remote queue.

MQQT_MODEL

Model queue definition.

Inquire Queue Status

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
		X	X	X	X

The Inquire Queue Status (MQCMD_INQUIRE_Q_STATUS) command inquires about the status of a local WebSphere MQ queue. You must specify the name of a local queue for which you want to receive status information.

Required parameters:

QName

Optional parameters:

CommandScope, IntegerFilterCommand, OpenType, QSGDisposition,
QStatusAttrs, StatusType, StringFilterCommand,

Required parameters

QName (MQCFST)

Queue name (parameter identifier: MQCA_Q_NAME).

Generic queue names are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all queues having names that start with the selected character string. An asterisk on its own matches all possible names.

The queue name is always returned, regardless of the attributes requested.

The maximum length of the string is MQ_Q_NAME_LENGTH.

Optional parameters

ByteStringFilterCommand (MQCFBF)

Byte string filter command descriptor. The parameter identifier must be MQBACF_EXTERNAL_UOW_ID or MQBACF_Q_MGR_UOW_ID. Use this to restrict the output from the command by specifying a filter condition. See "MQCFBF - PCF byte string filter parameter" on page 422 for information about using this filter condition.

If you specify a byte string filter, you cannot also specify an integer filter using the *IntegerFilterCommand* parameter, or a string filter using the *StringFilterCommand* parameter.

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.

- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

You cannot use *CommandScope* as a parameter to filter on.

IntegerFilterCommand (MQCFIF)

Integer filter command descriptor. The parameter identifier must be any integer type parameter allowed in *QStatusAttrs* except MQIACF_ALL, MQIACF_MONITORING, and MQIACF_Q_TIME_INDICATOR. Use this to restrict the output from the command by specifying a filter condition. See “MQCFIF - PCF integer filter parameter” on page 427 for information about using this filter condition.

If you specify an integer filter, you cannot also specify a byte string filter using the *ByteStringFilterCommand* parameter or a string filter using the *StringFilterCommand* parameter.

OpenType (MQCFIN)

Queue status open type (parameter identifier: MQIACF_OPEN_TYPE).

It is always returned, regardless of the queue instance attributes requested.

The value can be:

MQQSOT_ALL

Selects status for queues that are open with any type of access.

MQQSOT_INPUT

Selects status for queues that are open for input.

MQQSOT_OUTPUT

Selects status for queues that are open for output.

The default value if this parameter if not specified is MQQSOT_ALL.

Filtering is not supported for this parameter.

QSGDisposition (MQCFIN)

QSG disposition (parameter identifier: MQIA_QSG_DISP).

Specifies the disposition of the object (that is, where it is defined and how it behaves). This is valid only on z/OS. The value can be:

MQQSGD_COPY

The object is defined as MQQSGD_COPY.

MQQSGD_Q_MGR

The object is defined as MQQSGD_Q_MGR.

MQQSGD_SHARED

The object is defined as MQQSGD_SHARED.

You cannot use *QSGDisposition* as a parameter to filter on.

QStatusAttrs (MQCFIL)

Queue status attributes (parameter identifier: MQIACF_Q_STATUS_ATTRS).

The attribute list can specify the following on its own (this is the default value used if the parameter is not specified):

MQIACF_ALL

All attributes.

or a combination of the following:

Where *StatusType* is MQIACF_Q_STATUS:

MQCA_Q_NAME
Queue name.

MQCACF_LAST_GET_DATE
Date of the last message successfully destructively read from the queue.

MQCACF_LAST_GET_TIME
Time of the last message successfully destructively read from the queue.

MQCACF_LAST_PUT_DATE
Date of the last message successfully put to the queue.

MQCACF_LAST_PUT_TIME
Time of the last message successfully put to the queue.

MQCACF_MEDIA_LOG_EXTENT_NAME
Identity of the oldest log extent needed to perform media recovery of the queue.

On i5/OS, this identifies the name of the oldest journal receiver needed to perform media recovery of the queue.

MQIA_CURRENT_Q_DEPTH
The current number of messages on the queue.

MQIA_MONITORING_Q
Current level of monitoring data collection.

MQIA_OPEN_INPUT_COUNT
The number of handles that are currently open for input for the queue.
This does not include handles that are open for browse.

MQIA_OPEN_OUTPUT_COUNT
The number of handles that are currently open for output for the queue.

MQIACF_HANDLE_STATE
Whether an API call is in progress.

MQIACF_MONITORING
All of the queue status monitoring attributes. These are:

- MQCACF_LAST_GET_DATE
- MQCACF_LAST_GET_TIME
- MQCACF_LAST_PUT_DATE
- MQCACF_LAST_PUT_TIME
- MQIA_MONITORING_Q
- MQIACF_OLEDEST_MSG_AGE
- MQIACF_Q_TIME_INDICATOR

Filtering is not supported for this parameter.

MQIACF_OLEDEST_MSG_AGE
Age of oldest message on the queue.

MQIACF_Q_TIME_INDICATOR
Indicator of the time that messages remain on the queue.

MQIACF_UNCOMMITTED_MSGS

Whether there are uncommitted messages on the queue.

Where *StatusType* is MQIACF_Q_HANDLE:

MQBACF_EXTERNAL_UOW_ID

Unit of recovery identifier assigned by the queue manager.

MQBACF_Q_MGR_UOW_ID

External unit of recovery identifier associated with the connection.

MQCA_Q_NAME

Queue name.

MQCACF_APPL_TAG

This is a string containing the tag of the application connected to the queue manager.

MQCACF_ASID

Address-space identifier of the application identified by *ApplTag*. This parameter is valid on z/OS only.

MQCACF_PSB_NAME

Name of the program specification block (PSB) associated with the running IMS transaction. This parameter is valid on z/OS only.

MQCACF_PSTID

Identifier of the IMS program specification table (PST) for the connected IMS region. This parameter is valid on z/OS only.

MQCACF_TASK_NUMBER

CICS task number. This parameter is valid on z/OS only.

MQCACF_TRANSACTION_ID

CICS transaction identifier. This parameter is valid on z/OS only.

MQCACF_USER_IDENTIFIER

The username of the application that has opened the specified queue.

MQCACH_CHANNEL_NAME

The name of the channel that has the queue open, if any.

MQCACH_CONNECTION_NAME

The connection name of the channel that has the queue open, if any.

MQIA_APPL_TYPE

The type of application that has the queue open.

MQIACF_OPEN_BROWSE

Open browse.

Filtering is not supported for this parameter.

MQIACF_OPEN_INPUT_TYPE

Open input type.

Filtering is not supported for this parameter.

MQIACF_OPEN_INQUIRE

Open inquire.

Filtering is not supported for this parameter.

MQIACF_OPEN_OPTIONS

The options used to open the queue.

If this parameter is requested, the following parameter structures are also returned:

- *OpenBrowse*
- *OpenInputType*
- *OpenInquire*
- *OpenOutput*
- *OpenSet*

Filtering is not supported for this parameter.

MQIACF_OPEN_OUTPUT

Open output.

Filtering is not supported for this parameter.

MQIACF_OPEN_SET

Open set.

Filtering is not supported for this parameter.

MQIACF_PROCESS_ID

The process identifier of the application that has opened the specified queue.

MQIACF_THREAD_ID

The thread identifier of the application that has opened the specified queue.

MQIACF_UOW_TYPE

Type of external unit of recovery identifier as seen by the queue manager.

StatusType (**MQCFIN**)

Queue status type (parameter identifier: MQIACF_Q_STATUS_TYPE).

Specifies the type of status information required.

The value can be:

MQIACF_Q_STATUS

Selects status information relating to queues.

MQIACF_Q_HANDLE

Selects status information relating to the handles that are accessing the queues.

The default value, if this parameter is not specified, is MQIACF_Q_STATUS.

You cannot use *StatusType* as a parameter to filter on.

StringFilterCommand (**MQCFSF**)

String filter command descriptor. The parameter identifier must be any string type parameter allowed in *QStatusAttrs* except MQCA_Q_NAME. Use this to restrict the output from the command by specifying a filter condition. See “MQCFSF - PCF string filter parameter” on page 434 for information about using this filter condition.

If you specify a string filter, you cannot also specify a byte string filter using the *ByteStringFilterCommand* parameter or an integer filter using the *IntegerFilterCommand* parameter.

Error codes

This command might return the following in the response format header, in addition to the values shown on page "Error codes applicable to all commands" on page 27.

Reason (MQLONG)

The value can be:

MQRCCF_Q_TYPE_ERROR

Queue type not valid.

Inquire Queue Status (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
		X	X	X	X

The response to the Inquire Queue Status (MQCMD_INQUIRE_Q_STATUS) command consists of the response header followed by the *QName* structure and a set of attribute parameter structures determined by the value of *StatusType* in the Inquire command.

Always returned:

QName, QSGDisposition, StatusType

Possible values of *StatusType* are:

MQIACF_Q_STATUS

Returns status information relating to queues.

MQIACF_Q_HANDLE

Returns status information relating to the handles that are accessing the queues.

Returned if requested and *StatusType* is MQIACF_Q_STATUS:

CurrentQDepth, LastGetDate, LastGetTime, LastPutDate, LastPutTime, MediaRecoveryLogExtent, OldestMsgAge, OnQTime, OpenInputCount, OpenOutputCount, QueueMonitoring, UncommittedMsgs

Returned if requested and *StatusType* is MQIACF_Q_HANDLE:

ApplTag, ApplType, ASId, ChannelName, ConnectionName, ExternalUOWId, HandleState, OpenOptions, ProcessId, PSBName, PSTId, QMgrUOWId, TaskNumber, ThreadId, TransactionId, UOWIdentifier, UOWType, UserIdentifier

Response data if *StatusType* is MQIACF_Q_STATUS

CurrentQDepth (MQCFIN)

Current queue depth (parameter identifier: MQIA_CURRENT_Q_DEPTH).

LastGetDate (MQCFST)

Date on which the last message was destructively read from the queue (parameter identifier: MQCACF_LAST_GET_DATE).

The date, in the form yyyy-mm-dd, on which the last message was successfully read from the queue. The date is returned in the time zone in which the queue manager is running.

The maximum length of the string is MQ_DATE_LENGTH.

LastGetTime (MQCFST)

Time at which the last message was destructively read from the queue (parameter identifier: MQCACF_LAST_GET_TIME).

The time, in the form hh.mm.ss, at which the last message was successfully read from the queue. The time is returned in the time zone in which the queue manager is running.

The maximum length of the string is MQ_TIME_LENGTH.

LastPutDate (MQCFST)

Date on which the last message was successfully put to the queue (parameter identifier: MQCACF_LAST_PUT_DATE).

The date, in the form yyyy-mm-dd, on which the last message was successfully put to the queue. The date is returned in the time zone in which the queue manager is running.

The maximum length of the string is MQ_DATE_LENGTH.

LastPutTime (MQCFST)

Time at which the last message was successfully put to the queue (parameter identifier: MQCACF_LAST_PUT_TIME).

The time, in the form hh.mm.ss, at which the last message was successfully put to the queue. The time is returned in the time zone in which the queue manager is running.

The maximum length of the string is MQ_TIME_LENGTH.

MediaRecoveryLogExtent (MQCFST)

Name of the oldest log extent needed to perform media recovery of the queue (parameter identifier: MQCACF_MEDIA_LOG_EXTENT_NAME).

On i5/OS, this identifies the name of the oldest journal receiver needed to perform media recovery of the queue.

Note that the name returned is of the form Snnnnnn.LOG and is *not* a fully qualified path name. This allows the name to be easily correlated with the messages issued following an rcdmqimg command to identify those queues causing the media recovery LSN not to move forwards.

This is valid on AIX, HP-UX, Linux, i5/OS, Solaris, and Windows.

The maximum length of the string is MQ_LOG_EXTENT_NAME_LENGTH.

OldestMsgAge (MQCFIN)

Age of the oldest message (parameter identifier: MQIACF_OLDEST_MSG_AGE). Age, in seconds, of the oldest message on the queue.

If the value is unavailable, MQMON_NOT_AVAILABLE is returned. If the queue is empty, 0 is returned. If the value exceeds 999 999 999, it is returned as 999 999 999.

OnQTime (MQCFIL)

Indicator of the time that messages remain on the queue (parameter identifier: MQIACH_Q_TIME_INDICATOR). Amount of time, in microseconds, that a message spent on the queue. Two values are returned:

- A value based on recent activity over a short period of time.
- A value based on activity over a longer period of time.

Where no measurement is available, the value MQMON_NOT_AVAILABLE is returned. If the value exceeds 999 999 999, it is returned as 999 999 999.

***OpenInputCount* (MQCFIN)**

Open input count (parameter identifier: MQIA_OPEN_INPUT_COUNT).

***OpenOutputCount* (MQCFIN)**

Open output count (parameter identifier: MQIA_OPEN_OUTPUT_COUNT).

***QName* (MQCFST)**

Queue name (parameter identifier: MQCA_Q_NAME).

The maximum length of the string is MQ_Q_NAME_LENGTH.

***QSGDisposition* (MQCFIN)**

QSG disposition (parameter identifier: MQIA_QSG_DISP).

Returns the disposition of the object (that is, where it is defined and how it behaves). This parameter is valid on z/OS only. The value can be:

MQQSGD_COPY

The object is defined as MQQSGD_COPY.

MQQSGD_Q_MGR

The object is defined as MQQSGD_Q_MGR.

MQQSGD_SHARED

The object is defined as MQQSGD_SHARED.

***QueueMonitoring* (MQCFIN)**

Current level of monitoring data collection for the queue (parameter identifier: MQIA_MONITORING_Q). The value can be:

MQMON_OFF

Monitoring for the queue is switched off.

MQMON_LOW

Low rate of data collection.

MQMON_MEDIUM

Medium rate of data collection.

MQMON_HIGH

High rate of data collection.

***StatusType* (MQCFST)**

Queue status type (parameter identifier: MQIACF_Q_STATUS_TYPE).

Specifies the type of status information.

***UncommittedMsgs* (MQCFIN)**

Number of uncommitted messages (parameter identifier: MQIACF_UNCOMMITTED_MSGS).

Response data if StatusType is MQIACF_Q_HANDLE

***ApplTag* (MQCFST)**

Open application tag (parameter identifier: MQCACF_APPL_TAG).

The maximum length of the string is MQ_APPL_TAG_LENGTH.

***ApplType* (MQCFIN)**

Open application type (parameter identifier: MQIA_APPL_TYPE).

The value can be:

MQAT_QMGR

A queue manager process.

MQAT_CHANNEL_INITIATOR

The channel initiator.

MQAT_USER

A user application.

MQAT_BATCH

Application using a batch connection. This applies only to z/OS.

MQAT_RRS_BATCH

RRS-coordinated application using a batch connection. This applies only to z/OS.

MQAT_CICS

A CICS transaction. This applies only to z/OS.

MQAT_IMS

An IMS transaction. This applies only to z/OS.

ASId (MQCFST)

Address-space identifier (parameter identifier: MQCACF_ASID).

The 4-character address-space identifier of the application identified by *ApplTag*. It distinguishes duplicate values of *ApplTag*. This parameter applies only to z/OS.

The length of the string is MQ_ASID_LENGTH.

ChannelName (MQCFST)

Channel name (parameter identifier: MQCACH_CHANNEL_NAME).

The maximum length of the string is MQ_CHANNEL_NAME_LENGTH.

Connname (MQCFST)

Connection name (parameter identifier: MQCACH_CONNECTION_NAME).

The maximum length of the string is MQ_CONN_NAME_LENGTH.

ExternalUOWId (MQCFBS)

RRS unit-of-recovery identifier (parameter identifier: MQBACF_EXTERNAL_UOW_ID).

The RRS unit-of-recovery identifier associated with the handle. This parameter is valid only on z/OS only.

The length of the string is MQ_EXTERNAL_UOW_ID_LENGTH.

HandleState (MQCFIN)

State of the handle (parameter identifier: MQIACF_HANDLE_STATE).

The value may be:

MQHSTATE_ACTIVE

An API call from a connection is currently in progress for this object. For a queue, this condition can arise when an MQGET WAIT call is in progress.

If there is an MQGET SIGNAL outstanding, then this does not mean, by itself, that the handle is active.

MQHSTATE_INACTIVE

No API call from a connection is currently in progress for this object. For a queue, this condition can arise when no MQGET WAIT call is in progress.

***OpenBrowse* (MQCFIN)**

Open browse (parameter identifier: MQIACF_OPEN_BROWSE).

The value can be:

MQQSO_YES

The queue is open for browsing.

MQQSO_NO

The queue is not open for browsing.

***OpenInputType* (MQCFIN)**

Open input type (parameter identifier: MQIACF_OPEN_INPUT_TYPE).

The value can be:

MQQSO_NO

The queue is not open for inputing.

MQQSO_SHARED

The queue is open for shared input.

MQQSO_EXCLUSIVE

The queue is open for exclusive input.

***OpenInquire* (MQCFIN)**

Open inquire (parameter identifier: MQIACF_OPEN_INQUIRE).

The value can be:

MQQSO_YES

The queue is open for inquiring.

MQQSO_NO

The queue is not open for inquiring.

***OpenOptions* (MQCFIN)**

Open options currently in force for the queue (parameter identifier: MQIACF_OPEN_OPTIONS).

***OpenOutput* (MQCFIN)**

Open output (parameter identifier: MQIACF_OPEN_OUTPUT).

The value can be:

MQQSO_YES

The queue is open for outputting.

MQQSO_NO

The queue is not open for outputting.

***OpenSet* (MQCFIN)**

Open set (parameter identifier: MQIACF_OPEN_SET).

The value can be:

MQQSO_YES

The queue is open for setting.

MQQSO_NO

The queue is not open for setting.

ProcessId (**MQCFIN**)

Open application process ID (parameter identifier: MQIACF_PROCESS_ID).

PSBName (**MQCFST**)

Program specification block (PSB) name (parameter identifier: MQCACF_PSB_NAME).

The 8-character name of the PSB associated with the running IMS transaction. This parameter is valid on z/OS only.

The length of the string is MQ_PSB_NAME_LENGTH.

PSTId (**MQCFST**)

Program specification table (PST) identifier (parameter identifier: MQCACF_PST_ID).

The 4-character identifier of the PST region identifier for the connected IMS region. This parameter is valid on z/OS only.

The length of the string is MQ_PST_ID_LENGTH.

QMgrUOWId (**MQCFBS**)

The unit of recovery assigned by the queue manager (parameter identifier: MQBACF_Q_MGR_UOW_ID).

On z/OS, this is a 6-byte log RBA, displayed as 12 hexadecimal characters. On platforms other than z/OS, this is an 8-byte transaction identifier, displayed as 16 hexadecimal characters.

The maximum length of the string is MQ_UOW_ID_LENGTH.

QName (**MQCFST**)

Queue name (parameter identifier: MQCA_Q_NAME).

The maximum length of the string is MQ_Q_NAME_LENGTH.

QSGDisposition (**MQCFIN**)

QSG disposition (parameter identifier: MQIA_QSG_DISP).

Returns the disposition of the object (that is, where it is defined and how it behaves). This parameter is valid on z/OS only. The value can be:

MQQSGD_COPY

The object is defined as MQQSGD_COPY.

MQQSGD_Q_MGR

The object is defined as MQQSGD_Q_MGR.

MQQSGD_SHARED

The object is defined as MQQSGD_SHARED.

StatusType (**MQCFST**)

Queue status type (parameter identifier: MQIACF_Q_STATUS_TYPE).

Specifies the type of status information.

TaskNumber (**MQCFST**)

CICS task number (parameter identifier: MQCACF_TASK_NUMBER).

A 7-digit CICS task number. This parameter is valid on z/OS only.

The length of the string is MQ_TASK_NUMBER_LENGTH.

ThreadId (**MQCFIN**)

Open application thread ID (parameter identifier: MQIACF_THREAD_ID).

The maximum length of the string is MQ_PROCESS_NAME_LENGTH.

TransactionId (**MQCFST**)

CICS transaction identifier (parameter identifier: MQCACF_TRANSACTION_ID).

A 4-character CICS transaction identifier. This parameter is valid on z/OS only.

The length of the string is MQ_TRANSACTION_ID_LENGTH.

UOWIdentifier (**MQCFBS**)

The external unit of recovery associated with the connection (parameter identifier: MQBACF_EXTERNAL_UOW_ID).

This is the recovery identifier for the unit of recovery. Its format is determined by the value of *UOWType*.

The maximum length of the string is MQ_UOW_ID_LENGTH.

UOWType (**MQCFIN**)

Type of external unit of recovery identifier as perceived by the queue manager (parameter identifier: MQIACF_UOW_TYPE).

The value can be:

MQUOWT_Q_MGR

MQUOWT_CICS

Valid only on z/OS.

MQUOWT_RRS

Valid only on z/OS.

MQUOWT_IMS

Valid only on z/OS.

MQUOWT_XA

UOWType identifies the *UOWIdentifier* type and not the type of the transaction coordinator. When the value of *UOWType* is MQUOWT_Q_MGR, the associated identifier is in *QMgrUOWId* (and not *UOWIdentifier*).

UserIdentifier (**MQCFST**)

Open application username (parameter identifier: MQCACF_USER_IDENTIFIER).

The maximum length of the string is MQ_MAX_USER_ID_LENGTH.

Inquire Security

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The Inquire Security (MQCMD_INQUIRE_SECURITY) command returns information about the current settings for the security parameters.

Required parameters:

None

Optional parameters:

CommandScope, SecurityAttrs

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

SecurityAttrs (MQCFIL)

Security parameter attributes (parameter identifier: MQIACF_SECURITY_ATTRS).

The attribute list might specify the following on its own (this is the default value used if the parameter is not specified):

MQIACF_ALL

All attributes.

or a combination of the following:

MQIACF_SECURITY_SWITCH

Current setting of the switch profiles. If the subsystem security switch is off, no other switch profile settings are returned.

MQIACF_SECURITY_TIMEOUT

Timeout value.

MQIACF_SECURITY_INTERVAL

Time interval between checks.

Inquire Security (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The response to the Inquire Security (MQCMD_INQUIRE_SECURITY) command consists of the response header followed by the requested combination of attribute parameter structures. One message is returned if either *SecurityTimeout* or *SecurityInterval* is specified on the command. If *SecuritySwitch* is specified, one message per security switch found is returned. This includes the *SecuritySwitch*, *SecuritySwitchSetting*, and *SecuritySwitchProfile* attribute parameter structures.

Returned if requested:

SecurityInterval, *SecuritySwitch*, *SecuritySwitchProfile*,
SecuritySwitchSetting, *SecurityTimeout*

Response data

SecurityInterval (MQCFIN)

Time interval between checks (parameter identifier: MQIACF_SECURITY_INTERVAL).

The interval, in minutes, between checks for user IDs and their associated resources to determine whether *SecurityTimeout* has expired.

SecuritySwitch (MQCFIN)

Security switch profile (parameter identifier: MQIA_CF_LEVEL).

. The value can be:

MQSECSW_SUBSYSTEM

Subsystem security switch.

MQSECSW_Q_MGR

Queue manager security switch.

MQSECSW_QSG

Queue sharing group security switch.

MQSECSW_CONNECTION

Connection security switch.

MQSECSW_COMMAND

Command security switch.

MQSECSW_CONTEXT

Context security switch.

MQSECSW_ALTERNATE_USER

Alternate user security switch.

MQSECSW_PROCESS

Process security switch.

MQSECSW_NAMELIST

Namelist security switch.

MQSECSW_Q

Queue security switch.

MQSECSW_COMMAND_RESOURCES

Command resource security switch.

SecuritySwitchProfile (MQCFST)

Security switch profile (parameter identifier: MQCACF_SECURITY_PROFILE).

The maximum length of the string is MQ_SECURITY_PROFILE_LENGTH.

SecuritySwitchSetting (MQCFIN)

Setting of the security switch (parameter identifier: MQIACF_SECURITY_SETTING).

The value can be:

MQSECSW_ON_FOUND

Switch ON, profile found.

MQSECSW_OFF_FOUND

Switch OFF, profile found.

MQSECSW_ON_NOT_FOUND

Switch ON, profile not found.

MQSECSW_OFF_NOT_FOUND

Switch OFF, profile not found.

MQSECSW_OFF_ERROR

Switch OFF, profile error.

MQSECSW_ON_OVERRIDDEN

Switch ON, profile overridden.

SecurityTimeout (MQCFIN)

Timeout value (parameter identifier: MQIACF_SECURITY_TIMEOUT).

How long, in minutes, security information about an unused user ID and associated resources is retained.

Inquire Service

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
		X	X	X	

The Inquire Service (MQCMD_INQUIRE_SERVICE) command inquires about the attributes of existing WebSphere MQ services.

Required parameters:

ServiceName

Optional parameters:

IntegerFilterCommand, *ServiceAttrs*, *StringFilterCommand*

Required parameters

ServiceName (MQCFST)

Service name (parameter identifier: MQCA_SERVICE_NAME).

This is the name of the service whose attributes are required. Generic service names are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all services having names that start with the selected character string. An asterisk on its own matches all possible names.

The service name is always returned regardless of the attributes requested.

The maximum length of the string is MQ_OBJECT_NAME_LENGTH.

Optional parameters

IntegerFilterCommand (MQCFIF)

Integer filter command descriptor. The parameter identifier must be any integer type parameter allowed in *ServiceAttrs* except MQIACF_ALL. Use this to restrict the output from the command by specifying a filter condition. See “MQCFIF - PCF integer filter parameter” on page 427 for information about using this filter condition.

If you specify an integer filter, you cannot also specify a string filter using the *StringFilterCommand* parameter.

ServiceAttrs (MQCFIL)

Service attributes (parameter identifier: MQIACF_SERVICE_ATTRS).

The attribute list might specify the following on its own (this is the default value if the parameter is not specified):

MQIACF_ALL

All attributes.

or a combination of the following:

MQCA_ALTERATION_DATE

Date on which the definition was last altered.

MQCA_ALTERATION_TIME

Time at which the definition was last altered.

MQCA_SERVICE_DESC

Description of service definition.

MQCA_SERVICE_NAME

Name of service definition.

MQCA_SERVICE_START_ARGS

Arguments to be passed to the service program.

MQCA_SERVICE_START_COMMAND

Name of program to run to start the service.

MQCA_SERVICE_STOP_ARGS

Arguments to be passed to the stop program to stop the service.

MQCA_STDERR_DESTINATION

Destination of standard error for the process.

MQCA_STDOUT_DESTINATION

Destination of standard output for the process.

MQCA_SERVICE_START_ARGS

Arguments to be passed to the service program.

MQIA_SERVICE_CONTROL

When the queue manager should start the service.

MQIA_SERVICE_TYPE

Mode in which the service is to run.

StringFilterCommand (MQCFSF)

String filter command descriptor. The parameter identifier must be any string type parameter allowed in *ServiceAttrs* except MQCA_SERVICE_NAME. Use this to restrict the output from the command by specifying a filter condition. See “MQCFSF - PCF string filter parameter” on page 434 for information about using this filter condition.

If you specify a string filter, you cannot also specify an integer filter using the *IntegerFilterCommand* parameter.

Inquire Service (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
		X	X	X	

The response to the Inquire Service (MQCMD_INQUIRE_SERVICE) command consists of the response header followed by the *ServiceName* structure and the requested combination of attribute parameter structures. If a generic service name was specified, one such message is generated for each service found.

Always returned:

ServiceName

Returned if requested:

AlterationDate, AlterationTime, Arguments, ServiceDesc, ServiceType, StartArguments, StartCommand, StartMode, StderrDestination, StdoutDestination, StopArguments, StopCommand

Response data

AlterationDate (MQCFST)

Alteration date (parameter identifier: MQCA_ALTERATION_DATE).

The date on which the information was last altered in the form yyyy-mm-dd.

AlterationTime (MQCFST)

Alteration time (parameter identifier: MQCA_ALTERATION_TIME).

The time at which the information was last altered in the form hh.mm.ss.

ServiceDesc (MQCFST)

Description of service definition (parameter identifier: MQCA_SERVICE_DESC).

The maximum length of the string is MQ_SERVICE_DESC_LENGTH.

ServiceName (MQCFST)

Name of service definition (parameter identifier: MQCA_SERVICE_NAME).

The maximum length of the string is MQ_SERVICE_NAME_LENGTH.

ServiceType (MQCFIN)

The mode in which the service is to run (parameter identifier: MQIA_SERVICE_TYPE).

The value can be:

MQSVC_TYPE_SERVER

Only one instance of the service can be executed at a time, with the status of the service made available by the Inquire Service Status command.

MQSVC_TYPE_COMMAND

Multiple instances of the service can be started.

StartArguments (MQCFST)

The arguments to be passed to the user program at queue manager startup (parameter identifier: MQCA_SERVICE_START_ARGS).

The maximum length of the string is MQ_SERVICE_ARGS_LENGTH.

StartCommand (MQCFST)

Service program name (parameter identifier: MQCA_SERVICE_START_COMMAND).

The name of the program which is to run.

The maximum length of the string is MQ_SERVICE_COMMAND_LENGTH.

StartMode (MQCFIN)

Service mode (parameter identifier: MQIA_SERVICE_CONTROL).

Specifies how the service is to be started and stopped. The value can be:

MQSVC_CONTROL_MANUAL

The service is not to be started automatically or stopped automatically.
It is to be controlled by user command.

MQSVC_CONTROL_Q_MGR

The service is to be started and stopped at the same time as the queue manager is started and stopped.

MQSVC_CONTROL_Q_MGR_START

The service is to be started at the same time as the queue manager is started, but is not requested to stop when the queue manager is stopped.

StderrDestination (MQCFST)

The path to a file to which the standard error (stderr) of the service program is to be redirected (parameter identifier: MQCA_STDERR_DESTINATION).

The maximum length of the string is MQ_SERVICE_PATH_LENGTH.

StdoutDestination (MQCFST)

The path to a file to which the standard output (stdout) of the service program is to be redirected (parameter identifier: MQCA_STDOUT_DESTINATION).

The maximum length of the string is MQ_SERVICE_PATH_LENGTH.

StopArguments (MQCFST)

The arguments to be passed to the stop program when instructed to stop the service (parameter identifier: MQCA_SERVICE_STOP_ARGS).

The maximum length of the string is MQ_SERVICE_ARGS_LENGTH.

StopCommand (MQCFST)

Service program stop command (parameter identifier: MQCA_SERVICE_STOP_COMMAND).

This is the name of the program that is to run when the service is requested to stop.

The maximum length of the string is MQ_SERVICE_COMMAND_LENGTH.

Inquire Service Status

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
		X	X	X	

The Inquire Service Status (MQCMD_INQUIRE_SERVICE_STATUS) command inquires about the status of one or more WebSphere MQ service instances.

Required parameters:

ServiceName

Optional parameters:

IntegerFilterCommand, ServiceStatusAttrs, StringFilterCommand

Required parameters

ServiceName (MQCFST)

Service name (parameter identifier: MQCACH_SERVICE_NAME).

Generic service names are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all services having names that start with the selected character string. An asterisk on its own matches all possible names.

The service name is always returned, regardless of the attributes requested.

The maximum length of the string is MQ_OBJECT_NAME_LENGTH.

Optional parameters

IntegerFilterCommand (MQCFIF)

Integer filter command descriptor. The parameter identifier must be any integer type parameter allowed in *ServiceStatusAttrs* except MQIACF_ALL. Use this to restrict the output from the command by specifying a filter condition. See “MQCFIF - PCF integer filter parameter” on page 427 for information about using this filter condition.

If you specify an integer filter, you cannot also specify a string filter using the *StringFilterCommand* parameter.

ServiceStatusAttrs (MQCFIL)

Service status attributes (parameter identifier: MQIACF_SERVICE_STATUS_ATTRS).

The attribute list can specify the following on its own (this is the default value used if the parameter is not specified):

MQIACF_ALL

All attributes.

or a combination of the following:

MQCA_SERVICE_DESC

Description of service definition.

MQCA_SERVICE_NAME

Name of service definition.

MQCA_SERVICE_START_ARGS

The arguments to pass to the service program.

MQCA_SERVICE_START_COMMAND

The name of the program to run to start the service.

MQCA_SERVICE_STOP_ARGS

The arguments to pass to the stop command to stop the service.

MQCA_SERVICE_STOP_COMMAND

The name of the program to run to stop the service.

MQCA_STDERR_DESTINATION

Destination of standard error for the process.

MQCA_STDOUT_DESTINATION

Destination of standard output for the process.

MQCACF_SERVICE_START_DATE

The date on which the service was started.

MQCACF_SERVICE_START_TIME

The time at which the service was started.

MQIA_SERVICE_CONTROL

How the service is to be started and stopped.

MQIA_SERVICE_TYPE

The mode in which the service is to run.

MQIACF_PROCESS_ID

The process identifier of the operating system task under which this service is executing.

MQIACF_SERVICE_STATUS

Current status of the service.

***StringFilterCommand* (MQCFSF)**

String filter command descriptor. The parameter identifier must be any string type parameter allowed in *ServiceStatusAttrs* except MQCA_SERVICE_NAME. Use this to restrict the output from the command by specifying a filter condition. See “MQCFSF - PCF string filter parameter” on page 434 for information about using this filter condition.

If you specify a string filter, you cannot also specify an integer filter using the *IntegerFilterCommand* parameter.

Error codes

This command might return the following in the response format header, in addition to the values shown on page “Error codes applicable to all commands” on page 27.

***Reason* (MQLONG)**

The value can be:

MQRCCF_SERV_STATUS_NOT_FOUND

Service status not found.

Inquire Service Status (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
		X	X	X	

The response to the Inquire Service Status (MQCMD_INQUIRE_SERVICE_STATUS) command consists of the response header followed by the *ServiceName* structure and the requested combination of attribute parameter structures. If a generic service name was specified, one such message is generated for each service found.

Always returned:

ServiceName

Returned if requested:

ProcessId, *ServiceDesc*, *StartArguments*, *StartCommand*, *StartDate*, *StartMode*, *StartTime*, *Status*, *StderrDestination*, *StdoutDestination*, *StopArguments*, *StopCommand*

Response data

***ProcessId* (MQCFIN)**

Process identifier (parameter identifier: MQIACF_PROCESS_ID).

The operating system process identifier associated with the service.

ServiceDesc (**MQCFST**)

Description of service definition (parameter identifier:
MQCACH_SERVICE_DESC).

The maximum length of the string is MQ_SERVICE_DESC_LENGTH.

ServiceName (**MQCFST**)

Name of the service definition (parameter identifier:
MQCA_SERVICE_NAME).

The maximum length of the string is MQ_OBJECT_NAME_LENGTH.

StartArguments (**MQCFST**)

Arguments to be passed to the program on startup (parameter identifier:
MQCA_SERVICE_START_ARGS).

The maximum length of the string is MQ_SERVICE_ARGS_LENGTH.

StartCommand (**MQCFST**)

Service program name (parameter identifier:
MQCA_SERVICE_START_COMMAND).

Specifies the name of the program which is to run.

The maximum length of the string is MQ_SERVICE_COMMAND_LENGTH.

StartDate (**MQCFST**)

Start date (parameter identifier: MQIACH_SERVICE_START_DATE).

The date, in the form yyyy-mm-dd, on which the service was started.

The maximum length of the string is MQ_DATE_LENGTH

StartMode (**MQCFIN**)

Service mode (parameter identifier: MQIACH_SERVICE_CONTROL).

How the service is to be started and stopped. The value can be:

MQSVC_CONTROL_MANUAL

The service is not to be started automatically or stopped automatically.
It is to be controlled by user command.

MQSVC_CONTROL_Q_MGR

The service is to be started and stopped at the same time as the queue
manager is started and stopped.

MQSVC_CONTROL_Q_MGR_START

The service is to be started at the same time as the queue manager is
started, but is not request to stop when the queue manager is stopped.

StartTime (**MQCFST**)

Start date (parameter identifier: MQIACH_SERVICE_START_TIME).

The time, in the form hh.mm.ss, at which the service was started.

The maximum length of the string is MQ_TIME_LENGTH

Status (**MQCFIN**)

Service status (parameter identifier: MQIACH_SERVICE_STATUS).

The current status of the service. The value can be:

MQSVC_STATUS_STARTING

The service is in the process of initializing.

MQSVC_STATUS_RUNNING

The service is running.

MQSVC_STATUS_STOPPING

The service is stopping.

StderrDestination (MQCFST)

Specifies the path to a file to which the standard error (stderr) of the service program is to be redirected (parameter identifier: MQCA_STDERR_DESTINATION).

The maximum length of the string is MQ_SERVICE_PATH_LENGTH.

StdoutDestination (MQCFST)

Specifies the path to a file to which the standard output (stdout) of the service program is to be redirected (parameter identifier: MQCA_STDOUT_DESTINATION).

The maximum length of the string is MQ_SERVICE_PATH_LENGTH.

StopArguments (MQCFST)

Specifies the arguments to be passed to the stop program when instructed to stop the service (parameter identifier: MQCA_SERVICE_STOP_ARGS).

The maximum length of the string is MQ_SERVICE_ARGS_LENGTH.

StopCommand (MQCFST)

Service program stop command (parameter identifier: MQCA_SERVICE_STOP_COMMAND).

This is the name of the program that is to run when the service is requested to stop.

The maximum length of the string is MQ_SERVICE_COMMAND_LENGTH.

Inquire Storage Class

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The Inquire Storage Class (MQCMD_INQUIRE_STG_CLASS) command returns information about storage classes.

Required parameters:

StorageClassName

Optional parameters:

*CommandScope, IntegerFilterCommand, PageSetId, PassTicketApplication,
QSGDisposition, StgClassAttrs, StringFilterCommand*

Required parameters

StorageClassName (MQCFST)

Storage class name (parameter identifier: MQCA_STORAGE_CLASS).

Generic storage class names are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all storage classes having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length of the string is MQ_STORAGE_CLASS_LENGTH.

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

You cannot use *CommandScope* as a parameter to filter on.

IntegerFilterCommand (MQCFIF)

Integer filter command descriptor. The parameter identifier must be any integer type parameter allowed in *StgClassAttrs* except MQIACF_ALL. Use this to restrict the output from the command by specifying a filter condition. See “MQCFIF - PCF integer filter parameter” on page 427 for information about using this filter condition.

If you specify an integer filter for *PageSetId*, you cannot also specify the *PageSetId* parameter.

If you specify an integer filter, you cannot also specify a string filter using the *StringFilterCommand* parameter.

PageSetId (MQCFIN)

Page set identifier that the storage class is associated with (parameter identifier: MQIA_PAGESET_ID).

If you omit this parameter, storage classes with any page set identifiers qualify.

QSGDisposition (MQCFIN)

Disposition of the object within the group (parameter identifier: MQIA_QSG_DISP).

Specifies the disposition of the object (that is, where it is defined and how it behaves). The value can be:

MQQSGD_LIVE

The object is defined as MQQSGD_Q_MGR or MQQSGD_COPY. This is the default value if the parameter is not specified.

MQQSGD_ALL

The object is defined as MQQSGD_Q_MGR or MQQSGD_COPY.

If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this option also displays information for objects defined with MQQSGD_GROUP.

If MQQSGD_LIVE is specified or defaulted, or if MQQSGD_ALL is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

MQQSGD_COPY

The object is defined as MQQSGD_COPY.

MQQSGD_GROUP

The object is defined as MQQSGD_GROUP. This is permitted only in a shared queue environment.

MQQSGD_Q_MGR

The object is defined as MQQSGD_Q_MGR.

MQQSGD_PRIVATE

The object is defined with either MQQSGD_Q_MGR or MQQSGD_COPY. Note that MQQSGD_PRIVATE returns the same information as MQQSGD_LIVE.

You cannot use *QSGDisposition* as a parameter to filter on.

***StgClassAttrs* (MQCFIL)**

Storage class parameter attributes (parameter identifier: MQIACF_STORAGE_CLASS_ATTRS).

The attribute list might specify the following on its own (this is the default value used if the parameter is not specified):

MQIACF_ALL

All attributes.

or a combination of the following:

MQCA_STORAGE_CLASS

Storage class name.

MQCA_STORAGE_CLASS_DESC

Description of the storage class.

MQIA_PAGESET_ID

The page set identifier to which the storage class maps.

MQCA_XCF_GROUP_NAME

The name of the XCF group of which WebSphere MQ is a member.

MQIA_XCF_MEMBER_NAME

The XCF member name of the IMS system within the XCF group specified in MQCA_XCF_GROUP_NAME.

MQCA_ALTERATION_DATE

The date on which the definition was last altered.

MQCA_ALTERATION_TIME

The time at which the definition was last altered.

***StringFilterCommand* (MQCFSF)**

String filter command descriptor. The parameter identifier must be any string type parameter allowed in *StgClassAttrs* except MQCA_STORAGE_CLASS. Use this to restrict the output from the command by specifying a filter condition. See “MQCFSF - PCF string filter parameter” on page 434 for information about using this filter condition.

If you specify a string filter, you cannot also specify an integer filter using the *IntegerFilterCommand* parameter.

Inquire Storage Class (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The response to the Inquire Storage Class (MQCMD_INQUIRE_STG_CLASS) command consists of the response header followed by:

- The *StgClassName* structure
- The *PageSetId* structure
- The *QSGDisposition* structure

which are followed by the requested combination of attribute parameter structures.

Always returned:

PageSetId, QSGDisposition, StgClassName

Returned if requested:

*AlterationDate, AlterationTime, PassTicketApplication,
StorageClassDesc, XCFGGroupName, XCFMemberName,*

Response data

AlterationDate (MQCFST)

Alteration date (parameter identifier: MQCA_ALTERATION_DATE).

This is the date, in the form yyyy-mm-dd, on which the definition was last altered.

The maximum length of the string is MQ_DATE_LENGTH.

AlterationTime (MQCFST)

Alteration time (parameter identifier: MQCA_ALTERATION_TIME).

This is the time, in the form hh.mm.ss, at which the definition was last altered.

The maximum length of the string is MQ_TIME_LENGTH.

PageSetId (MQCFIN)

Page set identifier (parameter identifier: MQIA_PAGESET_ID).

The page set identifier to which the storage class maps.

PassTicketApplication (MQCFST)

Pass ticket application (parameter identifier: MQCA_PASS_TICKET_APPL).

The application name that is passed to RACF when authenticating the passticket specified in the MQIIH header.

The maximum length is MQ_PASS_TICKET_APPL_LENGTH.

QSGDisposition (MQCFIN)

QSG disposition (parameter identifier: MQIA_QSG_DISP).

Specifies the disposition of the object (that is, where it is defined and how it behaves). The value can be:

MQQSGD_COPY

The object is defined as MQQSGD_COPY.

MQQSGD_GROUP

The object is defined as MQQSGD_GROUP.

MQQSGD_Q_MGR

The object is defined as MQQSGD_Q_MGR.

***StorageClassDesc* (MQCFST)**

Description of the storage class (parameter identifier: MQCA_STORAGE_CLASS_DESC).

The maximum length is MQ_STORAGE_CLASS_LENGTH.

***StgClassName* (MQCFST)**

Name of the storage class (parameter identifier: MQCA_STORAGE_CLASS).

The maximum length of the string is MQ_STORAGE_CLASS_LENGTH.

***XCFGroupName* (MQCFST)**

Name of the XCF group of which WebSphere MQ is a member (parameter identifier: MQCA_XCF_GROUP_NAME).

The maximum length is MQ_XCF_GROUP_NAME_LENGTH.

***XCFMemberName* (MQCFST)**

Name of the XCF group of which WebSphere MQ is a member (parameter identifier: MQCA_XCF_MEMBER_NAME).

The maximum length is MQ_XCF_MEMBER_NAME_LENGTH.

Inquire Storage Class Names

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The Inquire Storage Class Names (MQCMD_INQUIRE_STG_CLASS_NAMES) command inquires a list of storage class names that match the generic storage class name specified.

Required parameters:

StorageClassName

Optional parameters:

CommandScope, *QSGDisposition*

Required parameters

***StorageClassName* (MQCFST)**

Storage class name (parameter identifier: MQCA_STORAGE_CLASS).

Generic storage class names are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all storage classes having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length of the string is MQ_STORAGE_CLASS_LENGTH.

Optional parameters

***CommandScope* (MQCFST)**

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

QSGDisposition (MQCFIN)

Disposition of the object within the group (parameter identifier: MQIA_QSG_DISP). This parameter applies to z/OS only.

Specifies the disposition of the object (that is, where it is defined and how it behaves). The value can be:

MQQSGD_LIVE

The object is defined as MQQSGD_Q_MGR or MQQSGD_COPY. This is the default value if the parameter is not specified.

MQQSGD_ALL

The object is defined as MQQSGD_Q_MGR or MQQSGD_COPY.

If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this option also displays information for objects defined with MQQSGD_GROUP.

If MQQSGD_LIVE is specified or defaulted, or if MQQSGD_ALL is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

MQQSGD_COPY

The object is defined as MQQSGD_COPY.

MQQSGD_GROUP

The object is defined as MQQSGD_GROUP.

MQQSGD_Q_MGR

The object is defined as MQQSGD_Q_MGR.

MQQSGD_PRIVATE

The object is defined with either MQQSGD_Q_MGR or MQQSGD_COPY. Note that MQQSGD_PRIVATE returns the same information as MQQSGD_LIVE.

Inquire Storage Class Names (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The response to the Inquire Storage Class Names (MQCMD_INQUIRE_STG_CLASS_NAMES) command consists of the response header followed by a parameter structure giving zero or more names that match the specified namelist name.

In addition to this, the *QSGDispositions* structure (with the same number of entries as the *StorageClassNames* structure) is returned. Each entry in this structure indicates the disposition of the object with the corresponding entry in the *StorageClassNames* structure.

Always returned:

StorageClassNames, QSGDispositions

Returned if requested:

None

Response data

StorageClassNames (MQCFSL)

List of storage class names (parameter identifier: MQCACF_STORAGE_CLASS_NAMES).

QSGDispositions (MQCFIL)

List of QSG dispositions (parameter identifier: MQIACF_QSG_DISPS). Possible values for fields in this structure are those permitted for the *QSGDisposition* parameter (MQQSGD_*). Possible values for fields in this structure are:

MQQSGD_COPY

The object is defined as MQQSGD_COPY.

MQQSGD_GROUP

The object is defined as MQQSGD_GROUP.

MQQSGD_Q_MGR

The object is defined as MQQSGD_Q_MGR.

Inquire System

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The Inquire System (MQCMD_INQUIRE_SYSTEM) command returns general system parameters and information.

Required parameters:

None

Optional parameters:

CommandScope

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

Inquire System (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The response to the Inquire System (MQCMD_INQUIRE_SYSTEM) command consists of the response header followed by the *ParameterType* structure and the combination of attribute parameter structures determined by the value of the parameter type.

Always returned:

ParameterType

Possible values of *ParameterType* are:

MQSYSP_TYPE_INITIAL

The initial settings of the system parameters.

MQSYSP_TYPE_SET

The settings of the system parameters if they have been altered since their initial setting.

Returned if *ParameterType* is MQSYSP_TYPE_INITIAL or MQSYSP_TYPE_SET (and a value is set):

CheckpointCount, ClusterCacheType, CodedCharSetId, CommandUserId, DB2BlobTasks, DB2Name, DB2Tasks, DSGName, ExitInterval, ExitTasks, MaxConnects, MaxConnectsBackground, MaxConnectsForeground, OTMADruExit, OTMAGroup, OTMAInterval, OTMAMember, OTMSTpipePrefix, QIndexDefer, QSGName, RESLEVELAudit, RoutingCode, Service, SMFAccounting, SMFStatistics, SMFInterval, TraceClass, TraceSize, WLMInterval, WLMIntervalUnits

Response data

CheckpointCount (MQCFIN)

The number of log records written by WebSphere MQ between the start of one checkpoint and the next (parameter identifier: MQIACF_SYS_P_CHKPOINT_COUNT).

ClusterCacheType (MQCFIN)

The type of the cluster cache (parameter identifier: MQIACF_SYS_P_CLUSTER_CACHE).

The value can be:

MQCLCT_STATIC

Static cluster cache.

MQCLCT_DYNAMIC

Dynamic cluster cache.

CodedCharSetId (MQCFIN)

Archive retention period (parameter identifier: MQIA_CODED_CHAR_SET_ID).

The coded character set identifier for the queue manager.

CommandUserId (MQCFST)

Command user ID (parameter identifier: MQCACF_SYSP_CMD_USER_ID).

Specifies the default user ID for command security checks.

The maximum length of the string is MQ_USER_ID_LENGTH.

DB2BlobTasks (MQCFIN)

The number of DB2 server tasks to be used for BLOBs (parameter identifier: MQIACF_SYSP_DB2_BLOB_TASKS).

DB2Name (MQCFST)

The name of the DB2 subsystem or group attachment to which the queue manager is to connect (parameter identifier: MQCACF_DB2_NAME).

The maximum length of the string is MQ_DB2_NAME_LENGTH.

DB2Tasks (MQCFIN)

The number of DB2 server tasks to use (parameter identifier: MQIACF_SYSP_DB2_TASKS).

DSGName (MQCFST)

The name of the DB2 data-sharing group to which the queue manager is to connect (parameter identifier: MQCACF_DSG_NAME).

The maximum length of the string is MQ_DSG_NAME_LENGTH.

ExitInterval (MQCFIN)

The time, in seconds, for which queue manager exits can execute during each invocation (parameter identifier: MQIACF_SYSP_EXIT_INTERVAL).

ExitTasks (MQCFIN)

Specifies how many started server tasks to use to run queue manager exits (parameter identifier: MQIACF_SYSP_EXIT_TASKS).

MaxConnects (MQCFIN)

The maximum number of connections from batch, CICS, IMS, and TSO tasks to a single instance of WebSphere MQ (parameter identifier: MQIACF_SYSP_MAX_CONNS).

MaxConnectsBackground (MQCFIN)

The maximum number of connections from batch or TSO background tasks to a single instance of WebSphere MQ (parameter identifier: MQIACF_SYSP_MAX_CONNS_BACK).

MaxConnectsForeground (MQCFIN)

The maximum number of connections from TSO foreground tasks to a single instance of WebSphere MQ (parameter identifier: MQIACF_SYSP_MAX_CONNS_FORE).

OTMADruExit (MQCFST)

The name of the OTMA destination resolution user exit to be run by IMS (parameter identifier: MQCACF_SYSP_OTMA_DRU_EXIT).

The maximum length of the string is MQ_EXIT_NAME_LENGTH.

OTMAGroup (**MQCFST**)

The name of the XCF group to which this instance of WebSphere MQ belongs (parameter identifier: MQCACF_SYSP_OTMA_GROUP).

The maximum length of the string is MQ_XCF_GROUP_NAME_LENGTH.

OTMAInterval (**MQCFIN**)

The length of time, in seconds, that a user ID from WebSphere MQ is considered previously verified by IMS (parameter identifier: MQIACF_SYSP_OTMA_INTERVAL).

OTMAMember (**MQCFST**)

The name of the XCF member to which this instance of WebSphere MQ belongs (parameter identifier: MQCACF_SYSP_OTMA_MEMBER).

The maximum length of the string is MQ_XCF_MEMBER_NAME_LENGTH.

OTMSTpipePrefix (**MQCFST**)

The prefix to be used for Tpipe names (parameter identifier: MQCACF_SYSP_OTMA_TPIPE_PFX).

The maximum length of the string is MQ_TPIPE_PFX_LENGTH.

QIndexDefer (**MQCFIN**)

Specifies whether queue manager restart completes before all indexes are built deferring building to later, or waits until all indexes are built (parameter identifier: MQIACF_SYSP_Q_INDEX_DEFER).

The value can be:

MQSYSP_YES

Queue manager restart completes before all indexes are built.

MQSYSP_NO

Queue manager restart waits until all indexes are built.

QSGName (**MQCFST**)

The name of the queue-sharing group to which the queue manager belongs (parameter identifier: MQCA_QSG_NAME).

The maximum length of the string is MQ_QSG_NAME_LENGTH.

RESLEVELAudit (**MQCFIN**)

Specifies whether RACF audit records are written for RESLEVEL security checks performed during connection processing (parameter identifier: MQIACF_SYSP_RESLEVEL_AUDIT).

The value can be:

MQSYSP_YES

RACF audit records are written.

MQSYSP_NO

RACF audit records are not written.

RoutingCode (**MQCFIL**)

z/OS routing code list (parameter identifier: MQIACF_SYSP_ROUTING_CODE).

Specifies the list of z/OS routing codes for messages that are not sent in direct response to an MQSC command. There can be between 1 and 16 entries in the list.

Service (MQCFST)

Service parameter setting (parameter identifier: MQCACF_SYSP_SERVICE).

The maximum length of the string is MQ_SERVICE_NAME_LENGTH.

SMFAccounting (MQCFIN)

Specifies whether WebSphere MQ sends accounting data to SMF automatically when the queue manager starts (parameter identifier: MQIACF_SYSP_SMF_ACCOUNTING).

The value can be:

MQSYSP_YES

Accounting data is sent automatically.

MQSYSP_NO

Accounting data is not sent automatically.

SMFStatistics (MQCFIN)

Specifies whether WebSphere MQ sends statistics data to SMF automatically when the queue manager starts (parameter identifier: MQIACF_SYSP_SMF_STATS).

The value can be:

MQSYSP_YES

Statistics data is sent automatically.

MQSYSP_NO

Statistics data is not sent automatically.

SMFInterval (MQCFIN)

The default time, in minutes, between each gathering of statistics (parameter identifier: MQIACF_SYSP_SMF_INTERVAL).

TraceClass (MQCFIL)

Classes for which tracing is started automatically (parameter identifier: MQIACF_SYSP_TRACE_CLASS). There can be between 1 and 4 entries in the list.

TraceSize (MQCFIN)

The size of the trace table, in 4 KB blocks, to be used by the global trace facility (parameter identifier: MQIACF_SYSP_TRACE_SIZE).

WLMinInterval (MQCFIN)

The time between scans of the queue index for WLM-managed queues (parameter identifier: MQIACF_SYSP_WLM_INTERVAL).

WLMinIntervalUnits (MQCFIN)

Whether the value of *WLMinInterval* is given in seconds or minutes (parameter identifier: MQIACF_SYSP_WLM_INT_UNITS). The value can be:

MQTIME_UNITS_SEC

The value of *WLMinInterval* is given in seconds.

MQTIME_UNITS_MINS

The value of *WLMinInterval* is given in minutes.

Inquire Usage

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The Inquire Usage (MQCMD_INQUIRE_USAGE) command inquires about the current state of a page set, or information about the log data sets.

Required parameters:

None

Optional parameters:

CommandScope, PageSetId, UsageType

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

PageSetId (MQCFIN)

Page set identifier (parameter identifier: MQIA_PAGESET_ID). If you omit this parameter, all page set identifiers are returned.

UsageType (MQCFIN)

The type of information to be returned (parameter identifier: MQIACF_USAGE_TYPE).

The value can be:

MQIACF_USAGE_PAGESET

Return page set and buffer pool information.

MQIACF_USAGE_DATA_SET

Return data set information for log data sets.

MQIACF_USAGE_ALL

Return page set and data set information.

Inquire Usage (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The response to the Inquire Usage (MQCMD_INQUIRE_USAGE) command consists of the response header followed by the *UsageType* structure and a set of attribute parameter structures determined by the value of *UsageType* in the Inquire command.

Always returned:

UsageType

Possible values of *ParameterType* are:

MQIACF_USAGE_PAGESET

Page set information.

MQIACF_USAGE_BUFFER_POOL

Buffer pool information.

MQIACF_USAGE_DATA_SET

Data set information for log data sets.

Returned if *UsageType* is MQIACF_USAGE_PAGESET:

BufferPoolId, ExpandCount, ExpandType, LogRBA, NonPersistentDataPages, PageSetId, PageSetStatus, PersistentDataPages, TotalPages, UnusedPages

Returned if *UsageType* is MQIACF_USAGE_BUFFER_POOL:

BufferPoolId, TotalBuffers

Returned if *UsageType* is MQIACF_USAGE_DATA_SET:

DataSetName, DataSetType, LogRBA, LogLSN

Response data if *UsageType* is MQIACF_USAGE_PAGESET

***BufferPoolId* (MQCFIN)**

Buffer pool identifier (parameter identifier: MQIACF_BUFFER_POOL_ID).

This identifies the buffer pool being used by the page set.

***ExpandCount* (MQCFIN)**

The number of times the page set has been dynamically expanded since restart (parameter identifier: MQIACF_USAGE_EXPAND_COUNT).

***ExpandType* (MQCFIN)**

How the queue manager expands a page set when it becomes nearly full, and further pages are required within it (parameter identifier: MQIACF_USAGE_EXPAND_TYPE).

The value can be:

MQUSAGE_EXPAND_NONE

No further page set expansion is to take place.

MQUSAGE_EXPAND_USER

The secondary extent size that was specified when the page set was defined is used. If no secondary extent size was specified, or it was specified as zero, then no dynamic page set expansion can take place.

At restart, if a previously used page set has been replaced with a data set that is smaller, it is expanded until it reaches the size of the previously used data set. Only one extent is required to reach this size.

MQUSAGE_EXPAND_SYSTEM

A secondary extent size that is approximately 10 per cent of the current size of the page set is used. This may be rounded up to the nearest cylinder of DASD.

NonPersistentDataPages (MQCFIN)

The number of pages holding nonpersistent data (parameter identifier: MQIACF_USAGE_NONPERSIST_PAGES).

These pages are being used to store nonpersistent message data.

PageSetId (MQCFIN)

Page set identifier (parameter identifier: MQIA_PAGESET_ID).

The string consists of two numeric characters, in the range 00 through 99.

PageSetStatus (MQCFIN)

Current status of the page set (parameter identifier: MQIACF_PAGESET_STATUS).

The value can be:

MQUSAGE_PS_AVAILABLE

The page set is available.

MQUSAGE_PS_DEFINED

The page set has been defined but has never been used.

MQUSAGE_PS_OFFLINE

The page set is currently not accessible by the queue manager, for example because the page set has not been defined to the queue manager.

MQUSAGE_PS_NOT_DEFINED

The command was issued for a specific page set that is not defined to the queue manager.

PersistentDataPages (MQCFIN)

The number of pages holding persistent data (parameter identifier: MQIACF_USAGE_PERSIST_PAGES).

These pages are being used to store object definitions and persistent message data.

TotalPages (MQCFIN)

The total number of 4 KB pages in the page set (parameter identifier: MQIACF_USAGE_TOTAL_PAGES).

UnusedPages (MQCFIN)

The number of pages that are not used (that is, available page sets) (parameter identifier: MQIACF_USAGE_UNUSED_PAGES).

Response data if UsageType is MQIACF_USAGE_BUFFER_POOL

BufferPoolId (MQCFIN)

Buffer pool identifier (parameter identifier: MQIACF_BUFFER_POOL_ID).

This identifies the buffer pool being used by the page set.

TotalBuffers (MQCFIN)

The number of buffers defined for specified buffer pool (parameter identifier: MQIACF_USAGE_TOTAL_BUFFERS).

Response data if UsageType is MQIACF_USAGE_DATA_SET

DataSetName (MQCFST)

Data set name (parameter identifier: MQCACF_DATA_SET_NAME).

The maximum length is MQ_DATA_SET_NAME_LENGTH.

DataSetType (MQCFIN)

The type of data set, and circumstance (parameter identifier: MQIACF_USAGE_DATA_SET_TYPE).

The value can be:

MQUSAGE_DS_OLDEST_ACTIVE_UOW

The log data set containing the start RBA of the oldest active unit of work for the queue manager

MQUSAGE_DS_OLDEST_PS_RECOVERY

The log data set containing the oldest restart RBA of any page set for the queue manager.

MQUSAGE_DS_OLDEST_CF_RECOVERY

The log data set containing the LRSN which matches the time of the oldest current backup of any CF structure in the queue-sharing group.

LogRBA (MQCFST)

Log RBA (parameter identifier: MQCACF_USAGE_LOG_RBA).

The maximum length is MQ_RBA_LENGTH.

LogLRSN (MQCFST)

Log LRSN (parameter identifier: MQIACF_USAGE_LOG_LRSN).

The length of the string is MQ_LRSN_LENGTH.

Move Queue

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The Move Queue (MQCMD_MOVE_Q) command moves all the messages from one local queue to another.

Required parameters:

FromQName

Optional parameters:

CommandScope, MoveType, QSGDisposition, ToQName

Required parameters

FromQName (MQCFST)

From queue name (parameter identifier: MQCACF_FROM_Q_NAME).

The name of the local queue from which messages are moved. The name must be defined to the local queue manager.

The command fails if the queue contains uncommitted messages.

If an application has this queue open, or has open a queue that eventually resolves to this queue, the command fails. For example, the command fails if this queue is a transmission queue, and any queue that is, or resolves to, a remote queue that references this transmission queue, is open.

An application can open this queue while the command is in progress but the application waits until the command has completed.

The maximum length of the string is MQ_Q_NAME_LENGTH.

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.

The maximum length is MQ_QSG_NAME_LENGTH.

MoveType (MQCFIN)

Move type (parameter identifier: MQIA_QSG_DISP).

Specifies how the messages are moved. The value can be:

MQIACF_MOVE_TYPE_MOVE

Move the messages from the source queue to the empty target queue.

The command fails if the target queue already contains one or more messages. The messages are deleted from the source queue. This is the default value.

MQIACF_MOVE_TYPE_ADD

Move the messages from the source queue and add them to any messages already on the target queue.

The messages are deleted from the source queue.

QSGDisposition (MQCFIN)

Disposition of the object within the group (parameter identifier: MQIA_QSG_DISP).

Specifies the disposition of the object for which information is to be returned (that is, where it is defined and how it behaves). The value can be:

MQQSGD_PRIVATE

The object is defined as either MQQSGD_Q_MGR or MQQSGD_COPY. This is the default value.

MQQSGD_SHARED

The object is defined as MQQSGD_SHARED. This is valid only in a shared queue environment.

ToQName (MQCFST)

To queue name (parameter identifier: MQCACF_TO_Q_NAME).

The name of the local queue to which messages are moved. The name must be defined to the local queue manager.

The name of the target queue can be the same as that of the source queue only if the queue exists as both a shared and a private queue. In this case, the command moves messages to the queue that has the opposite disposition (shared or private) from that specified for the source queue on the *QSGDisposition* parameter.

If an application has this queue open, or has open a queue that eventually resolves to this queue, the command fails. The command also fails if this queue is a transmission queue, and any queue that is, or resolves to, a remote queue that references this transmission queue, is open.

No application can open this queue while the command is in progress.

If you specify a value of MQIACF_MOVE_TYPE_MOVE on the *MoveType* parameter, the command fails if the target queue already contains one or more messages.

The *DefinitionType*, *HardenGetBackout*, *Usage* parameters of the target queue must be the same as those of the source queue.

The maximum length of the string is MQ_Q_NAME_LENGTH.

Ping Channel

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Ping Channel (MQCMD_PING_CHANNEL) command tests a channel by sending data as a special message to the remote message queue manager and checking that the data is returned. The data is generated by the local queue manager.

This command can only be used for channels with a *ChannelType* value of MQCHT_SENDER, MQCHT_SERVER, or MQCHT_CLUSSDR.

Where there is both a locally defined channel and an auto-defined cluster-sender channel of the same name, the command applies to the locally defined channel.

If there is no locally defined channel but more than one auto-defined cluster-sender channel, the command applies to the last channel added to the repository on the local queue manager.

The command is not valid if the channel is running; however it is valid if the channel is stopped or in retry mode.

Required parameters:

ChannelName

Optional parameters:

DataCount, CommandScope, ChannelDisposition

Required parameters

ChannelName (MQCFST)

Channel name (parameter identifier: MQCACH_CHANNEL_NAME).

The name of the channel to be tested. The maximum length of the string is MQ_CHANNEL_NAME_LENGTH.

Optional parameters

DataCount (MQCFIN)

Data count (parameter identifier: MQIACH_DATA_COUNT).

Specifies the length of the data.

Specify a value in the range 16 through 32 768. The default value is 64 bytes.

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

ChannelDisposition (MQCFIN)

Channel disposition (parameter identifier: MQIACH_CHANNEL_DISP). This parameter applies to z/OS only.

Specifies the disposition of the channels to be tested.

The value can be:

MQCHLD_PRIVATE

A receiving channel is private if it was started in response to an inbound transmission directed to the queue manager.

A sending channel is private if its transmission queue has a disposition other than MQQSGD_SHARED.

MQCHLD_SHARED

A receiving channel is shared if it was started in response to an inbound transmission directed to the queue-sharing group.

A sending channel is shared if its transmission queue has a disposition of MQQSGD_SHARED.

MQCHLD_FIXSHARED

Tests shared channels, tied to a specific queue manager.

The combination of the *ChannelDisposition* and *CommandScope* parameters also controls from which queue manager the channel is operated. The possible options are:

- On the local queue manager where the command is issued.
- On another specific named queue manager in the group.
- On the most suitable queue manager in the group, determined automatically by the queue manager itself.

The various combinations of *ChannelDisposition* and *CommandScope* are summarized in Table 10

Table 10. ChannelDisposition and CommandScope for PING CHANNEL

<i>ChannelDisposition</i>	<i>CommandScope</i> blank or local-qmgr	<i>CommandScope</i> qmgr-name	<i>CommandScope</i> (*)
MQCHLD_PRIVATE	Ping private channel on the local queue manager	Ping private channel on the named queue manager	Ping private channel on all active queue managers
MQCHLD_SHARED	Ping a shared channel on the most suitable queue manager in the group This might automatically generate a command using <i>CommandScope</i> and send it to the appropriate queue manager. If there is no definition for the channel on the queue manager to which the command is sent, or if the definition is unsuitable for the command, the command fails. The definition of a channel on the queue manager where the command is entered might be used to determine the target queue manager where the command is actually run. Therefore, it is important that channel definitions are consistent. Inconsistent channel definitions might result in unexpected command behavior.	Not permitted	Not permitted
MQCHLD_FIXSHARED	Ping a shared channel on the local queue manager	Ping a shared channel on the named queue manager	Not permitted

Error codes

This command might return the following in the response format header, in addition to the values shown on page "Error codes applicable to all commands" on page 27.

Reason (MQLONG)

The value can be:

MQRCCF_ALLOCATE_FAILED

Allocation failed.

MQRCCF_BIND_FAILED

Bind failed.

MQRCCF_CCSID_ERROR	Coded character-set identifier error.
MQRCCF_CHANNEL_CLOSED	Channel closed.
MQRCCF_CHANNEL_IN_USE	Channel in use.
MQRCCF_CHANNEL_NOT_FOUND	Channel not found.
MQRCCF_CHANNEL_TYPE_ERROR	Channel type not valid.
MQRCCF_CONFIGURATION_ERROR	Configuration error.
MQRCCF_CONNECTION_CLOSED	Connection closed.
MQRCCF_CONNECTION_REFUSED	Connection refused.
MQRCCF_DATA_TOO_LARGE	Data too large.
MQRCCF_ENTRY_ERROR	Connection name not valid.
MQRCCF_HOST_NOT_AVAILABLE	Remote system not available.
MQRCCF_NO_COMM_MANAGER	Communications manager not available.
MQRCCF_PING_DATA_COMPARE_ERROR	Ping Channel command failed.
MQRCCF_PING_DATA_COUNT_ERROR	Data count not valid.
MQRCCF_PING_ERROR	Ping error.
MQRCCF_RECEIVE_FAILED	Receive failed.
MQRCCF_RECEIVED_DATA_ERROR	Received data error.
MQRCCF_REMOTE_QM_TERMINATING	Remote queue manager terminating.
MQRCCF_REMOTE_QM_UNAVAILABLE	Remote queue manager not available.
MQRCCF_SEND_FAILED	Send failed.
MQRCCF_STRUCTURE_TYPE_ERROR	Structure type not valid.
MQRCCF_TERMINATED_BY_SEC_EXIT	Channel terminated by security exit.

MQRCCF_UNKNOWN_REMOTE_CHANNEL

Remote channel not known.

MQRCCF_USER_EXIT_NOT_AVAILABLE

User exit not available.

Ping Queue Manager

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	

The Ping Queue Manager (MQCMD_PING_Q_MGR) command tests whether the queue manager and its command server is responsive to commands. If the queue manager is responding a positive reply is returned.

Required parameters:

None

Optional parameters:

None

Recover CF Structure

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The Recover CF Structure (MQCMD_RECOVER_CF_STRUC) command initiates recovery of CF application structures.

Note: This command is valid only on z/OS when the queue manager is a member of a queue-sharing group.

Required parameters:

CFStrucName

Optional parameters:

CommandScope, *Purge*

Required parameters

CFStrucName (**MQCFST**)

CF application structure name (parameter identifier:
MQCA_CF_STRUC_NAME).

The maximum length of the string is MQ_CF_STRUC_NAME_LENGTH.

Optional parameters

CommandScope (**MQCFST**)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.

The maximum length is MQ_Q_MGR_NAME_LENGTH.

Purge (MQCFIN)

Recover to empty CF structure (parameter identifier: MQIACF_PURGE).

Specifies whether the CF application structure is emptied. The value can be:

MQPO_YES

Recover to empty CF structure. Any messages in the CF structure are lost.

MQPO_NO

Performs a true recovery of the CF structure. This is the default value.

Refresh Cluster

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Refresh Cluster (MQCMD_REFRESH_CLUSTER) command discards all locally held cluster information, including any auto-defined channels that are not in doubt, and forces the repository to be rebuilt.

Required parameters:

ClusterName

Optional parameters:

CommandScope, RefreshRepository

Required parameters

ClusterName (MQCFST)

Cluster name (parameter identifier: MQCA_CLUSTER_NAME).

The name of the cluster to be refreshed.

The maximum length of the string is MQ_CLUSTER_NAME_LENGTH.

This is the name of the cluster to be refreshed. If an asterisk (*) is specified for the name, the queue manager is refreshed in all the clusters to which it belongs.

If an asterisk (*) is specified with *RefreshRepository* set to MQCFO_REFRESH_REPOSITORY_YES, the queue manager restarts its search for repository queue managers, using information in the local cluster-sender channel definitions.

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.

The maximum length is MQ_QSG_NAME_LENGTH.

RefreshRepository (MQCFIN)

Whether repository information should be refreshed (parameter identifier: MQIACF_REFRESH_REPOSITORY).

This indicates whether the information about repository queue managers should be refreshed.

The value can be:

MQCFO_REFRESH_REPOSITORY_YES

Refresh repository information.

This value cannot be specified if the queue manager is itself a repository queue manager.

MQCFO_REFRESH_REPOSITORY_YES specifies that in addition to MQCFO_REFRESH_REPOSITORY_NO behavior, objects representing full repository cluster queue managers are also refreshed. Do not use this option if the queue manager is itself a full repository.

If it is a full repository, you must first alter it so that it is not a full repository for the cluster in question.

The full repository location is recovered from the manually defined cluster-sender channel definitions. After the refresh with MQCFO_REFRESH_REPOSITORY_YES has been issued the queue manager can be altered so that it is once again a full repository.

MQCFO_REFRESH_REPOSITORY

Do not refresh repository information. This is the default.

If you select MQCFO_REFRESH_REPOSITORY_YES, check that all cluster-sender channels in the relevant cluster are inactive or stopped before you issue the Refresh Cluster command. If there are cluster-sender channels running at the time when the Refresh is processed, and they are used exclusively by the cluster or clusters being refreshed and MQCFO_REFRESH_REPOSITORY_YES is used, the channels are stopped, by using the Stop Channel command with a value of MQMODE_FORCE in the *Mode* parameter if necessary.

This ensures that the Refresh can remove the channel state and that the channel will run with the refreshed version after the Refresh has completed. If a channel's state cannot be deleted, for example because it is in doubt, or

because it is also running as part of another cluster, its state is not new after the refresh and it does not automatically restart if it was stopped.

Refresh Queue Manager

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The Refresh Queue Manager (MQCMD_REFRESH_Q_MGR) to perform special operations on queue managers.

Required parameters:

RefreshType

Optional parameters:

CommandScope, ObjectName, ObjectType, RefreshInterval

Required parameters

RefreshType (MQCFIN)

Type of information to be refreshed (parameter identifier: MQIACF_REFRESH_TYPE).

Use this to specify the type of information to be refreshed. The value can be:

MQRT_CONFIGURATION

This causes the queue manager to generate configuration event messages for every object definition that matches the selection criteria specified by the *ObjectType*, *ObjectName*, and *RefreshInterval* parameters.

A Refresh Queue Manager command with a *RefreshType* value of MQRT_CONFIGURATION is generated automatically when the value of the queue manager's *ConfigurationEvent* parameter changes from MQEVR_DISABLED to MQEVR_ENABLED.

Use this command with a *RefreshType* of MQRT_CONFIGURATION to recover from problems such as errors on the event queue. In such cases, use appropriate selection criteria, to avoid excessive processing time and event message generation.

MQRT_EXPIRY

This requests that the queue manager performs a scan to discard expired messages for every queue that matches the selection criteria specified by the *ObjectName* parameter.

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.

- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

ObjectName (MQCFST)

Name of object to be included in the processing of this command (parameter identifier: MQCACF_OBJECT_NAME).

Use this to specify the name of the object to be included in the processing of this command.

Generic names are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all objects having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length is MQ_OBJECT_NAME_LENGTH.

ObjectType (MQCFIN)

Object type for which configuration data is to be refreshed (parameter identifier: MQIACF_OBJECT_TYPE).

Use this to specify the object type for which configuration data is to be refreshed. This parameter is valid only if the value of *RefreshType* is MQRT_CONFIGURATION. The default value, in that case, is MQOT_ALL. The value can be one of:

MQOT_AUTH_INFO

Authentication information object.

MQOT_CF_STRUC

CF structure.

MQOT_CHANNEL

Channel.

MQOT_NAMELIST

Namelist.

MQOT_PROCESS

Process definition.

MQOT_Q

Queue.

MQOT_LOCAL_Q

Local queue.

MQOT_MODEL_Q

Model queue.

MQOT_ALIAS_Q

Alias queue.

MQOT_REMOTE_Q

Remote queue.

MQOT_Q_MGR

Queue manager.

MQOT_CFSTRUC

CF structure.

MQOT_STORAGE_CLASS

Storage class.

RefreshInterval (MQCFIN)

Refresh interval (parameter identifier: MQIACF_REFRESH_INTERVAL).

Use this to specify a value, in minutes, defining a period immediately prior to the current time. This requests that only objects that have been created or altered within that period (as defined by their *AlterationDate* and *AlterationTime* attributes) are included.

Specify a value in the range zero through 999 999. A value of zero means there is no time limit (this is the default).

This parameter is valid only if the value of *RefreshType* is MQRT_CONFIGURATION.

Refresh Security

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
	X	X	X	X	X

The Refresh Security (MQCMD_REFRESH_SECURITY) command refreshes the list of authorizations held internally by the authorization service component.

Required parameters:

None

Optional parameters:*CommandScope, SecurityItem, SecurityType*

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

SecurityItem (MQCFIN)

Resource class for which the security refresh is to be performed (parameter identifier: MQIACF_SECURITY_ITEM). This parameter applies to z/OS only.

Use this to specify the resource class for which the security refresh is to be performed. The value can be:

MQSECITEM_ALL

A full refresh of the type specified is performed. This is the default value.

MQSECITEM_MQADMIN

Specifies that Administration type resources are to be refreshed. Valid only if the value of *SecurityType* is MQSECTYPE_CLASSES..

MQSECITEM_MQNLIST

Specifies that Namelist resources are to be refreshed. Valid only if the value of *SecurityType* is MQSECTYPE_CLASSES.

MQSECITEM_MQPROC

Specifies that Process resources are to be refreshed. Valid only if the value of *SecurityType* is MQSECTYPE_CLASSES.

MQSECITEM_MQQUEUE

Specifies that Queue resources are to be refreshed. Valid only if the value of *SecurityType* is MQSECTYPE_CLASSES.

SecurityType (MQCFIN)

Security type (parameter identifier: MQIACF_SECURITY_TYPE).

Use this to specify the type of security refresh to be performed. The value can be:

MQSECTYPE_AUTHSERV

The list of authorizations held internally by the authorization services component is refreshed. This is not valid on z/OS.

This is the default on platforms other than z/OS.

MQSECTYPE_CLASSES

Permits you to select specific resource classes for which to perform the security refresh.

This is valid only on z/OS where it is the default.

MQSECTYPE_SSL

This refreshes the locations of:

- The LDAP servers to be used for Certified Revocation Lists
- The key repository

as well as any cryptographic hardware parameters specified through WebSphere MQ. It also refreshes the cached view of the Secure Sockets Layer key repository and allows updates to become effective on successful completion of the command.

This updates all SSL channels currently running, as follows:

- Sender, server and cluster-sender channels using SSL are allowed to complete the current batch. In general, they then run the SSL handshake again with the refreshed view of the SSL key repository. However, you must manually restart a requester-server channel on which the server definition has no CONNAME parameter.

- All other channel types using SSL are stopped with a STOP CHANNEL MODE(FORCE) STATUS(INACTIVE) command. If the partner end of the stopped MCA channel has retry values defined, the channel retries and the new SSL handshake uses the refreshed view of the contents of the SSL key repository, the location of the LDAP server to be used for Certification Revocation Lists, and the location of the key repository. In the case of a server-connection channel, the client application loses its connection to the queue manager and has to reconnect in order to continue.

Reset Channel

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Reset Channel (MQCMD_RESET_CHANNEL) command resets the message sequence number for a WebSphere MQ channel with, optionally, a specified sequence number to be used the next time that the channel is started.

This command can be issued to a channel of any type (except MQCHT_SVRCONN and MQCHT_CLNTCONN). However, if it is issued to a sender (MQCHT_SENDER), server (MQCHT_SERVER), or cluster-sender (MQCHT_CLUSSDR) channel, the value at both ends (issuing end and receiver or requester end), is reset when the channel is next initiated or resynchronized. The value at both ends is reset to be equal.

If the command is issued to a receiver (MQCHT_RECEIVER), requester (MQCHT_REQUESTER), or cluster-requester (MQCHT_CLUSRCVR) channel, the value at the other end is *not* reset as well; this must be done separately if necessary.

Where there is both a locally defined channel and an auto-defined cluster-sender channel of the same name, the command applies to the locally defined channel.

If there is no locally defined channel but more than one auto-defined cluster-sender channel, the command applies to the last channel added to the repository on the local queue manager.

Required parameters:

ChannelName

Optional parameters:

CommandScope, *ChannelDisposition*, *MsgSeqNumber*

Required parameters

ChannelName (MQCFST)

Channel name (parameter identifier: MQCACH_CHANNEL_NAME).

The name of the channel to be reset. The maximum length of the string is MQ_CHANNEL_NAME_LENGTH.

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.

The maximum length is MQ_QSG_NAME_LENGTH.

ChannelDisposition (MQCFIN)

Channel disposition (parameter identifier: MQIACH_CHANNEL_DISP). This parameter applies to z/OS only.

Specifies the disposition of the channels to be reset.

The value can be:

MQCHLD_PRIVATE

A receiving channel is private if it was started in response to an inbound transmission directed to the queue manager.

A sending channel is private if its transmission queue has a disposition other than MQQSGD_SHARED.

MQCHLD_SHARED

A receiving channel is shared if it was started in response to an inbound transmission directed to the queue-sharing group.

A sending channel is shared if its transmission queue has a disposition of MQQSGD_SHARED.

The combination of the *ChannelDisposition* and *CommandScope* parameters also controls from which queue manager the channel is operated. The possible options are:

- On the local queue manager where the command is issued.
- On another specific named queue manager in the group.

The various combinations of *ChannelDisposition* and *CommandScope* are summarized in Table 11

Table 11. *ChannelDisposition* and *CommandScope* for RESET CHANNEL

<i>ChannelDisposition</i>	<i>CommandScope</i> blank or local-qmgr	<i>CommandScope</i> qmgr-name
MQCHLD_PRIVATE	Reset private channel on the local queue manager	Reset private channel on the named queue manager

Table 11. ChannelDisposition and CommandScope for RESET CHANNEL (continued)

<i>ChannelDisposition</i>	<i>CommandScope blank or local-qmgr</i>	<i>CommandScope qmgr-name</i>
MQCHLD_SHARED	<p>Reset a shared channel on all active queue managers.</p> <p>This might automatically generate a command using <i>CommandScope</i> and send it to the appropriate queue manager. If there is no definition for the channel on the queue manager to which the command is sent, or if the definition is unsuitable for the command, the command fails.</p> <p>The definition of a channel on the queue manager where the command is entered might be used to determine the target queue manager where the command is actually run. Therefore, it is important that channel definitions are consistent. Inconsistent channel definitions might result in unexpected command behavior.</p>	Not permitted

MsgSeqNumber (MQCFIN)

Message sequence number (parameter identifier: MQIACH_MSG_SEQUENCE_NUMBER).

Specifies the new message sequence number.

The value must be in the range 1 through 999 999 999. The default value is one.

Error codes

This command might return the following in the response format header, in addition to the values shown on page “Error codes applicable to all commands” on page 27.

Reason (MQLONG)

The value can be:

MQRCCF_CHANNEL_NOT_FOUND

Channel not found.

Reset Cluster

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Reset Cluster (MQCMD_RESET_CLUSTER) command forces a queue manager to leave a cluster.

Required parameters:

ClusterName, QMgrIdentifier or QMgrName, Action

Optional parameters:
CommandScope, RemoveQueues

Required parameters

ClusterName (MQCFST)

Cluster name (parameter identifier: MQCA_CLUSTER_NAME).

The name of the cluster to be reset.

The maximum length of the string is MQ_CLUSTER_NAME_LENGTH.

QMgrIdentifier (MQCFST)

Queue manager identifier (parameter identifier: MQCA_Q_MGR_IDENTIFIER).

This is the unique identifier of the queue manager to be forcibly removed from the cluster. Only one of QMgrIdentifier and QMgrName can be specified. Use QMgrIdentifier in preference to QmgrName, because QmgrName might not be unique.

QMgrName (MQCFST)

Queue manager name (parameter identifier: MQCA_Q_MGR_NAME).

This is the name of the queue manager to be forcibly removed from the cluster. Only one of QMgrIdentifier and QMgrName can be specified. Use QMgrIdentifier in preference to QmgrName, because QmgrName might not be unique.

Action (MQCFIN)

Action (parameter identifier: MQIACF_ACTION).

Specifies the action to take place. This can be requested only by a repository queue manager.

The value can be:

MQACT_FORCE_REMOVE

Requests that a queue manager is forcibly removed from a cluster.

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.

The maximum length is MQ_QSG_NAME_LENGTH.

RemoveQueues (MQCFIN)

Whether cluster queues should be removed from the cluster (parameter identifier: MQIACF_REMOVE_QUEUES).

This indicates whether the cluster queues that belong to the queue manager being removed from the cluster should be removed from the cluster. This parameter can be specified even if the queue manager identified by the *QMgrName* parameter is not currently in the cluster.

The value can be:

MQCFO_REMOVE_QUEUES_YES

Remove queues belonging to the queue manager being removed from the cluster.

MQCFO_REMOVE_QUEUES_NO

Do not remove queues belonging to the queue manager being removed. This is the default.

Error codes

This command might return the following in the response format header, in addition to the values shown on page “Error codes applicable to all commands” on page 27.

Reason (MQLONG)

The value can be:

MQRCCF_ACTION_VALUE_ERROR

Value not valid.

Reset Queue Manager

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
		X	X	X	

Use the Reset Queue Manager (MQCMD_RESET_Q_MGR) command as part of your backup and recovery procedures. You can use this command to request that the queue manager starts writing to a new log extent, making the previous log extent available for archiving.

Valid only on AIX, HP-UX, Linux, Solaris, i5/OS, and Windows.

Required parameters:

Action

Optional parameters:

None

Required parameters

Action (MQCFIN)

Action (parameter identifier: MQIACF_ACTION).

Specifies the action to take place.

The value can be:

MQACT_ADVANCE_LOG

Requests that the queue manager starts writing to a new log extent,

making the previous log extent available for archiving. This command is accepted only if the queue manager is configured to use linear logging.

MQACT_COLLECT_STATISTICS

Requests that the queue manager ends the current statistics collection period, and writes the statistics collected.

Error codes

This command might return the following in the response format header, in addition to the values shown on page "Error codes applicable to all commands" on page 27.

Reason (MQLONG)

The value can be:

MQRC_RESOURCE_PROBLEM

Insufficient system resources available.

Reset Queue Statistics

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Reset Queue Statistics (MQCMD_RESET_Q_STATS) command reports the performance data for a queue and then resets the performance data.

Performance data is maintained for each local queue (including transmission queues). It is reset at the following times:

- When a Reset Queue Statistics command is issued
- When the queue manager is restarted

Required parameters:

QName

Optional parameters:

CommandScope

Required parameters

QName (MQCFST)

Queue name (parameter identifier: MQCA_Q_NAME).

The name of the local queue to be tested and reset.

Generic queue names are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all objects having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length of the string is MQ_Q_NAME_LENGTH.

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

Error codes

This command might return the following in the response format header, in addition to the values shown on page “Error codes applicable to all commands” on page 27.

Reason (MQLONG)

The value can be:

MQRCCF_Q_WRONG_TYPE

Action not valid for the queue of specified type.

Reset Queue Statistics (Response)

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The response to the Reset Queue Statistics (MQCMD_RESET_Q_STATS) command consists of the response header followed by the *QName* structure and the attribute parameter structures shown below. If a generic queue name was specified, one such message is generated for each queue found.

Always returned:

HighQDepth, *MsgDeqCount*, *MsgEnqCount*, *QName*, *QSGDisposition*,
TimeSinceReset

Response data

HighQDepth (MQCFIN)

Maximum number of messages on a queue (parameter identifier: MQIA_HIGH_Q_DEPTH).

This count is the peak value of the *CurrentQDepth* local queue attribute since the last reset. The *CurrentQDepth* is incremented during an MPUT call, and

during backout of an MQGET call, and is decremented during a (nonbrowse) MQGET call, and during backout of an MQPUT call.

***MsgDqCount* (MQCFIN)**

Number of messages dequeued (parameter identifier: MQIA_MSG_DEQ_COUNT).

This count includes messages that have been successfully retrieved (with a nonbrowse MQGET) from the queue, even though the MQGET has not yet been committed. The count is not decremented if the MQGET is subsequently backed out.

On z/OS, if the value exceeds 999 999 999, it is returned as 999 999 999

***MsgEnqCount* (MQCFIN)**

Number of messages enqueued (parameter identifier: MQIA_MSG_ENQ_COUNT).

This count includes messages that have been put to the queue, but have not yet been committed. The count is not decremented if the put is subsequently backed out.

On z/OS, if the value exceeds 999 999 999, it is returned as 999 999 999

***QName* (MQCFST)**

Queue name (parameter identifier: MQCA_Q_NAME).

The maximum length of the string is MQ_Q_NAME_LENGTH.

***QSGDisposition* (MQCFIN)**

QSG disposition (parameter identifier: MQIA_QSG_DISP).

Specifies the disposition of the object (that is, where it is defined and how it behaves). This parameter is valid on z/OS only. The value can be:

MQQSGD_COPY

The object is defined as MQQSGD_COPY.

MQQSGD_SHARED

The object is defined as MQQSGD_SHARED.

MQQSGD_Q_MGR

The object is defined as MQQSGD_Q_MGR.

***TimeSinceReset* (MQCFIN)**

Time since statistics reset in seconds (parameter identifier: MQIA_TIME_SINCE_RESET).

Resolve Channel

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Resolve Channel (MQCMD_RESOLVE_CHANNEL) command requests a channel to commit or back out in-doubt messages.

This command is used when the other end of a link fails during the confirmation stage, and for some reason it is not possible to reestablish the connection. In this situation the sending end remains in an in-doubt state, as to whether or not the

messages were received. Any outstanding units of work must be resolved using Resolve Channel with either backout or commit.

Care must be exercised in the use of this command. If the resolution specified is not the same as the resolution at the receiving end, messages can be lost or duplicated.

This command can only be used for channels with a *ChannelType* value of MQCHT_SENDER, MQCHT_SERVER, or MQCHT_CLUSSDR.

Where there is both a locally defined channel and an auto-defined cluster-sender channel of the same name, the command applies to the locally defined channel.

If there is no locally defined channel but more than one auto-defined cluster-sender channel, the command applies to the last channel added to the repository on the local queue manager.

Required parameters:

ChannelName, InDoubt

Optional parameters:

CommandScope, ChannelDisposition

Required parameters

ChannelName (MQCFST)

Channel name (parameter identifier: MQCACH_CHANNEL_NAME).

The name of the channel to be resolved. The maximum length of the string is MQ_CHANNEL_NAME_LENGTH.

InDoubt (MQCFIN)

Indoubt resolution (parameter identifier: MQIACH_IN_DOUBT).

Specifies whether to commit or back out the in-doubt messages.

The value can be:

MQIDO_COMMIT

Commit.

MQIDO_BACKOUT

Backout.

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.

The maximum length is MQ_QSG_NAME_LENGTH.

ChannelDisposition (MQCFIN)

Channel disposition (parameter identifier: MQIACH_CHANNEL_DISP). This parameter applies to z/OS only.

Specifies the disposition of the channels to be resolved.

The value can be:

MQCHLD_PRIVATE

A receiving channel is private if it was started in response to an inbound transmission directed to the queue manager.

A sending channel is private if its transmission queue has a disposition other than MQQSGD_SHARED.

MQCHLD_SHARED

A receiving channel is shared if it was started in response to an inbound transmission directed to the queue-sharing group.

A sending channel is shared if its transmission queue has a disposition of MQQSGD_SHARED.

The combination of the *ChannelDisposition* and *CommandScope* parameters also controls from which queue manager the channel is operated. The possible options are:

- On the local queue manager where the command is issued.
- On another specific named queue manager in the group.

The various combinations of *ChannelDisposition* and *CommandScope* are summarized in Table 12

Table 12. ChannelDisposition and CommandScope for RESOLVE CHANNEL

<i>ChannelDisposition</i>	<i>CommandScope</i> blank or local-qmgr	<i>CommandScope</i> qmgr-name
MQCHLD_PRIVATE	Resolve private channel on the local queue manager	Resolve private channel on the named queue manager
MQCHLD_SHARED	This might automatically generate a command using <i>CommandScope</i> and send it to the appropriate queue manager. If there is no definition for the channel on the queue manager to which the command is sent, or if the definition is unsuitable for the command, the command fails. The definition of a channel on the queue manager where the command is entered might be used to determine the target queue manager where the command is actually run. Therefore, it is important that channel definitions are consistent. Inconsistent channel definitions might result in unexpected command behavior.	Not permitted

Error codes

This command might return the following in the response format header, in addition to the values shown on page “Error codes applicable to all commands” on page 27.

Reason (MQLONG)

The value can be:

MQRCCF_CHANNEL_NOT_FOUND

Channel not found.

MQRCCF_INDOUBT_VALUE_ERROR

In-doubt value not valid.

Resume Queue Manager

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The Resume Queue Manager (MQCMD_RESUME_Q_MGR) command renders the queue manager available again for the processing of IMS or DB2 messages.

It reverses the action of the Suspend Queue Manager (MQCMD_SUSPEND_Q_MGR) command.

Required parameters:

Facility

Optional parameters:

None

Required parameters

Facility (MQCFIN)

Facility (parameter identifier: MQIACF_FACILITY).

The type of facility for which activity is to be resumed. The value can be:

MQQMFACT_DB2

Resumes normal activity with DB2.

MQQMFACT_IMS_BRIDGE

Resumes normal IMS Bridge activity.

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you

specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.

The maximum length is MQ_QSG_NAME_LENGTH.

Resume Queue Manager Cluster

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Resume Queue Manager Cluster (MQCMD_RESUME_Q_MGR_CLUSTER) command informs other queue managers in a cluster that the local queue manager is again available for processing, and can be sent messages.

It reverses the action of the Suspend Queue Manager Cluster (MQCMD_SUSPEND_Q_MGR_CLUSTER) command.

Required parameters:

ClusterName, or *ClusterNamelist*

Optional parameters:

CommandScope

Required parameters

ClusterName (MQCFST)

Cluster name (parameter identifier: MQCA_CLUSTER_NAME).

The name of the cluster for which availability is to be resumed.

The maximum length of the string is MQ_CLUSTER_NAME_LENGTH.

ClusterNamelist (MQCFST)

Cluster Namelist (parameter identifier: MQCA_CLUSTER_NAMELIST).

The name of the namelist specifying a list of clusters for which availability is to be resumed.

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.

The maximum length is MQ_QSG_NAME_LENGTH.

Error codes

This command might return the following in the response format header, in addition to the values shown on page “Error codes applicable to all commands” on page 27.

Reason (MQLONG)

The value can be:

MQRCCF_CLUSTER_NAME_CONFLICT

Cluster name conflict.

Reverify Security

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The Reverify Security (MQCMD_REVERIFY_SECURITY) to set a reverification flag for all specified users. The user is reverified the next time that security is checked for that user.

Required parameters:

UserId

Optional parameters:

CommandScope

Required parameters

UserId (MQCFST)

User ID (parameter identifier: MQCACF_USER_IDENTIFIER).

Use this to specify one or more user IDs. Each user ID specified is signed off and signed back on again the next time that a request requiring a security check is issued on behalf of that user.

The maximum length of the string is MQ_USER_ID_LENGTH.

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

Set Archive

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

Use the Set Archive (MQCMD_SET_ARCHIVE) to dynamically change certain archive system parameter values initially set by your system parameter module at queue manager startup.

Required parameters:

ParameterType

Optional parameters if *ParameterType* type is MQSYSP_SET:

*AllocPrimary, AllocSecondary, AllocUnits, ArchivePrefix1,
ArchivePrefix2, ArchiveRetention, ArchiveUnit1, ArchiveUnit2,
ArchiveWTOR, BlockSize, Catalog, CommandScope, Compact, Protect,
QuiesceInterval, RoutingCode,TimeStampFormat*

Optional parameters if *ParameterType* type is MQSYSP_INITIAL:

CommandScope

Required parameters

ParameterType (MQCFIN)

Parameter type (parameter identifier: MQIACF_SYSP_TYPE).

Specifies how the parameters are to be reset:

MQSYSP_TYPE_INITIAL

The initial settings of the archive system parameters. This resets all the archive system parameters to the values set at queue manager startup.

MQSYSP_TYPE_SET

This indicates that you intend to change one, or more, of the archive system parameter settings.

Optional parameters

AllocPrimary (MQCFIN)

Primary space allocation for DASD data sets (parameter identifier: MQIACF_SYSP_ALLOC_PRIMARY).

Specifies the primary space allocation for DASD data sets in the units specified in the *AllocUnits* parameter.

Specify a value greater than zero. This value must be sufficient for a copy of either the log data set or its corresponding BSDS, whichever is the larger.

AllocSecondary (MQCFIN)

Secondary space allocation for DASD data sets (parameter identifier: MQIACF_SYSP_ALLOC_SECONDARY).

Specifies the secondary space allocation for DASD data sets in the units specified in the *AllocUnits* parameter.

Specify a value greater than zero.

AllocUnits (**MQCFIN**)

Allocation unit (parameter identifier: MQIACF_SYSP_ALLOC_UNIT).

Specifies the unit in which primary and secondary space allocations are made. The value can be:

MQSYSP_ALLOC_BLK
Blocks.

MQSYSP_ALLOC_TRK
Tracks.

MQSYSP_ALLOC_CYL
Cylinders.

ArchivePrefix1 (**MQCFST**)

Specifies the prefix for the first archive log data set name (parameter identifier: MQCACF_SYSP_ARCHIVE_PFX1).

The maximum length of the string is MQ_ARCHIVE_PFX_LENGTH.

ArchivePrefix2 (**MQCFST**)

Specifies the prefix for the second archive log data set name (parameter identifier: MQCACF_SYSP_ARCHIVE_PFX2).

The maximum length of the string is MQ_ARCHIVE_PFX_LENGTH.

ArchiveRetention (**MQCFIN**)

Archive retention period (parameter identifier: MQIACF_SYSP_ARCHIVE_RETAIN).

Specifies the retention period, in days, to be used when the archive log data set is created. Specify a value in the range zero through 9999.

See the *z/OS System Administration Guide* for information about discarding archive log data sets.

ArchiveUnit1 (**MQCFST**)

Specifies the device type or unit name of the device that is used to store the first copy of the archive log data set (parameter identifier: MQCACF_SYSP_ARCHIVE_UNIT1).

Specify a device type or unit name of 1 through 8 characters.

If you archive to DASD, you can specify a generic device type with a limited volume range.

The maximum length of the string is MQ_ARCHIVE_UNIT_LENGTH.

ArchiveUnit2 (**MQCFST**)

Specifies the device type or unit name of the device that is used to store the second copy of the archive log data set (parameter identifier: MQCACF_SYSP_ARCHIVE_UNIT2).

Specify a device type or unit name of 1 through 8 characters.

If this parameter is blank, the value set for the *ArchiveUnit1* parameter is used.

The maximum length of the string is MQ_ARCHIVE_UNIT_LENGTH.

ArchiveWTOR (**MQCFIN**)

Specifies whether a message is to be sent to the operator and a reply is received before attempting to mount an archive log data set (parameter identifier: MQIACF_SYSP_ARCHIVE_WTOR).

Other WebSphere MQ users might be forced to wait until the data set is mounted, but they are not affected while WebSphere MQ is waiting for the reply to the message.

The value can be:

MQSYSP_YES

A message is to be sent and a reply received before an attempt to mount an archive log data set.

MQSYSP_NO

A message is not to be sent and a reply received before an attempt to mount an archive log data set.

BlockSize (**MQCFIN**)

Block size of the archive log data set (parameter identifier: MQIACF_SYSP_BLOCK_SIZE).

The block size you specify must be compatible with the device type you specify in the *ArchiveUnit1* and *ArchiveUnit2* parameters.

Specify a value in the range 4 097 through 28 672. The value you specify is rounded up to a multiple of 4 096.

This parameter is ignored for data sets that are managed by the storage management system (SMS).

Catalog (**MQCFIN**)

Specifies whether archive log data sets are cataloged in the primary integrated catalog facility (parameter identifier: MQIACF_SYSP_CATALOG).

The value can be:

MQSYSP_YES

Archive log data sets are cataloged.

MQSYSP_NO

Archive log data sets are not cataloged.

CommandScope (**MQCFST**)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

Compact (**MQCFIN**)

Specifies whether data written to archive logs is to be compacted (parameter identifier: MQIACF_SYSP_COMPACT).

This parameter applies to a 3480 or 3490 device that has the improved data recording capability (IDRC) feature. When this feature is turned on, hardware in the tape control unit writes data at a much higher density than normal,

allowing for more data on each volume. Specify MQSYSP_NO if you do not use a 3480 device with the IDRC feature or a 3490 base model, with the exception of the 3490E. Specify MQSYSP_YES if you want the data to be compacted.

The value can be:

MQSYSP_YES

Data is to be compacted.

MQSYSP_NO

Data is not to be compacted.

Protect (MQCFIN)

Protection by external security manager (ESM) (parameter identifier: MQIACF_SYSP_PROTECT).

Specifies whether archive log data sets are protected by ESM profiles when the data sets are created.

If you specify MQSYSP_YES, ensure that:

- ESM protection is active for WebSphere MQ.
 - The user ID associated with the WebSphere MQ address space has authority to create these profiles.
 - The TAPEVOL class is active if you are archiving to tape.
- otherwise, off-loads will fail.

The value can be:

MQSYSP_YES

Data set profiles are created when logs are off-loaded.

MQSYSP_NO

Profiles are not created.

QuiesceInterval (MQCFIN)

Maximum time allowed for the quiesce (parameter identifier: MQIACF_SYSP QUIESCE_INTERVAL).

Specifies the maximum time, in seconds, allowed for the quiesce.

Specify a value in the range 1 through 999.

RoutingCode (MQCFIL)

z/OS routing code list (parameter identifier: MQIACF_SYSP_ROUTING_CODE).

Specifies the list of z/OS routing codes for messages about the archive log data sets to the operator.

Specify up to 14 routing codes, each with a value in the range zero through 16. You must specify at least one code.

TimeStampFormat (MQCFIN)

Time stamp included (parameter identifier: MQIACF_SYSP_TIMESTAMP).

Specifies whether the archive log data set name has a time stamp in it.

The value can be:

MQSYSP_YES

Names include a time stamp. The archive log data sets are named:
arcpxi.cyyddd.Thhmmssst.Annnnnnnn

where c is 'D' for the years up to and including 1999 or 'E' for the year 2000 and later, and $arcpxi$ is the data set name prefix specified by *ArchivePrefix1* or *ArchivePrefix2*. $arcpxi$ can have up to 19 characters.

MQSYS_P_NO

Names do not include a time stamp. The archive log data sets are named:

$arcpxi.Annnnnnn$

Where $arcpxi$ is the data set name prefix specified by *ArchivePrefix1* or *ArchivePrefix2*. $arcpxi$ can have up to 35 characters.

MQSYS_P_EXTENDED

Names include a time stamp. The archive log data sets are named:

$arcpxi.Dyyyyddd.Thhmmssst.Annnnnnn$

Where $arcpxi$ is the data set name prefix specified by *ArchivePrefix1* or *ArchivePrefix2*. $arcpxi$ can have up to 17 characters.

Set Authority Record

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
		X	X	X	

The Set Authority Record (MQCMD_SET_AUTH_REC) command sets the authorizations of a profile, object or class of objects. Authorizations can be granted to, or revoked from, any number of principals or groups.

Required parameters:

ObjectType, ProfileName

Optional parameters:

*AuthorityAdd, AuthorityRemove, GroupNames, PrincipalNames,
ServiceComponent*

Required parameters

***ObjectType* (MQCFIN)**

The type of object for which to set authorizations (parameter identifier: MQIACF_OBJECT_TYPE).

The value can be:

MQOT_AUTH_INFO

Authentication information.

MQOT_CHANNEL

Channel object.

MQOT_CLNTCONN_CHANNEL

Client-connection channel object.

MQOT_LISTENER

Listener object.

MQOT_NAMELIST

Namelist.

MQOT_PROCESS

Process.

MQOT_Q

Queue, or queues, that match the object name parameter.

MQOT_Q_MGR

Queue manager.

MQOT_SERVICE

Service object.

***ProfileName* (MQCFST)**

Profile name (parameter identifier: MQCACF_AUTH_PROFILE_NAME).

The authorizations apply to all WebSphere MQ objects with names that match the profile name specified. You may define a generic profile. If you specify an explicit profile name, the object must exist.

The maximum length of the string is MQ_AUTH_PROFILE_NAME_LENGTH.

Optional parameters

AuthorityAdd (MQCFIL)

Authority values to set (parameter identifier: MQIACF_AUTH_ADD_AUTHS).

This is a list of authority values to set for the named profile. The values can be:

MQAUTH_ALT_USER_AUTHORITY

Specify an alternate user ID on an MQI call.

MQAUTH_BROWSE

Retrieve a message from a queue by issuing an MQGET call with the BROWSE option.

MQAUTH_CHANGE

Change the attributes of the specified object, using the appropriate command set.

MQAUTH_CLEAR

Clear a queue.

MQAUTH_CONNECT

Connect the application to the specified queue manager by issuing an MQCONN call.

MQAUTH_CREATE

Create objects of the specified type using the appropriate command set.

MQAUTH_DELETE

Delete the specified object using the appropriate command set.

MQAUTH_DISPLAY

Display the attributes of the specified object using the appropriate command set.

MQAUTH_INPUT

Retrieve a message from a queue by issuing an MQGET call.

MQAUTH_INQUIRE

Make an inquiry on a specific queue by issuing an MQINQ call.

MQAUTH_NONE

Entity has an explicit access of zero to the selected profile.

MQAUTH_OUTPUT

Put a message on a specific queue by issuing an MQPUT call.

MQAUTH_PASS_ALL_CONTEXT

Pass all context.

MQAUTH_PASS_IDENTITY_CONTEXT

Pass the identity context.

MQAUTH_SET

Set attributes on a queue from the MQI by issuing an MQSET call.

MQAUTH_SET_ALL_CONTEXT

Set all context on a queue.

MQAUTH_SET_IDENTITY_CONTEXT

Set the identity context on a queue.

The contents of the *AuthorityAdd* and *AuthorityRemove* lists should be mutually exclusive. You must specify a value for either *AuthorityAdd* or *AuthorityRemove*. An error occurs if you do not specify either.

AuthorityRemove (MQCFIL)

Authority values to remove (parameter identifier:
MQIACF_AUTH_REMOVE_AUTHS).

This is a list of authority values to remove from the named profile. The values can be:

MQAUTH_ALT_USER_AUTHORITY

Specify an alternate user ID on an MQI call.

MQAUTH_BROWSE

Retrieve a message from a queue by issuing an MQGET call with the BROWSE option.

MQAUTH_CHANGE

Change the attributes of the specified object, using the appropriate command set.

MQAUTH_CLEAR

Clear a queue.

MQAUTH_CONNECT

Connect the application to the specified queue manager by issuing an MQCONN call.

MQAUTH_CREATE

Create objects of the specified type using the appropriate command set.

MQAUTH_DELETE

Delete the specified object using the appropriate command set.

MQAUTH_DISPLAY

Display the attributes of the specified object using the appropriate command set.

MQAUTH_INPUT

Retrieve a message from a queue by issuing an MQGET call.

MQAUTH_INQUIRE

Make an inquiry on a specific queue by issuing an MQINQ call.

MQAUTH_NONE

Entity has an explicit access of zero to the selected profile.

MQAUTH_OUTPUT

Put a message on a specific queue by issuing an MQPUT call.

MQAUTH_PASS_ALL_CONTEXT

Pass all context.

MQAUTH_PASS_IDENTITY_CONTEXT

Pass the identity context.

MQAUTH_SET

Set attributes on a queue from the MQI by issuing an MQSET call.

MQAUTH_SET_ALL_CONTEXT

Set all context on a queue.

MQAUTH_SET_IDENTITY_CONTEXT

Set the identity context on a queue.

The contents of the *AuthorityAdd* and *AuthorityRemove* lists should be mutually exclusive. You must specify a value for either *AuthorityAdd* or *AuthorityRemove*. An error occurs if you do not specify either.

GroupNames (MQCFSL)

Group names (parameter identifier: MQCACF_GROUP_ENTITY_NAMES).

The names of groups having their authorizations set. At least one group name or principal name must be specified. An error occurs if neither are specified.

Each member in this list can be a maximum length of
MQ_ENTITY_NAME_LENGTH.

PrincipalNames (MQCFSL)

Principal names (parameter identifier:
MQCACF_PRINCIPAL_ENTITY_NAMES).

The names of principals having their authorizations set. At least one group name or principal name must be specified. An error occurs if neither are specified.

Each member in this list can be a maximum length of
MQ_ENTITY_NAME_LENGTH.

ServiceComponent (MQCFST)

Service component (parameter identifier: MQCACF_SERVICE_COMPONENT).

If installable authorization services are supported, this specifies the name of the authorization service to which the authorizations apply.

If you omit this parameter, the authorization inquiry is made to the first installable component for the service.

The maximum length of the string is MQ_SERVICE_COMPONENT_LENGTH.

Error codes

This command might return the following in the response format header, in addition to the values shown on page “Error codes applicable to all commands” on page 27.

Reason (MQLONG)

The value can be:

MQRC_UNKNOWN_ENTITY	Userid not authorized, or unknown.
MQRCCF_AUTH_VALUE_ERROR	Invalid authorization.
MQRCCF_AUTH_VALUE_MISSING	Authorization missing.
MQRCCF_ENTITY_NAME_MISSING	Entity name missing.
MQRCCF_OBJECT_TYPE_MISSING	Object type missing.
MQRCCF_PROFILE_NAME_ERROR	Invalid profile name.

Set Log

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

Use the Set Log (MQCMD_SET_LOG) command to dynamically change certain log system parameter values initially set by your system parameter module at queue manager startup.

Required parameters:

ParameterType

Optional parameters (if the value of *ParameterType* is MQSYSP_TYPE_SET):

*CommandScope, DeallocateInterval, MaxArchiveLog, MaxReadTapeUnits,
OutputBufferCount*

Optional parameters if *ParameterType* type is MQSYSP_INITIAL:

CommandScope

Required parameters

ParameterType (MQCFIN)

Parameter type (parameter identifier: MQIACF_SYSP_TYPE).

Specifies how the parameters are to be set:

MQSYSP_TYPE_INITIAL

The initial settings of the log system parameters. This resets all the log system parameters to the values at queue manager startup.

MQSYSP_TYPE_SET

This indicates that you intend to change one, or more, of the archive log system parameter settings.

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

DeallocateInterval (MQCFIN)

Deallocation interval (parameter identifier:
MQIACF_SYSP DEALLOC_INTERVAL).

Specifies the length of time, in minutes, that an allocated archive read tape unit is allowed to remain unused before it is deallocated. This parameter, together with the *MaxReadTapeUnits* parameter, allows WebSphere MQ to optimize archive log reading from tape devices. You are recommended to specify the maximum possible values, within system constraints, for both parameters, in order to achieve the optimum performance for reading archive tapes.

Specify a value in the range zero and 1440. Zero means that a tape unit is deallocated immediately. If you specify a value of 1440, the tape unit is never deallocated.

MaxArchiveLog (MQCFIN)

Specifies the maximum number of archive log volumes that can be recorded in the BSDS (parameter identifier: MQIACF_SYSP_MAX_ARCHIVE).

When this value is exceeded, recording recommences at the start of the BSDS.

Specify a value in the range 10 through 100.

MaxReadTapeUnits (MQCFIN)

Specifies the maximum number of dedicated tape units that can be allocated to read archive log tape volumes (parameter identifier:
MQIACF_SYSP_MAX_READ_TAPES).

This parameter, together with the *DeallocateInterval* parameter, allows WebSphere MQ to optimize archive log reading from tape devices.

Specify a value in the range 1 through 99.

If you specify a value that is greater than the current specification, the maximum number of tape units allowable for reading archive logs increases. If you specify a value that is less than the current specification, tape units that are not being used are immediately deallocated to adjust to the new value. Active, or premounted, tapes remain allocated.

OutputBufferCount (MQCFIN)

Specifies the number of 4 KB output buffers to be filled before they are written to the active log data sets (parameter identifier:
MQIACF_SYSP_OUT_BUFFER_COUNT).

Specify the number of buffers in the range 1 through 256.

The larger the number of buffers, the less often the write takes place, and this improves the performance of WebSphere MQ. The buffers might be written before this number is reached if significant events, such as a commit point, occur.

Set System

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

Use the Set System (MQCMD_SET_SYSTEM) command to dynamically change certain general system parameter values initially set from your system parameter module at queue manager startup.

Required parameters:

ParameterType

Optional parameters (if the value of *ParameterType* is MQSYSP_TYPE_SET:

*CheckpointCount, CommandScope, MaxConnects, MaxConnectsBackground,
MaxConnectsForeground, Service, SMFInterval, TraceSize*

Optional parameters if *ParameterType* type is MQSYSP_INITIAL:

CommandScope

Required parameters

***ParameterType* (MQCFIN)**

Parameter type (parameter identifier: MQIACF_SYSP_TYPE).

Specifies how the parameters are to be set:

MQSYSP_TYPE_INITIAL

The initial settings of the system parameters. This resets the parameters to the values specified in the system parameters at queue manager startup.

MQSYSP_TYPE_SET

This indicates that you intend to change one, or more, of the log parameter settings.

Optional parameters

***CheckpointCount* (MQCFIN)**

The number of log records written by WebSphere MQ between the start of one checkpoint and the next (parameter identifier: MQIACF_SYSP_CHKPOINT_COUNT).

WebSphere MQ starts a new checkpoint after the number of records that you specify has been written.

Specify a value in the range 200 through 16 000 000.

***CommandScope* (MQCFST)**

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you

specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.

- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

MaxConnects (MQCFIN)

The maximum number of connections from batch, CICS, IMS, and TSO tasks to a single instance of WebSphere MQ (parameter identifier: MQIACF_SYSP_MAX_CONNS).

Specify a value in the range 1 through 32 767.

MaxConnectsBackground (MQCFIN)

The maximum number of connections from batch or TSO background tasks to a single instance of WebSphere MQ (parameter identifier: MQIACF_SYSP_MAX_CONNS_BACK).

Specify a value in the range zero through 32 767.

MaxConnectsForeground (MQCFIN)

The maximum number of connections from TSO foreground tasks to a single instance of WebSphere MQ (parameter identifier: MQIACF_SYSP_MAX_CONNS_FORE).

Specify a value in the range zero through 32 767.

Service (MQCFST)

Service parameter setting (parameter identifier: MQIACF_SYSP_SERVICE).

This parameter is reserved for use by IBM.

SMFInterval (MQCFIN)

The default time, in minutes, between each gathering of statistics (parameter identifier: MQIACF_SYSP_SMF_INTERVAL).

Specify a value in the range zero through 1440.

If you specify a value of zero, statistics data and accounting data are both collected at the SMF data collection broadcast.

TraceSize (MQCFIN)

The size of the trace table, in 4 KB blocks, to be used by the global trace facility (parameter identifier: MQIACF_SYSP_TRACE_SIZE).

Specify a value in the range zero through 999.

Start Channel

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Start Channel (MQCMD_START_CHANNEL) command starts a WebSphere MQ channel.

This command can be issued to a channel of any type (except MQCHT_CLNTCONN). If, however, it is issued to a channel with a *Channel Type*

value of MQCHT_RECEIVER, MQCHT_SVRCONN, or MQCHT_CLUSRCVR, the only action is to enable the channel, not start it.

Where there is both a locally defined channel and an auto-defined cluster-sender channel of the same name, the command applies to the locally defined channel.

If there is no locally defined channel but more than one auto-defined cluster-sender channel, the command applies to the last channel added to the repository on the local queue manager.

Required parameters:

ChannelName

Optional parameters:

CommandScope, *ChannelDisposition*

Required parameters

ChannelName (**MQCFST**)

Channel name (parameter identifier: MQCACH_CHANNEL_NAME).

The name of the channel to be started. The maximum length of the string is MQ_CHANNEL_NAME_LENGTH.

Optional parameters

CommandScope (**MQCFST**)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

ChannelDisposition (**MQCFIN**)

Channel disposition (parameter identifier: MQIACH_CHANNEL_DISP). This parameter applies to z/OS only.

Specifies the disposition of the channels to be started.

The value can be:

MQCHLD_PRIVATE

A receiving channel is private if it was started in response to an inbound transmission directed to the queue manager.

A sending channel is private if its transmission queue has a disposition other than MQQSGD_SHARED.

MQCHLD_SHARED

A receiving channel is shared if it was started in response to an inbound transmission directed to the queue-sharing group.

A sending channel is shared if its transmission queue has a disposition of MQQSGD_SHARED.

MQCHLD_FIXSHARED

Shared channels tied to a specific queue manager.

The combination of the *ChannelDisposition* and *CommandScope* parameters also controls from which queue manager the channel is operated. The possible options are:

- On the local queue manager where the command is issued.
- On another specific named queue manager in the group.
- On every active queue manager in the group.
- On the most suitable queue manager in the group, determined automatically by the queue manager itself.

The various combinations of *ChannelDisposition* and *CommandScope* are summarized in Table 13

Table 13. *ChannelDisposition* and *CommandScope* for START CHANNEL

<i>ChannelDisposition</i>	<i>CommandScope</i> blank or local-qmgr	<i>CommandScope</i> qmgr-name	<i>CommandScope</i> (*)
MQCHLD_PRIVATE	Start as a private channel on the local queue manager	Start as a private channel on the named queue manager	Start as a private channel on all active queue managers

Table 13. ChannelDisposition and CommandScope for START CHANNEL (continued)

ChannelDisposition	CommandScope blank or local-qmgr	CommandScope qmgr-name	CommandScope (*)
MQCHLD_SHARED	<p>For channels of <i>ChannelType</i> MQCHT_SENDER, MQCHT_REQUESTER, and MQCHT_SERVER, start as a shared channel on the most suitable queue manager in the group.</p> <p>For a shared channel of <i>ChannelType</i> MQCHT_RECEIVER and MQCHT_SVRCONN, start the channel on all active queue managers.</p> <p>For a shared channel of <i>ChannelType</i> MQCHT_CLUSSDR and MQCHT_CLUSRCVR, this option is not permitted.</p> <p>This might automatically generate a command using <i>CommandScope</i> and send it to the appropriate queue manager. If there is no definition for the channel on the queue manager to which the command is sent, or if the definition is unsuitable for the command, the command fails.</p> <p>The definition of a channel on the queue manager where the command is entered might be used to determine the target queue manager where the command is actually run. Therefore, it is important that channel definitions are consistent. Inconsistent channel definitions might result in unexpected command behavior.</p>	Not permitted	Not permitted
MQCHLD_FIXSHARED	For a shared channel of <i>ChannelType</i> MQCHT_SENDER, MQCHT_REQUESTER, and MQCHT_SERVER, with a nonblank <i>ConnectionName</i> , start as a shared channel on the local queue manager.	For a shared channel of <i>ChannelType</i> MQCHT_SENDER, MQCHT_REQUESTER, and MQCHT_SERVER, with a nonblank <i>ConnectionName</i> , start as a shared channel on the named queue manager.	Not permitted

Error codes

This command might return the following in the response format header, in addition to the values shown on page “Error codes applicable to all commands” on page 27.

Reason (MQLONG)

The value can be:

MQRCCF_CHANNEL_INDOUBT	Channel in-doubt.
MQRCCF_CHANNEL_IN_USE	Channel in use.
MQRCCF_CHANNEL_NOT_FOUND	Channel not found.
MQRCCF_CHANNEL_TYPE_ERROR	Channel type not valid.
MQRCCF_MQCONN_FAILED	MQCONN call failed.
MQRCCF_MQINQ_FAILED	MQINQ call failed.
MQRCCF_MQOPEN_FAILED	MQOPEN call failed.
MQRCCF_NOT_XMIT_Q	Queue is not a transmission queue.

Start Channel Initiator

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
	X	X	X	X	X

The Start Channel Initiator (MQCMD_START_CHANNEL_INIT) command starts a WebSphere MQ channel initiator.

Required parameters:

None on z/OS, *InitiationQName* on other platforms.

Optional parameters:

CommandScope, *EnvironmentInfo*

Required parameters

InitiationQName (MQCFST)

Initiation queue name (parameter identifier: MQCA_INITIATION_Q_NAME).

The name of the initiation queue for the channel initiation process. That is, the initiation queue that is specified in the definition of the transmission queue.

This parameter is not valid on z/OS.

The maximum length of the string is MQ_Q_NAME_LENGTH.

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.

The maximum length is MQ_QSG_NAME_LENGTH.

EnvironmentInfo (MQCFST)

Environment information (parameter identifier: MQCACF_ENV_INFO).

The parameters and values to be substituted in the JCL procedure (xxxxCHIN, where xxxx is the queue manager name) that is used to start the channel initiator address space. This parameter applies to z/OS only.

The maximum length of the string is MQ_ENV_INFO_LENGTH.

Error codes

This command might return the following in the response format header, in addition to the values shown on page “Error codes applicable to all commands” on page 27.

Reason (MQLONG)

The value can be:

MQRCCF_MQCONN_FAILED

MQCONN call failed.

MQRCCF_MQGET_FAILED

MQGET call failed.

MQRCCF_MQOPEN_FAILED

MQOPEN call failed.

Start Channel Listener

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
		X	X	X	X

The Start Channel Listener (MQCMD_START_CHANNEL_LISTENER) command starts a WebSphere MQ listener.

On z/OS, this command is valid for any transmission protocol; on other platforms, it is valid only for TCP transmission protocols.

Required parameters:

None

Optional parameters:

*CommandScope, InboundDisposition, IPAddress, ListenerName, LUName, Port,
TransportType*

Optional parameters

CommandScope (**MQCFST**)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.

The maximum length is MQ_Q_MGR_NAME_LENGTH.

InboundDisposition (**MQCFIN**)

Inbound transmission disposition (parameter identifier: MQIACH_INBOUND_DISP). This parameter applies to z/OS only.

Specifies the disposition of the inbound transmissions that are to be handled. The value can be:

MQINBD_Q_MGR

Listen for transmissions directed to the queue manager. This is the default.

MQINBD_GROUP

Listen for transmissions directed to the queue-sharing group. This is permitted only if there is a shared queue manager environment.

IPAddress (**MQCFST**)

IP address (parameter identifier: MQCACH_IP_ADDRESS). This parameter applies to z/OS only.

The IP address for TCP/IP specified in IPv4 dotted decimal, IPv6 hexadecimal, or alphanumeric form. This parameter is valid only for channels that have a *TransportType* of MQXPT_TCP.

The maximum length of the string is MQ_IP_ADDRESS_LENGTH.

ListenerName (**MQCFST**)

Listener name (parameter identifier: MQCACH_LISTENER_NAME). This parameter does not apply to z/OS.

The name of the listener definition to be started. On those platforms on which this parameter is valid, if this parameter is not specified, the default listener SYSTEM.DEFAULT.LISTENER is assumed. If this parameter is specified, no other parameters may be specified.

The maximum length of the string is MQ_LISTENER_NAME_LENGTH.

LUName (**MQCFST**)

LU name (parameter identifier: MQCACH_LU_NAME). This parameter applies to z/OS only.

The symbolic destination name for the logical unit (LU) as specified in the APPC side information data set. The LU must be the same LU that is specified in the channel initiator parameters to be used for outbound transmissions. This parameter is valid only for channels with a *TransportType* of MQXPT_LU62.

The maximum length of the string is MQ_LU_NAME_LENGTH.

Port (**MQCFIN**)

Port number for TCP (parameter identifier: MQIACH_PORT_NUMBER). This parameter applies to z/OS only.

The port number for TCP. This parameter is valid only for channels with a *TransportType* of MQXPT_TCP.

TransportType (**MQCFIN**)

Transmission protocol type (parameter identifier: MQIACH_XMIT_PROTOCOL_TYPE).

The value can be:

MQXPT_LU62

LU 6.2.

MQXPT_TCP

TCP.

MQXPT_NETBIOS

NetBIOS.

MQXPT_SPX

SPX.

On platforms other than z/OS, this parameter is invalid.

Error codes

This command might return the following in the response format header, in addition to the values shown on page "Error codes applicable to all commands" on page 27.

Reason (**MQLONG**)

The value can be:

MQRCCF_COMMs_LIBRARY_ERROR

Communications protocol library error.

MQRCCF_LISTENER_NOT_STARTED

Listener not started.

MQRCCF_LISTENER_RUNNING

Listener already running.

MQRCCF_NETBIOS_NAME_ERROR

NetBIOS listener name error.

Start Service

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
		X	X	X	

The Start Service (MQCMD_START_SERVICE) command starts an existing WebSphere MQ service definition.

Required parameters:

ServiceName

Optional parameters:

None

Required parameters

***ServiceName* (MQCFST)**

Service name (parameter identifier: MQCA_SERVICE_NAME).

This is the name of the service definition to be started. The maximum length of the string is MQ_OBJECT_NAME_LENGTH.

Error codes

This command might return the following in the response format header, in addition to the values shown on page “Error codes applicable to all commands” on page 27.

***Reason* (MQLONG)**

The value can be:

MQRCCF_NO_START_CMD

The *StartCommand* parameter of the service is blank.

MQRCCF_SERVICE_RUNNING

Service is already running.

Stop Channel

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Stop Channel (MQCMD_STOP_CHANNEL) command stops a WebSphere MQ channel.

This command can be issued to a channel of any type (except MQCHT_CLNTCONN).

Where there is both a locally defined channel and an auto-defined cluster-sender channel of the same name, the command applies to the locally defined channel.

If there is no locally defined channel but more than one auto-defined cluster-sender channel, the command applies to the last channel added to the repository on the local queue manager.

Required parameters:

ChannelName

Optional parameters:

ChannelDisposition, ChannelStatus, CommandScope, ConnectionName, Mode, QMgrName,

Required parameters

ChannelName (MQCFST)

Channel name (parameter identifier: MQCACH_CHANNEL_NAME).

The name of the channel to be stopped. The maximum length of the string is MQ_CHANNEL_NAME_LENGTH.

Optional parameters

ChannelDisposition (MQCFIN)

Channel disposition (parameter identifier: MQIACH_CHANNEL_DISP). This parameter applies to z/OS only.

Specifies the disposition of the channels to be stopped.

The value can be:

MQCHLD_PRIVATE

A receiving channel is private if it was started in response to an inbound transmission directed to the queue manager.

A sending channel is private if its transmission queue has a disposition other than MQQSGD_SHARED.

MQCHLD_SHARED

A receiving channel is shared if it was started in response to an inbound transmission directed to the queue-sharing group.

A sending channel is shared if its transmission queue has a disposition of MQQSGD_SHARED.

The combination of the *ChannelDisposition* and *CommandScope* parameters also controls from which queue manager the channel is operated. The possible options are:

- On the local queue manager where the command is issued.
- On another specific named queue manager in the group.
- On every active queue manager in the group.
- On the most suitable queue manager in the group, determined automatically by the queue manager itself.

The various combinations of *ChannelDisposition* and *CommandScope* are summarized in Table 14

Table 14. *ChannelDisposition* and *CommandScope* for STOP CHANNEL

<i>ChannelDisposition</i>	<i>CommandScope</i> blank or local-qmgr	<i>CommandScope</i> qmgr-name	<i>CommandScope</i> (*)
MQCHLD_PRIVATE	Stop as a private channel on the local queue manager	Stop as a private channel on the named queue manager	Stop as a private channel on all active queue managers

Table 14. ChannelDisposition and CommandScope for STOP CHANNEL (continued)

<i>ChannelDisposition</i>	<i>CommandScope</i> blank or local-qmgr	<i>CommandScope</i> qmgr-name	<i>CommandScope</i> (*)
MQCHLD_SHARED	<p>For channels of <i>ChannelType</i> MQCHT_RECEIVER or MQCHT_SVRCONN, stop as shared channel on all active queue managers.</p> <p>For channels of <i>ChannelType</i> MQCHT_SENDER, MQCHT_REQUESTER, and MQCHT_SERVER, stop as a shared channel on the queue manager where it is running. If the channel is in an inactive state (not running), or if it is in RETRY state because the channel initiator on which it was running has stopped, a STOP request for the channel is issued on the local queue manager.</p> <p>This might automatically generate a command using <i>CommandScope</i> and send it to the appropriate queue manager. If there is no definition for the channel on the queue manager to which the command is sent, or if the definition is unsuitable for the command, the command fails.</p> <p>The definition of a channel on the queue manager where the command is entered might be used to determine the target queue manager where the command is actually run. Therefore, it is important that channel definitions are consistent. Inconsistent channel definitions might result in unexpected command behavior.</p>	Not permitted	Not permitted

ChannelStatus (MQCFIN)

The new state of the channel after the command is executed (parameter identifier: MQIACH_CHANNEL_STATUS).

The value can be:

MQCHS_INACTIVE

Channel is inactive.

MQCHS_STOPPED

Channel is stopped. This is the default if nothing is specified.

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

ConnectionName (MQCFST)

Connection name of channel to be stopped (parameter identifier: MQCACH_CONNECTION_NAME).

This is the connection name of the channel to be stopped. If this parameter is omitted, all channels with the specified channel name and remote queue manager name are stopped. On platforms other than z/OS, the maximum length of the string is MQ_CONN_NAME_LENGTH. On z/OS, the maximum length of the string is MQ_LOCAL_ADDRESS_LENGTH.

If this parameter is specified, ChannelStatus must be MQCHS_INACTIVE.

Mode (MQCFIN)

How the channel should be stopped (parameter identifier: MQIACF_MODE).

The value can be:

MQMODE QUIESCE

Quiesce the channel. This is the default.

MQMODE_FORCE

Stop the channel immediately; the channel's thread or process is not terminated.

MQMODE TERMINATE

Stop the channel immediately; the channel's thread or process is terminated.

Note: This parameter was previously called *Quiesce* (MQIACF QUIESCE), with values MQQQ_YES and MQQQ_NO. The old names can still be used.

QMgrName (MQCFST)

Name of remote queue manager (parameter identifier: MQCA_Q_MGR_NAME).

This is the name of the remote queue manager to which the channel is connected. If this parameter is omitted, all channels with the specified channel name and connection name are stopped. The maximum length of the string is MQ_Q_MGR_NAME_LENGTH.

If this parameter is specified, ChannelStatus must be MQCHS_INACTIVE.

Error codes

This command might return the following in the response format header, in addition to the values shown on page "Error codes applicable to all commands" on page 27.

Reason (MQLONG)

The value can be:

MQRCCF_CHANNEL_DISABLED

Channel disabled.

MQRCCF_CHANNEL_NOT_ACTIVE

Channel not active.

MQRCCF_CHANNEL_NOT_FOUND

Channel not found.

MQRCCF_MODE_VALUE_ERROR

Mode value not valid.

MQRCCF_MQCONN_FAILED

MQCONN call failed.

MQRCCF_MQOPEN_FAILED

MQOPEN call failed.

MQRCCF_MQSET_FAILED

MQSET call failed.

Stop Channel Initiator

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The Stop Channel Initiator (MQCMD_STOP_CHANNEL_INIT) command stops a WebSphere MQ channel initiator.

Required parameters:

None

Optional parameters:

CommandScope, SharedChannelRestart

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.
- an asterisk (*). The command is executed on the local queue manager and is also passed to every active queue manager in the queue-sharing group.

The maximum length is MQ_QSG_NAME_LENGTH.

SharedChannelRestart (MQCFIN)

Shared channel restart (parameter identifier: MQIACH_SHARED_CHANNEL_RESTART).

Specifies whether the channel initiator should attempt to restart any active sending channels , started with the *ChannelDisposition* parameter set to MQCHLD_SHARED, that it owns on another queue manager. The value can be:

MQCHSH_RESTART_YES

Shared sending channels are to be restarted. This is the default.

MQCHSH_RESTART_NO

Shared sending channels are not to be restarted, so will become inactive.

Active channels started with the *ChannelDisposition* parameter set to MQCHLD_FIXSHARED are not restarted, and always become inactive.

Stop Channel Listener

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
		X	X	X	X

The Stop Channel Listener (MQCMD_STOP_CHANNEL_LISTENER) command stops a WebSphere MQ listener.

Required parameters:

None on z/OS, *ListenerName* on other platforms

Optional parameters:

CommandScope, *InboundDisposition*, *IPAddress*, *Port*, *TransportType*

Required parameters

ListenerName (MQCFST)

Listener name (parameter identifier: MQCACH_LISTENER_NAME). This parameter does not apply to z/OS.

The name of the listener definition to be stopped. If this parameter is specified, no other parameters may be specified.

The maximum length of the string is MQ_LISTENER_NAME_LENGTH.

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you

specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.

This is valid only on z/OS.

The maximum length is MQ_QSG_NAME_LENGTH.

InboundDisposition (**MQCFIN**)

Inbound transmission disposition (parameter identifier: MQIACH_INBOUND_DISP).

Specifies the disposition of the inbound transmissions that the listener handles. The value can be:

MQINBD_Q_MGR

Handling for transmissions directed to the queue manager. This is the default.

MQINBD_GROUP

Handling for transmissions directed to the queue-sharing group. This is permitted only if there is a shared queue manager environment.

This is valid only on z/OS.

IPAddress (**MQCFST**)

IP address (parameter identifier: MQCACH_IP_ADDRESS).

The IP address for TCP/IP specified in dotted decimal or alphanumeric form. This parameter is valid on z/OS only where channels have a *TransportType* of MQXPT_TCP.

The maximum length of the string is MQ_IP_ADDRESS_LENGTH.

This is valid only on z/OS.

Port (**MQCFIN**)

Port number for TCP (parameter identifier: MQIACH_PORT_NUMBER).

The port number for TCP. This parameter is valid only on z/OS where channels have a *TransportType* of MQXPT_TCP.

TransportType (**MQCFIN**)

Transmission protocol type (parameter identifier: MQIACH_XMIT_PROTOCOL_TYPE).

The value can be:

MQXPT_LU62
LU 6.2.

MQXPT_TCP
TCP.

This is valid only on z/OS.

Error codes

This command might return the following in the response format header, in addition to the values shown on page “Error codes applicable to all commands” on page 27.

Reason (MQLONG)

The value can be:

MQRCCF_LISTENER_STOPPED

Listener not running.

Stop Connection

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
		X	X	X	

The Stop Connection (MQCMD_STOP_CONNECTION) command attempts to break a connection between an application and the queue manager. There may be circumstances in which the queue manager cannot implement this command.

Required parameters:

ConnectionId

Optional parameters:

None

Required parameters

ConnectionId (MQCFBS)

Connection identifier (parameter identifier: MQBACF_CONNECTION_ID).

This is the unique connection identifier associated with an application that is connected to the queue manager.

The length of the byte string is MQ_CONNECTION_ID_LENGTH.

Stop Service

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
		X	X	X	

The Stop Service (MQCMD_STOP_SERVICE) command stops an existing WebSphere MQ service definition that is running..

Required parameters:

ServiceName

Optional parameters:

None

Required parameters

ServiceName (MQCFST)

Service name (parameter identifier: MQCA_SERVICE_NAME).

This is the name of the service definition to be stopped. The maximum length of the string is MQ_OBJECT_NAME_LENGTH.

Error codes

This command might return the following in the response format header, in addition to the values shown on page “Error codes applicable to all commands” on page 27.

Reason (MQLONG)

The value can be:

MQRCCF_NO_STOP_CMD

The *StopCommand* parameter of the service is blank.

MQRCCF_SERVICE_STOPPED

Service is not running.

Suspend Queue Manager

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
					X

The Suspend Queue Manager (MQCMD_SUSPEND_Q_MGR) command renders the local queue manager unavailable for the processing of IMS or DB2 messages.

Its action can be reversed by the Resume Queue Manager command (MQCMD_RESUME_Q_MGR) command.

Required parameters:

Facility

Optional parameters:

CommandScope

Required parameters

Facility (MQCFIN)

Facility (parameter identifier: MQIACF_FACILITY).

The type of facility for which activity is to be suspended. The value can be:

MQQMFACT_DB2

The existing connection to DB2 is terminated.

Any in-flight or subsequent MQGET or MQPUT requests are suspended and applications wait until the DB2 connection is re-established by the Resume Queue Manager command, or if the queue manager is stopped.

MQQMFACT_IMS_BRIDGE

Resumes normal IMS Bridge activity.

Stops the sending of messages from IMS Bridge queues to OTMA. No further messages are sent to IMS until one of these events occurs:

- OTMA is stopped and restarted
- IMS or WebSphere MQ is stopped or restarted
- A Resume Queue Manager command is processed

Messages returning from IMS OTMA to the queue manager are unaffected.

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE).

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.

The maximum length is MQ_QSG_NAME_LENGTH.

Suspend Queue Manager Cluster

Compaq NonStop Kernel	HP OpenVMS	i5/OS	UNIX systems	Windows	z/OS
X	X	X	X	X	X

The Suspend Queue Manager Cluster (MQCMD_SUSPEND_Q_MGR_CLUSTER) command informs other queue managers in a cluster that the local queue manager is not available for processing, and cannot be sent messages.

Its action can be reversed by the Resume Queue Manager Cluster (MQCMD_RESUME_Q_MGR_CLUSTER) command.

Required parameters:

ClusterName or *ClusterNamelist*

Optional parameters:

CommandScope, *Mode*

Required parameters

ClusterName (MQCFST)

Cluster name (parameter identifier: MQCA_CLUSTER_NAME).

The name of the cluster for which availability is to be suspended.

The maximum length of the string is MQ_CLUSTER_NAME_LENGTH.

ClusterNamelist (MQCFST)

Cluster Namelist (parameter identifier: MQCA_CLUSTER_NAMELIST).

The name of the namelist specifying a list of clusters for which availability is to be suspended.

Optional parameters

CommandScope (MQCFST)

Command scope (parameter identifier: MQCACF_COMMAND_SCOPE). This parameter applies to z/OS only.

Specifies how the command is executed when the queue manager is a member of a queue-sharing group. You can specify one of the following:

- blank (or omit the parameter altogether). The command is executed on the queue manager on which it was entered.
- a queue manager name. The command is executed on the queue manager you specify, providing it is active within the queue sharing group. If you specify a queue manager name other than the queue manager on which it was entered, you must be using a queue-sharing group environment, and the command server must be enabled.

The maximum length is MQ_QSG_NAME_LENGTH.

Mode (MQCFIN)

How the local queue manager should be suspended from the cluster (parameter identifier: MQIACF_MODE).

The value can be:

MQMODE QUIESCE

Other queue managers in the cluster are advised that the local queue manager should not be sent further messages.

MQMODE FORCE

All inbound and outbound channels to other queue managers in the cluster are stopped forcibly.

Note: This parameter was previously called *Quiesce* (MQIACF QUIESCE), with values MQQO_YES and MQQO_NO. The old names can still be used.

Error codes

This command might return the following in the response format header, in addition to the values shown on page “Error codes applicable to all commands” on page 27.

Reason (MQLONG)

The value can be:

MQRCCF_CLUSTER_NAME_CONFLICT

Cluster name conflict.

MQRCCF_MODE_VALUE_ERROR

Mode value not valid.

Chapter 4. Structures for commands and responses

Commands and responses have the form:

- PCF header (MQCFH) structure (described in topic “MQCFH - PCF header” on page 418), followed by
- Zero or more parameter structures. Each of these is one of the following:
 - PCF byte string filter parameter (MQCFBF, see topic “MQCFBF - PCF byte string filter parameter” on page 422)
 - PCF byte string parameter (MQCFBS, see topic “MQCFBS - PCF byte string parameter” on page 425)
 - PCF integer filter parameter (MQCFIF, see topic “MQCFIF - PCF integer filter parameter” on page 427)
 - PCF integer list parameter (MQCFIL, see topic “MQCFIL - PCF integer list parameter” on page 430)
 - PCF integer parameter (MQCFIN, see topic “MQCFIN - PCF integer parameter” on page 433)
 - PCF string filter parameter (MQCFSF, see topic “MQCFSF - PCF string filter parameter” on page 434)
 - PCF string list parameter (MQCFSL, see topic “MQCFSL - PCF string list parameter” on page 439)
 - PCF string parameter (MQCFST, see topic “MQCFST - PCF string parameter” on page 442)

How the structures are shown

The structures are described in a language-independent form. The declarations are shown in the following programming languages:

- C
- COBOL
- PL/I
- S/390® assembler
- Visual Basic

Data types

For each field of the structure the data type is given in brackets after the field name. These are the elementary data types described in the *WebSphere MQ Application Programming Reference* manual.

Initial values and default structures

See the *WebSphere MQ Constants* book for details of the supplied header files that contain the structures, constants, initial values and default structures.

Usage notes

If all of the strings in a PCF message have the same coded character-set identifier, the *CodedCharSetId* field in the message descriptor MQMD should be set to that identifier when the message is put, and the *CodedCharSetId* fields in the MQCFST, MQCFSL, and MQCFSF structures within the message should be set to MQCCSI_DEFAULT.

If the format of the PCF message is MQFMT_ADMIN, MQFMT_EVENT, or MQFMT_PCF and some of the strings in the message have different character-set identifiers, the *CodedCharSetId* field in MQMD should be set to MQCCSI_EMBEDDED when the message is put, and the *CodedCharSetId* fields in the MQCFST, MQCFSL, and MQCFSF structures within the message should all be set to the identifiers that apply.

This enables conversions of the strings within the message, to the *CodedCharSetId* value in the MQMD specified on the MQGET call, if the MQGMO_CONVERT option is also specified.

For more information about the MQEPH structure, see the *WebSphere MQ Application Programming Reference*.

Note: If you request conversion of the internal strings in the message, the conversion will occur only if the value of the *CodedCharSetId* field in the MQMD of the message is different from the *CodedCharSetId* field of the MQMD specified on the MQGET call.

Do not specify MQCCSI_EMBEDDED in MQMD when the message is put, with MQCCSI_DEFAULT in the MQCFST, MQCFSL, or MQCFSF structures within the message, as this will prevent conversion of the message.

MQCFH - PCF header

The MQCFH structure describes the information that is present at the start of the message data of a command message, or a response to a command message. In either case, the message descriptor *Format* field is MQFMT_ADMIN.

The PCF structures are also used for event messages. In this case the message descriptor *Format* field is MQFMT_EVENT.

The PCF structures can also be used for user-defined message data. In this case the message descriptor *Format* field is MQFMT_PCF (see “Message descriptor for a PCF command” on page 11). Also in this case, not all of the fields in the structure are meaningful. The supplied initial values can be used for most fields, but the application must set the *StrucLength* and *ParameterCount* fields to the values appropriate to the data.

Fields

Type (MQLONG)

Structure type.

This indicates the content of the message. The following are valid for commands:

MQCFT_COMMAND

Message is a command.

MQCFT_COMMAND_XR

Message is a command to which standard or extended responses may be sent.

MQCFT_RESPONSE

Message is a response to a command.

MQCFT_XR_MSG

Message is an extended response to a command. It contains informational or error details.

MQCFT_XR_ITEM

Message is an extended response to an Inquire command. It contains item data.

MQCFT_XR_SUMMARY

Message is an extended response to a command. It contains summary information.

MQCFT_USER

User-defined PCF message.

***StrucLength* (MQLONG)**

Structure length.

This is the length in bytes of the MQCFH structure. The value must be:

MQCFH_STRUC_LENGTH

Length of command format header structure.

***Version* (MQLONG)**

Structure version number.

The value must be:

MQCFH_VERSION_3

Version number for command format header structure.

The following constant specifies the version number of the current version:

MQCFH_CURRENT_VERSION

Current version of command format header structure.

***Command* (MQLONG)**

Command identifier.

For a command message, this identifies the function to be performed. For a response message, it identifies the command to which this is the reply. See the description of each command for the value of this field.

***MsgSeqNumber* (MQLONG)**

Message sequence number.

This is the sequence number of the message within a set of related messages. For a command, this field must have the value one (because a command is always contained within a single message). For a response, the field has the value one for the first (or only) response to a command, and increases by one for each successive response to that command.

The last (or only) message in a set has the MQCFC_LAST flag set in the *Control* field. T

Control (**MQLONG**)

Control options.

The following are valid:

MQCFC_LAST

Last message in the set.

For a command, this value must always be set.

MQCFC_NOT_LAST

Not the last message in the set.

CompCode (**MQLONG**)

Completion code.

This field is meaningful only for a response; its value is not significant for a command. The following are possible:

MQCC_OK

Command completed successfully.

MQCC_WARNING

Command completed with warning.

MQCC_FAILED

Command failed.

MQCC_UNKNOWN

Whether command succeeded is not known.

Reason (**MQLONG**)

Reason code qualifying completion code.

This field is meaningful only for a response; its value is not significant for a command.

The possible reason codes that could be returned in response to a command are listed in Chapter 3, “Definitions of the Programmable Command Formats,” on page 25, and in the description of each command.

ParameterCount (**MQLONG**)

Count of parameter structures.

This is the number of parameter structures (MQCFBF, MQCFBS, MQCFIF, MQCFIL, MQCFIN, MQCFSL, MQCFSF, and MQCFST) that follow the MQCFH structure. The value of this field is zero or greater.

Language declarations

This structure is available in the following languages:

C language declaration

```
typedef struct tagMQCFH {
    MQLONG Type;           /* Structure type */
    MQLONG StrucLength;   /* Structure length */
    MQLONG Version;        /* Structure version number */
    MQLONG Command;        /* Command identifier */
    MQLONG MsgSeqNumber;  /* Message sequence number */
    MQLONG Control;        /* Control options */
    MQLONG CompCode;       /* Completion code */
    MQLONG Reason;         /* Reason code qualifying completion code */
    MQLONG ParameterCount; /* Count of parameter structures */
} MQCFH;
```

COBOL language declaration

```
**   MQCFH structure
    10 MQCFH.
    **   Structure type
        15 MQCFH-TYPE          PIC S9(9) BINARY.
    **   Structure length
        15 MQCFH-STRUCLLENGTH  PIC S9(9) BINARY.
    **   Structure version number
        15 MQCFH-VERSION        PIC S9(9) BINARY.
    **   Command identifier
        15 MQCFH-COMMAND       PIC S9(9) BINARY.
    **   Message sequence number
        15 MQCFH-MSGSEQNUMBER  PIC S9(9) BINARY.
    **   Control options
        15 MQCFH-CONTROL        PIC S9(9) BINARY.
    **   Completion code
        15 MQCFH-COMPLETECODE   PIC S9(9) BINARY.
    **   Reason code qualifying completion code
        15 MQCFH-REASON         PIC S9(9) BINARY.
    **   Count of parameter structures
        15 MQCFH-PARAMETERCOUNT PIC S9(9) BINARY.
```

PL/I language declaration (z/OS only)

```
dcl
 1 MQCFH based,
 3 Type      fixed bin(31), /* Structure type */
 3 StrucLength fixed bin(31), /* Structure length */
 3 Version    fixed bin(31), /* Structure version number */
 3 Command    fixed bin(31), /* Command identifier */
 3 MsgSeqNumber fixed bin(31), /* Message sequence number */
 3 Control    fixed bin(31), /* Control options */
 3 CompCode   fixed bin(31), /* Completion code */
 3 Reason     fixed bin(31), /* Reason code qualifying completion
                           code */

 3 ParameterCount fixed bin(31); /* Count of parameter structures */
```

System/390 assembler-language declaration (z/OS only)

MQCFH	DSECT	
MQCFH_TYPE	DS F	Structure type
MQCFH_STRUCLLENGTH	DS F	Structure length
MQCFH_VERSION	DS F	Structure version number
MQCFH_COMMAND	DS F	Command identifier
MQCFH_MSGSEQNUMBER	DS F	Message sequence number
MQCFH_CONTROL	DS F	Control options
MQCFH_COMP_CODE	DS F	Completion code
MQCFH_REASON	DS F	Reason code qualifying completion code
*		
MQCFH_PARAMETERCOUNT	DS F	Count of parameter structures
*		
MQCFH_LENGTH	EQU *-MQCFH	Length of structure
*		
MQCFH_AREA	ORG MQCFH	
	DS CL(MQCFH_LENGTH)	

Visual Basic language declaration (Windows only)

Type MQCFH	
Type As Long	'Structure type
StrucLength As Long	'Structure length
Version As Long	'Structure version number
Command As Long	'Command identifier
MsgSeqNumber As Long	'Message sequence number
Control As Long	'Control options
CompCode As Long	'Completion code
Reason As Long	'Reason code qualifying completion code

```
    ParameterCount As Long  'Count of parameter structures
End Type
```

```
Global MQCFH_DEFAULT As MQCFH
```

RPG language declaration (iSeries only)

```
D*..1.....:....2.....:....3.....:....4.....:....5.....:....6.....:....7...
D* MQCFH Structure
D*
D* Structure type
D  FHTYP           1      4I 0 INZ(1)
D* Structure length
D  FHLEN           5      8I 0 INZ(36)
D* Structure version number
D  FHVER           9      12I 0 INZ(1)
D* Command identifier
D  FHCMD          13      16I 0 INZ(0)
D* Message sequence number
D  FHSEQ           17      20I 0 INZ(1)
D* Control options
D  FHCTL           21      24I 0 INZ(1)
D* Completion code
D  FHCMP           25      28I 0 INZ(0)
D* Reason code qualifying completion code
D  FHREA           29      32I 0 INZ(0)
D* Count of parameter structures
D  FHCNT          33      36I 0 INZ(0)
D*
```

MQCFBF - PCF byte string filter parameter

The MQCFBF structure describes a byte string filter parameter. The format name in the message descriptor is MQFMT_ADMIN.

The MQCFBF structure is used in Inquire commands to provide a filter description. This filter description is used to filter the results of the Inquire command and return to the user only those objects that satisfy the filter description.

When an MQCFBF structure is present, the Version field in the MQCFH structure at the start of the PCF must be MQCFH_VERSION_3 or higher.

Fields

Type (MQLONG)

Structure type.

This indicates that the structure is a MQCFBF structure describing a byte string filter parameter. The value must be:

MQCFT_BYTE_STRING_FILTER

Structure defining a byte string filter.

StrucLength (MQLONG)

Structure length.

This is the length, in bytes, of the MQCFBF structure, including the string at the end of the structure (the *FilterValue* field). The length must be a multiple of 4, and must be sufficient to contain the string. Bytes between the end of the string and the length defined by the *StrucLength* field are not significant.

The following constant gives the length of the *fixed* part of the structure, that is the length excluding the *FilterValue* field:

MQCFBF_STRUC_LENGTH_FIXED

Length of fixed part of command format filter string-parameter structure.

Parameter (MQLONG)

Parameter identifier.

This identifies the parameter that is to be filtered on. The value of this identifier depends on the parameter to be filtered on.

The parameter is one of the following:

- MQBACF_EXTERNAL_UOW_ID
- MQBACF_Q_MGR_UOW_ID
- MQBACF_ORIGIN_UOW_ID (on z/OS only)

Operator (MQLONG)

Operator identifier.

This identifies the operator that is being used to evaluate whether the parameter satisfies the filter-value.

Possible values are:

MQCFOP_GREATER

Greater than

MQCFOP_LESS

Less than

MQCFOP_EQUAL

Equal to

MQCFOP_NOT_EQUAL

Not equal to

MQCFOP_NOT_LESS

Greater than or equal to

MQCFOP_NOT_GREATER

Less than or equal to

FilterValueLength (MQLONG)

Length of filter-value string.

This is the length, in bytes, of the data in the *FilterValue* field. This must be zero or greater, and does not need to be a multiple of 4.

FilterValue (MQBYTE×FilterValueLength)

Filter value.

This specifies the filter-value that must be satisfied. Use this parameter where the response type of the filtered parameter is a byte string. Depending on the filter-keyword, this can be:

Note: If the specified byte string is shorter than the standard length of the parameter in MQFMT_ADMIN command messages, the omitted characters are assumed to be blanks. If the specified string is longer than the standard length, it is an error.

Language declarations

This structure is available in the following languages:

C language declaration

```
typedef struct tagMQCFBF {  
    MQLONG Type; /* Structure type */  
    MQLONG StrucLength; /* Structure length */  
    MQLONG Parameter; /* Parameter identifier */  
    MQLONG Operator; /* Operator identifier */  
    MQLONG FilterValueLength; /* Filter value length */  
    MQBYTE FilterValue[1]; /* Filter value -- first byte */  
} MQCFBF;
```

COBOL language declaration

```
** MQCFBF structure  
10 MQCFBF.  
** Structure type  
15 MQCFBF-TYPE PIC S9(9) BINARY.  
** Structure length  
15 MQCFBF-STRUCLENGTH PIC S9(9) BINARY.  
** Parameter identifier  
15 MQCFBF-PARAMETER PIC S9(9) BINARY.  
** Operator identifier  
15 MQCFBF-OPERATOR PIC S9(9) BINARY.  
** Filter value length  
15 MQCFBF-FILTERVALUELENGTH PIC S9(9) BINARY.
```

PL/I language declaration (z/OS only)

```
dcl  
1 MQCFBF based,  
3 Type fixed bin(31)  
    init(MQCFT_BYTE_STRING_FILTER), /* Structure type */  
3 StrucLength fixed bin(31)  
    init(MQCFBF_STRUC_LENGTH_FIXED), /* Structure length */  
3 Parameter fixed bin(31)  
    init(0), /* Parameter identifier */  
3 Operator fixed bin(31)  
    init(0), /* Operator identifier */  
3 FilterValueLength fixed bin(31)  
    init(0); /* Filter value length */
```

System/390 assembler-language declaration (z/OS only)

MQCFBF	DSECT
MQCFBF_TYPE	DS F Structure type
MQCFBF_STRUCLENGTH	DS F Structure length
MQCFBF_PARAMETER	DS F Parameter identifier
MQCFBF_OPERATOR	DS F Operator identifier
MQCFBF_FILTERVALUELENGTH	DS F Filter value length
MQCFBF_LENGTH	EQU *--MQCFIF Length of structure
MQCFBF_AREA	ORG MQCFBF
	DS CL(MQCFBF_LENGTH)

Visual Basic language declaration (Windows only)

```
Type MQCFBF  
    Type As Long 'Structure type'  
    StrucLength As Long 'Structure length'  
    Parameter As Long 'Parameter identifier'  
    Operator As Long 'Operator identifier'  
    FilterValueLength As Long 'Filter value length'  
    FilterValue As 1 'Filter value -- first byte'  
End Type  
Global MQCFBF_DEFAULT As MQCFBF
```

RPG language declaration (iSeries only)

```
D* MQCFBF Structure  
D*  
D* Structure type
```

D FBFTYP	1	4I 0 INZ(15)
D* Structure length		
D FBFLEN	5	8I 0 INZ(20)
D* Parameter identifier		
D FBFPRM	9	12I 0 INZ(0)
D* Operator identifier		
D FBFOP	13	16I 0 INZ(0)
D* Filter value length		
D FBFFVL	17	20I 0 INZ(0)
D* Filter value -- first byte		
D FBFFV	21	21 INZ

MQCFBS - PCF byte string parameter

The MQCFBS structure describes a byte-string parameter in a PCF message. The format name in the message descriptor is MQFMT_ADMIN.

When an MQCFBS structure is present, the *Version* field in the MQCFH structure at the start of the PCF must be MQCFH_VERSION_2 or greater.

In a user PCF message, the *Parameter* field has no significance, and can be used by the application for its own purposes.

The structure ends with a variable-length byte string; see the *String* field below for further details.

Fields

Type (MQLONG)

Structure type.

This indicates that the structure is an MQCFBS structure describing byte string parameter. The value must be:

MQCFT_BYTE_STRING

Structure defining a byte string.

StrucLength (MQLONG)

Structure length.

This is the length in bytes of the MQCFBS structure, including the variable-length string at the end of the structure (the *String* field). The length must be a multiple of four, and must be sufficient to contain the string; any bytes between the end of the string and the length defined by the *StrucLength* field are not significant.

The following constant gives the length of the *fixed* part of the structure, that is the length excluding the *String* field:

MQCFBS_STRUC_LENGTH_FIXED

Length of fixed part of MQCFBS structure.

Parameter (MQLONG)

Parameter identifier.

This identifies the parameter whose value is contained in the structure. The values that can occur in this field depend on the value of the *Command* field in the MQCFH structure; see “MQCFH - PCF header” on page 418 for details. In user PCF messages (MQCFT_USER), this field has no significance.

The parameter is from the MQBACF_* group of parameters.

StringLength (MQLONG)

Length of string.

This is the length in bytes of the data in the *string* field; it must be zero or greater. This length need not be a multiple of four.

String (MQBYTE×*StringLength*)

String value.

This is the value of the parameter identified by the *parameter* field. The string is a byte string, and so is not subject to character-set conversion when sent between different systems.

Note: A null character in the string is treated as normal data, and does not act as a delimiter for the string

For MQFMT_ADMIN messages, if the specified string is shorter than the standard length of the *parameter*, the omitted characters are assumed to be nulls. If the specified string is longer than the standard length, it is an error.

The way that this field is declared depends on the programming language:

- For the C programming language, the field is declared as an array with one element. Storage for the structure must be allocated dynamically, and pointers used to address the fields within it.
- For other programming languages, the field is omitted from the structure declaration. When an instance of the structure is declared, you must include MQCFBS in a larger structure, and declare additional fields following MQCFBS, to represent the *String* field as required.

Language declarations

This structure is available in the following languages:

C language declaration

```
typedef struct tagMQCFBS {
    MQLONG Type;           /* Structure type */
    MQLONG StrucLength;   /* Structure length */
    MQLONG Parameter;     /* Parameter identifier */
    MQLONG StringLength;  /* Length of string */
    MQBYTE String[1];      /* String value - first byte */
} MQCFBS;
```

COBOL language declaration

```
**  MQCFBS structure
10 MQCFBS.
**  Structure type
15 MQCFBS-TYPE      PIC S9(9) BINARY.
**  Structure length
15 MQCFBS-STRUCLENGTH PIC S9(9) BINARY.
**  Parameter identifier
15 MQCFBS-PARAMETER PIC S9(9) BINARY.
**  Length of string
15 MQCFBS-STRINGLENGTH PIC S9(9) BINARY.
```

PL/I language declaration (z/OS only)

```
dcl
1 MQCFBS based,
  3 Type      fixed bin(31), /* Structure type */
```

```

3 StrucLength fixed bin(31), /* Structure length */
3 Parameter   fixed bin(31), /* Parameter identifier */
3 StringLength fixed bin(31)  /* Length of string */

```

System/390 assembler-language declaration (z/OS only)

MQCFBS	DSECT
MQCFBS_TYPE	DS F Structure type
MQCFBS_STRUCLENGTH	DS F Structure length
MQCFBS_PARAMETER	DS F Parameter identifier
MQCFBS_STRINGLENGTH	DS F Length of string
	ORG MQCFBS
MQCFBS_AREA	DS CL(MQCFBS_LENGTH)

Visual Basic language declaration (Windows only)

```

Type MQCFBS
    Type As Long      ' Structure type
    StrucLength As Long ' Structure length
    Parameter As Long   ' Parameter identifier
    StringLength As Long ' Operator identifier
    String as 1        ' String value - first byte
End Type

```

```
Global MQCFBS_DEFAULT As MQCFBS
```

RPG language declaration (iSeries only)

```

D* MQCFBS Structure
D*
D* Structure type
D  BSTYP          1     4I 0 INZ(3)
D* Structure length
D  BSLEN          5     8I 0 INZ(16)
D* Parameter identifier
D  BSPRM          9     12I 0 INZ(0)
D* Length of string
D  BSSTL         13     16I 0 INZ(0)
D* String value - first byte
D  BSSRA         17     16
D*

```

MQCFIF - PCF integer filter parameter

The MQCFIF structure describes an integer filter parameter. The format name in the message descriptor is MQFMT_ADMIN.

The MQCFIF structure is used in Inquire commands to provide a filter condition. This filter condition is used to filter the results of the Inquire command and return to the user only those objects that satisfy the filter condition.

When an MQCFIF structure is present, the Version field in the MQCFH structure at the start of the PCF must be MQCFH_VERSION_3 or higher.

Fields

Type (MQLONG)

Structure type.

This indicates that the structure is a MQCFIF structure describing an integer filter parameter. The value must be:

MQCFT_INTEGER_FILTER

Structure defining an integer filter.

StrucLength (**MQLONG**)

Structure length.

This is the length in bytes of the MQCFIF structure. The value must be:

MQCFIF_STRUC_LENGTH

Length of command format integer-parameter structure.

Parameter (**MQLONG**)

Parameter identifier.

This identifies the parameter that is to be filtered on. The value of this identifier depends on the parameter to be filtered on. Any of the parameters which can be used in the Inquire command can be used in this field.

The parameter is from the following groups of parameters:

- MQIA_*
- MQIACF_*
- MQIAMO_*
- MQIACH_*

Operator (**MQLONG**)

Operator identifier.

This identifies the operator that is being used to evaluate whether the parameter satisfies the filter-value.

Possible values are:

MQCFOP_GREATER

Greater than

MQCFOP_LESS

Less than

MQCFOP_EQUAL

Equal to

MQCFOP_NOT_EQUAL

Not equal to

MQCFOP_NOT_LESS

Greater than or equal to

MQCFOP_NOT_GREATER

Less than or equal to

MQCFOP_CONTAINS

Contains a specified value. Use this when filtering on lists of values or integers.

MQCFOP_EXCLUDES

Does not contain a specified value. Use this when filtering on lists of values or integers.

See the *FilterValue* description for details telling you which operators may be used in which circumstances.

FilterValue (**MQLONG**)

Filter value identifier.

This specifies the filter-value that must be satisfied.

Depending on the parameter, the value and the permitted operators can be:

- An explicit integer value, if the parameter takes a single integer value.

You can only use the following operators:

- MQCFOP_GREATER
- MQCFOP_LESS
- MQCFOP_EQUAL
- MQCFOP_NOT_EQUAL
- MQCFOP_NOT_GREATER
- MQCFOP_NOT_LESS

- An MQ constant, if the parameter takes a single value from a possible set of values (for example, the value MQCHT_SENDER on the *ChannelType* parameter). You can only use MQCFOP_EQUAL or MQCFOP_NOT_EQUAL.
- An explicit value or an MQ constant, as the case may be, if the parameter takes a list of values. You can use either MQCFOP_CONTAINS or MQCFOP_EXCLUDES. For example, if the value 6 is specified with the operator MQCFOP_CONTAINS, all items where one of the parameter values is 6 are listed.

| For example, if you need to filter on queues that are enabled for put operations in your Inquire Queue command, the parameter would be MQIA_INHIBIT_PUT and the filter-value would be MQQA_PUT_ALLOWED.

The filter value must be a valid value for the parameter being tested.

Language declarations

This structure is available in the following languages:

C language declaration

```
typedef struct tagMQCFIF {  
    MQLONG Type;           /* Structure type */  
    MQLONG StrucLength;   /* Structure length */  
    MQLONG Parameter;     /* Parameter identifier */  
    MQLONG Operator;      /* Operator identifier */  
    MQLONG FilterValue;   /* Filter value */  
} MQCFIF;
```

COBOL language declaration

```
**  MQCFIF structure  
10 MQCFIF.  
**  Structure type  
15 MQCFIF-TYPE      PIC S9(9) BINARY.  
**  Structure length  
15 MQCFIF-STRUCLLENGTH PIC S9(9) BINARY.  
**  Parameter identifier  
15 MQCFIF-PARAMETER  PIC S9(9) BINARY.  
**  Operator identifier  
15 MQCFIF-OPERATOR   PIC S9(9) BINARY.  
**  Filter value  
15 MQCFIF-FILTERVALUE PIC S9(9) BINARY.
```

PL/I language declaration (z/OS only)

```
dcl  
1 MQCFIF based,  
3 Type      fixed bin(31), /* Structure type */  
3 StrucLength fixed bin(31), /* Structure length */  
3 Parameter  fixed bin(31), /* Parameter identifier */  
3 Operator   fixed bin(31) /* Operator identifier */  
3 FilterValue fixed bin(31); /* Filter value */
```

System/390 assembler-language declaration (z/OS only)

```
MQCFIF          DSECT
MQCFIF_TYPE     DS F      Structure type
MQCFIF_STRCLLENGTH DS F      Structure length
MQCFIF_PARAMETER DS F      Parameter identifier
MQCFIF_OPERATOR  DS F      Operator identifier
MQCFIF_FILTERVALUE DS F      Filter value
MQCFIF_LENGTH    EQU  *-MQCFIF Length of structure
MQCFIF_AREA      ORG  MQCFIF
                  DS CL(MQCFIF_LENGTH)
```

Visual Basic language declaration (Windows only)

```
Type MQCFIF
  Type As Long      ' Structure type
  StrucLength As Long ' Structure length
  Parameter As Long   ' Parameter identifier
  Operator As Long    ' Operator identifier
  FilterValue As Long  ' Filter value
End Type

Global MQCFIF_DEFAULT As MQCFIF
```

RPG language declaration (iSeries only)

```
D* MQCFIF Structure
D*
D* Structure type
D  FIFTYP           1      4I 0 INZ(3)
D* Structure length
D  FIFLEN            5      8I 0 INZ(16)
D* Parameter identifier
D  FIFPRM            9      12I 0 INZ(0)
D* Operator identifier
D  FIFOP             13     16I 0 INZ(0)
D* Condition identifier
D  FIFFV              17     20I 0 INZ(0)
D*
```

MQCFIL - PCF integer list parameter

The MQCFIL structure describes an integer-list parameter in a message that is a command or a response to a command. In either case, the format name in the message descriptor is MQFMT_ADMIN.

The MQCFIL structure can also be used for user-defined message data. In this case the message descriptor *Format* field is MQFMT_PCF (see “Message descriptor for a PCF command” on page 11). Also in this case, not all of the fields in the structure are meaningful. The supplied initial values can be used for most fields, but the application must set the *StrucLength*, *Count*, and *Values* fields to the values appropriate to the data.

The structure ends with a variable-length array of integers; see the *Values* field below for further details.

Fields

Type (MQLONG)

Structure type.

This indicates that the structure is an MQCFIL structure describing an integer-list parameter. The value must be:

MQCFT_INTEGER_LIST

Structure defining an integer list.

StrucLength (MQLONG)

Structure length.

This is the length in bytes of the MQCFIL structure, including the array of integers at the end of the structure (the *Values* field). The length must be a multiple of four, and must be sufficient to contain the array; any bytes between the end of the array and the length defined by the *StrucLength* field are not significant.

The following constant gives the length of the *fixed* part of the structure, that is the length excluding the *Values* field:

MQCFIL_STRUC_LENGTH_FIXED

Length of fixed part of command format integer-list parameter structure.

Parameter (MQLONG)

Parameter identifier.

This identifies the parameter whose values are contained in the structure. The values that can occur in this field depend on the value of the *Command* field in the MQCFH structure; see “MQCFH - PCF header” on page 418 for details.

The parameter is from the following groups of parameters:

- MQIA_*
- MQIACF_*
- MQIAMO_*
- MQIACH_*

Count (MQLONG)

Count of parameter values.

This is the number of elements in the *Values* array; it must be zero or greater.

Values (MQLONG×Count)

Parameter values.

This is an array of values for the parameter identified by the *Parameter* field. For example, for MQIACF_Q_ATTRS, this is a list of attribute selectors (MQCA_* and MQIA_* values).

The way that this field is declared depends on the programming language:

- For the C programming language, the field is declared as an array with one element. Storage for the structure must be allocated dynamically, and pointers used to address the fields within it.
- For the COBOL, PL/I, RPG, and System/390® assembler programming languages, the field is omitted from the structure declaration. When an instance of the structure is declared, you must include MQCFIL in a larger structure, and declare additional fields following MQCFIL, to represent the *Values* field as required.

Language declarations

This structure is available in the following languages:

C language declaration

```
typedef struct tagMQCFIL {  
    MQLONG Type;          /* Structure type */  
    MQLONG StrucLength;   /* Structure length */  
    MQLONG Parameter;    /* Parameter identifier */  
    MQLONG Count;         /* Count of parameter values */  
    MQLONG Values[1];     /* Parameter values - first element */  
} MQCFIL;
```

COBOL language declaration

```
**  MQCFIL structure  
10 MQCFIL.  
**    Structure type  
    15 MQCFIL-TYPE      PIC S9(9) BINARY.  
**    Structure length  
    15 MQCFIL-STRUCLENGTH PIC S9(9) BINARY.  
**    Parameter identifier  
    15 MQCFIL-PARAMETER  PIC S9(9) BINARY.  
**    Count of parameter values  
    15 MQCFIL-COUNT     PIC S9(9) BINARY.
```

PL/I language declaration (z/OS only)

```
dcl  
1 MQCFIL based,  
 3 Type      fixed bin(31), /* Structure type */  
 3 StrucLength fixed bin(31), /* Structure length */  
 3 Parameter  fixed bin(31), /* Parameter identifier */  
 3 Count      fixed bin(31); /* Count of parameter values */
```

System/390 assembler-language declaration (z/OS only)

MQCFIL	DSECT
MQCFIL_TYPE	DS F Structure type
MQCFIL_STRUCLENGTH	DS F Structure length
MQCFIL_PARAMETER	DS F Parameter identifier
MQCFIL_COUNT	DS F Count of parameter values
MQCFIL_LENGTH	EQU *-MQCFIL Length of structure
MQCFIL_AREA	ORG MQCFIL
	DS CL(MQCFIL_LENGTH)

Visual Basic language declaration (Windows only)

```
Type MQCFIL  
    Type As Long      ' Structure type  
    StrucLength As Long ' Structure length  
    Parameter As Long ' Parameter identifier  
    Count As Long     ' Count of parameter values  
End Type
```

Global MQCFIL_DEFAULT As MQCFIL

RPG language declaration (iSeries only)

```
D* MQCFIL Structure  
D*  
D* Structure type  
D ILTYP           1   4I 0 INZ(5)  
D* Structure length  
D ILEN            5   8I 0 INZ(16)  
D* Parameter identifier  
D ILPRM           9   12I 0 INZ(0)  
D* Count of parameter values  
D ILCNT          13   16I 0 INZ(0)  
D*
```

MQCFIN - PCF integer parameter

The MQCFIN structure describes an integer parameter in a message that is a command or a response to a command. In either case, the format name in the message descriptor is MQFMT_ADMIN.

The MQCFIN structure can also be used for user-defined message data. In this case the message descriptor *Format* field is MQFMT_PCF (see “Message descriptor for a PCF command” on page 11). Also in this case, not all of the fields in the structure are meaningful. The supplied initial values can be used for most fields, but the application must set the *Value* field to the value appropriate to the data.

Fields

Type (MQLONG)

Structure type.

This indicates that the structure is a MQCFIN structure describing an integer parameter. The value must be:

MQCFT_INTEGER

Structure defining an integer.

StrucLength (MQLONG)

Structure length.

This is the length in bytes of the MQCFIN structure. The value must be:

MQCFIN_STRUC_LENGTH

Length of command format integer-parameter structure.

Parameter (MQLONG)

Parameter identifier.

This identifies the parameter whose value is contained in the structure. The values that can occur in this field depend on the value of the *Command* field in the MQCFH structure; see “MQCFH - PCF header” on page 418 for details.

The parameter is from the following groups of parameters:

- MQIA_*
- MQIACF_*
- MQIAMO_*
- MQIACH_*

Value (MQLONG)

Parameter value.

This is the value of the parameter identified by the *Parameter* field.

Language declarations

This structure is available in the following languages:

C language declaration

```
typedef struct tagMQCFIN {
    MQLONG Type;           /* Structure type */
    MQLONG StrucLength;   /* Structure length */
    MQLONG Parameter;     /* Parameter identifier */
    MQLONG Value;          /* Parameter value */
} MQCFIN;
```

COBOL language declaration

```
**   MQCFIN structure
10 MQCFIN.
**   Structure type
15 MQCFIN-TYPE      PIC S9(9) BINARY.
**   Structure length
15 MQCFIN-STRUCLENGTH PIC S9(9) BINARY.
**   Parameter identifier
15 MQCFIN-PARAMETER  PIC S9(9) BINARY.
**   Parameter value
15 MQCFIN-VALUE      PIC S9(9) BINARY.
```

PL/I language declaration (z/OS only)

```
dcl
1 MQCFIN based,
3 Type      fixed bin(31), /* Structure type */
3 StrucLength fixed bin(31), /* Structure length */
3 Parameter  fixed bin(31), /* Parameter identifier */
3 Value     fixed bin(31); /* Parameter value */
```

System/390 assembler-language declaration (z/OS only)

	DSECT	
MQCFIN	DS F	Structure type
MQCFIN_TYPE	DS F	Structure length
MQCFIN_STRUCLENGTH	DS F	Parameter identifier
MQCFIN_PARAMETER	DS F	Parameter value
MQCFIN_VALUE	EQU	*-MQCFIN Length of structure
MQCFIN_LENGTH	ORG	MQCFIN
MQCFIN_AREA	DS	CL(MQCFIN_LENGTH)

Visual Basic language declaration (Windows only)

```
Type MQCFIN
    Type As Long      ' Structure type
    StrucLength As Long ' Structure length
    Parameter As Long   ' Parameter identifier
    Value As Long       ' Parameter value
End Type
```

```
Global MQCFIN_DEFAULT As MQCFIN
```

RPG language declaration (iSeries only)

```
D* MQCFIN Structure
D*
D* Structure type
D  INTYP           1      4I 0 INZ(3)
D* Structure length
D  INLEN            5      8I 0 INZ(16)
D* Parameter identifier
D  INPRM            9      12I 0 INZ(0)
D* Parameter value
D  INVAL           13      16I 0 INZ(0)
D*
```

MQCFSF - PCF string filter parameter

The MQCFSF structure describes a string filter parameter. The format name in the message descriptor is MQFMT_ADMIN.

The MQCFSF structure is used in Inquire commands to provide a filter condition. This filter condition is used to filter the results of the Inquire command and return to the user only those objects that satisfy the filter condition.

The results of filtering character strings on EBCDIC-based systems may be different from those achieved on ASCII-based systems. This is because comparison of character strings is based on the collating sequence of the internal built-in values representing the characters.

When an MQCFSF structure is present, the Version field in the MQCFH structure at the start of the PCF must be MQCFH_VERSION_3 or higher.

Fields

Type (MQLONG)

Structure type.

This indicates that the structure is a MQCFSF structure describing a string filter parameter. The value must be:

MQCFT_STRING_FILTER

Structure defining a string filter.

StrucLength (MQLONG)

Structure length.

This is the length in bytes of the MQCFSF structure. The value must be:

MQCFSF_STRUC_LENGTH

This is the length, in bytes, of the MQCFSF structure, including the string at the end of the structure (the *FilterValue* field). The length must be a multiple of 4, and must be sufficient to contain the string. Bytes between the end of the string and the length defined by the *StrucLength* field are not significant.

The following constant gives the length of the *fixed* part of the structure, that is the length excluding the *FilterValue* field:

MQCFSF_STRUC_LENGTH_FIXED

Length of fixed part of command format filter string-parameter structure.

Parameter (MQLONG)

Parameter identifier.

This identifies the parameter that is to be filtered on. The value of this identifier depends on the parameter to be filtered on. Any of the parameters which can be used in the Inquire command can be used in this field.

The parameter is from the following groups of parameters:

- MQCA_*
- MQCACF_*
- MQCAMO_*
- MQCACH_*

Operator (MQLONG)

Operator identifier.

This identifies the operator that is being used to evaluate whether the parameter satisfies the filter-value.

Possible values are:

MQCFOP_GREATER

Greater than

MQCFOP_LESS
Less than

MQCFOP_EQUAL
Equal to

MQCFOP_NOT_EQUAL
Not equal to

MQCFOP_NOT_LESS
Greater than or equal to

MQCFOP_NOT_GREATER
Less than or equal to

MQCFOP_LIKE
Matches a generic string

MQCFOP_NOT_LIKE
Does not match a generic string

MQCFOP_CONTAINS
Contains a specified string. Use this when filtering on lists of strings.

MQCFOP_EXCLUDES
Does not contain a specified string. Use this when filtering on lists of strings.

MQCFOP_CONTAINS_GEN
Contains an item which matches a generic string. Use this when filtering on lists of strings.

MQCFOP_EXCLUDES_GEN
Does not contain any item which matches a generic string. Use this when filtering on lists of strings.

See the *FilterValue* description for details telling you which operators may be used in which circumstances.

CodedCharSetId (MQLONG)

Coded character set identifier.

This specifies the coded character set identifier of the data in the *FilterValue* field. The following special value can be used:

MQCCSI_DEFAULT

Default character set identifier.

The string data is in the character set defined by the *CodedCharSetId* field in the MQ header structure that precedes the MQCFH structure, or by the *CodedCharSetId* field in the MQMD if the MQCFH structure is at the start of the message.

FilterValueLength (MQLONG)

Length of filter-value string.

This is the length, in bytes, of the data in the *FilterValue* field. This must be zero or greater, and does not need to be a multiple of 4.

FilterValue (MQCHAR×*FilterValueLength*)

Filter value.

This specifies the filter-value that must be satisfied. Depending on the parameter, the value and the permitted operators can be:

- An explicit string value.

You can only use the following operators:

- MQCFOP_GREATER
- MQCFOP_LESS
- MQCFOP_EQUAL
- MQCFOP_NOT_EQUAL
- MQCFOP_NOT_GREATER
- MQCFOP_NOT_LESS

- A generic string value. This is a character string with an asterisk at the end, for example ABC*. The operator must be either MQCFOP_LIKE or MQCFOP_NOT_LIKE. The characters must be valid for the attribute you are testing. If the operator is MQCFOP_LIKE, all items where the attribute value begins with the string (ABC in the example) are listed. If the operator is MQCFOP_NOT_LIKE, all items where the attribute value does not begin with the string are listed.
- If the parameter takes a list of string values, the operator can be:
 - MQCFOP_CONTAINS
 - MQCFOP_EXCLUDES
 - MQCFOP_CONTAINS_GEN
 - MQCFOP_EXCLUDES_GEN

An item in a list of values. The value can be explicit or generic. If it is explicit, use MQCFOP_CONTAINS or MQCFOP_EXCLUDES as the operator. For example, if the value DEF is specified with the operator MQCFOP_CONTAINS, all items where one of the attribute values is DEF are listed. If it is generic, use MQCFOP_CONTAINS_GEN or MQCFOP_EXCLUDES_GEN as the operator. If ABC* is specified with the operator MQCFOP_CONTAINS_GEN, all items where one of the attribute values begins with ABC are listed.

Note:

1. If the specified string is shorter than the standard length of the parameter in MQFMT_ADMIN command messages, the omitted characters are assumed to be blanks. If the specified string is longer than the standard length, it is an error.
2. When the queue manager reads an MQCFSF structure in an MQFMT_ADMIN message from the command input queue, the queue manager processes the string as though it had been specified on an MQI call. This means that within the string, the first null and the characters following it (up to the end of the string) are treated as blanks.

The filter value must be a valid value for the parameter being tested.

Language declarations

This structure is available in the following languages:

C language declaration

```
typedef struct tagMQCFSF {
    MQLONG Type;          /* Structure type */
    MQLONG StrucLength;   /* Structure length */
    MQLONG Parameter;     /* Parameter identifier */
    MQLONG Operator;      /* Operator identifier */
    MQLONG CodedCharSetId; /* Coded character set identifier */
    MQLONG FilterValueLength /* Filtervalue length */
    MQCHAR[1] FilterValue; /* Filter value */
} MQCFSF;
```

COBOL language declaration

```
**  MQCSF structure
10 MQCSF.
**    Structure type
15 MQCSF-TYPE      PIC S9(9) BINARY.
**    Structure length
15 MQCSF-STRUCLENGTH PIC S9(9) BINARY.
**    Parameter identifier
15 MQCSF-PARAMETER  PIC S9(9) BINARY.
**    Operator identifier
15 MQCSF-OPERATOR   PIC S9(9) BINARY.
**    Coded character set identifier
15 MQCSF-CODEDCHARSETID PIC S9(9) BINARY.
**    Filter value length
15 MQCSF-FILTERVALUE PIC S9(9) BINARY.
```

PL/I language declaration (z/OS only)

```
dcl
1 MQCSF based,
3 Type      fixed bin(31), /* Structure type */
3 StrucLength fixed bin(31), /* Structure length */
3 Parameter  fixed bin(31), /* Parameter identifier */
3 Operator   fixed bin(31) /* Operator identifier */
3 CodedCharSetId  fixed bin(31) /* Coded character set identifier */
3 FilterValueLength fixed bin(31); /* Filter value length */
```

System/390 assembler-language declaration (z/OS only)

MQCSF	DSECT
MQCSF_TYPE	DS F Structure type
MQCSF_STRUCLENGTH	DS F Structure length
MQCSF_PARAMETER	DS F Parameter identifier
MQCSF_OPERATOR	DS F Operator identifier
MQCSF_CODEDCHARSETID	DS F Coded character set identifier
MQCSF_FILTERVALUELENGTH	DS F Filter value length
MQCSF_LENGTH	EQU *-MQCSF Length of structure
MQCSF_AREA	ORG MQCSF
	DS CL(MQCSF_LENGTH)

Visual Basic language declaration (Windows only)

```
Type MQCSF
    Type As Long      ' Structure type
    StrucLength As Long ' Structure length
    Parameter As Long  ' Parameter identifier
    Operator As Long   ' Operator identifier
    CodedCharSetId As Long ' Coded character set identifier
    FilterValueLength As Long ' Operator identifier
    FilterValue As String*1 ' Condition value -- first character
End Type

Global MQCSF_DEFAULT As MQCSF
```

RPG language declaration (iSeries only)

```
D* MQCSF Structure
D*
D* Structure type
D  FISTYP           1     4I 0 INZ(3)
D* Structure length
D  FSFLEN            5     8I 0 INZ(16)
D* Parameter identifier
D  FSFPRM            9     12I 0 INZ(0)
D* Reserved field
D  FSFRSV           13    16I 0 INZ(0)
D* Parameter value
D  FSFVAL            17    16
D* Structure type
```

D FSFTYP	17	20I 0
D* Structure length		
D FSFLEN	21	24I 0
D* Parameter value		
D FSFPRM	25	28I 0
D* Operator identifier		
D FSFOP	29	32I 0
D* Coded character set identifier		
D FSFCSI	33	36I 0
D* Length of condition		
D FSFFVL	37	40 0
D* Condition value -- first character		
D FSFFV	41	41
D*		

MQCFSL - PCF string list parameter

The MQCFSL structure describes a string-list parameter in a message which is a command or a response to a command. In either case, the format name in the message descriptor is MQFMT_ADMIN.

The MQCFSL structure can also be used for user-defined message data. In this case the message descriptor *Format* field is MQFMT_PCF (see “Message descriptor for a PCF command” on page 11). Also in this case, not all of the fields in the structure are meaningful. The supplied initial values can be used for most fields, but the application must set the *StrucLength*, *Count*, *StringLength*, and *Strings* fields to the values appropriate to the data.

The structure ends with a variable-length array of character strings; see the *Strings* field below for further details.

See “Usage notes” on page 418 for further information on how to use the structure.

Fields

Type (MQLONG)

Structure type.

This indicates that the structure is an MQCFSL structure describing a string-list parameter. The value must be:

MQCFT_STRING_LIST

Structure defining a string list.

StrucLength (MQLONG)

Structure length.

This is the length in bytes of the MQCFSL structure, including the data at the end of the structure (the *Strings* field). The length must be a multiple of four, and must be sufficient to contain all of the strings; any bytes between the end of the strings and the length defined by the *StrucLength* field are not significant.

The following constant gives the length of the *fixed* part of the structure, that is the length excluding the *Strings* field:

MQCFSL_STRUC_LENGTH_FIXED

Length of fixed part of command format string-list parameter structure.

Parameter (MQLONG)

Parameter identifier.

This identifies the parameter whose values are contained in the structure. The values that can occur in this field depend on the value of the *Command* field in the MQCFH structure; see “MQCFH - PCF header” on page 418 for details.

The parameter is from the following groups of parameters:

- MQCA_*
- MQCACF_*
- MQCAMO_*
- MQCACH_*

CodedCharSetId (MQLONG)

Coded character set identifier.

This specifies the coded character set identifier of the data in the *Strings* field. The following special value can be used:

MQCCSI_DEFAULT

Default character set identifier.

The string data is in the character set defined by the *CodedCharSetId* field in the MQ header structure that precedes the MQCFH structure, or by the *CodedCharSetId* field in the MQMD if the MQCFH structure is at the start of the message.

Count (MQLONG)

Count of parameter values.

This is the number of strings present in the *Strings* field; it must be zero or greater.

StringLength (MQLONG)

Length of one string.

This is the length in bytes of one parameter value, that is the length of one string in the *Strings* field; all of the strings are this length. The length must be zero or greater, and need not be a multiple of four.

Strings (MQCHAR×*StringLength*×*Count*)

String values.

This is a set of string values for the parameter identified by the *Parameter* field. The number of strings is given by the *Count* field, and the length of each string is given by the *StringLength* field. The strings are concatenated together, with no bytes skipped between adjacent strings. The total length of the strings is the length of one string multiplied by the number of strings present (that is, *StringLength*×*Count*).

- In MQFMT_ADMIN command messages, if the specified string is shorter than the standard length of the parameter, the omitted characters are assumed to be blanks. If the specified string is longer than the standard length, it is an error.
- In MQFMT_ADMIN response messages, string parameters may be returned padded with blanks to the standard length of the parameter.
- In MQFMT_EVENT messages, trailing blanks may be omitted from string parameters (that is, the string may be shorter than the standard length of the parameter).

In all cases, *StringLength* gives the length of the string actually present in the message.

The strings can contain any characters that are in the character set defined by *CodedCharSetId*, and that are valid for the parameter identified by *Parameter*.

Note: When the queue manager reads an MQCFSL structure in an MQFMT_ADMIN message from the command input queue, the queue manager processes each string in the list as though it had been specified on an MQI call. This means that within each string, the first null and the characters following it (up to the end of the string) are treated as blanks.

In responses and all other cases, a null character in a string is treated as normal data, and does not act as a delimiter for the string. This means that when a receiving application reads a MQFMT_PCF, MQFMT_EVENT, or MQFMT_ADMIN message, the receiving application receives all of the data specified by the sending application.

The way that this field is declared depends on the programming language:

- For the C programming language, the field is declared as an array with one element. Storage for the structure must be allocated dynamically, and pointers used to address the fields within it.
- For the COBOL, PL/I, RPG, and System/390 assembler programming languages, the field is omitted from the structure declaration. When an instance of the structure is declared, you must include MQCFSL in a larger structure, and declare additional fields following MQCFSL, to represent the *Strings* field as required.

Language declarations

The declarations available for this structure are:

C language declaration

```
typedef struct tagMQCFSL {  
    MQLONG Type;           /* Structure type */  
    MQLONG StrucLength;    /* Structure length */  
    MQLONG Parameter;      /* Parameter identifier */  
    MQLONG CodedCharSetId; /* Coded character set identifier */  
    MQLONG Count;          /* Count of parameter values */  
    MQLONG StringLength;   /* Length of one string */  
    MQCHAR Strings[1];     /* String values - first  
                           character */  
} MQCFSL;
```

COBOL language declaration

```
**  MQCFSL structure  
10 MQCFSL.  
**  Structure type  
15 MQCFSL-TYPE      PIC S9(9) BINARY.  
**  Structure length  
15 MQCFSL-STRUCLength  PIC S9(9) BINARY.  
**  Parameter identifier  
15 MQCFSL-PARAMETER  PIC S9(9) BINARY.  
**  Coded character set identifier  
15 MQCFSL-CODEDCHARSETID PIC S9(9) BINARY.  
**  Count of parameter values  
15 MQCFSL-COUNT      PIC S9(9) BINARY.  
**  Length of one string  
15 MQCFSL-STRINGLENGTH PIC S9(9) BINARY.
```

PL/I language declaration (z/OS only)

```
dc1  
1 MQCFSL based,  
 3 Type      fixed bin(31), /* Structure type */  
 3 StrucLength fixed bin(31), /* Structure length */  
 3 Parameter   fixed bin(31), /* Parameter identifier */  
 3 CodedCharSetId fixed bin(31), /* Coded character set identifier */  
 3 Count       fixed bin(31), /* Count of parameter values */  
 3 StringLength fixed bin(31); /* Length of one string */
```

System/390 assembler-language declaration (z/OS only)

MQCFSL	DSECT
MQCFSL_TYPE	DS F Structure type
MQCFSL_STRUCLENGTH	DS F Structure length
MQCFSL_PARAMETER	DS F Parameter identifier
MQCFSL_CODEDCCHARSETID	DS F Coded character set identifier
*	
MQCFSL_COUNT	DS F Count of parameter values
MQCFSL_STRINGLENGTH	DS F Length of one string
MQCFSL_LENGTH	EQU *-MQCFSL Length of structure
MQCFSL_ORG	ORG MQCFSL
MQCFSL_AREA	DS CL(MQCFSL_LENGTH)

Visual Basic language declaration (Windows only)

```
Type MQCFSL  
    Type As Long          ' Structure type  
    StrucLength As Long   ' Structure length  
    Parameter As Long     ' Parameter identifier  
    CodedCharSetId As Long ' Coded character set identifier  
    Count As Long         ' Count of parameter values  
    StringLength As Long   ' Length of one string  
End Type
```

Global MQCFSL_DEFAULT As MQCFSL

RPG language declaration (iSeries only)

```
D* MQCFSL Structure  
D*  
D* Structure type  
D SLTYP           1    4I 0 INZ(6)  
D* Structure length  
D SLLEN           5    8I 0 INZ(24)  
D* Parameter identifier  
D SLPRM           9    12I 0 INZ(0)  
D* Coded character set identifier  
D SLCSI          13    16I 0 INZ(0)  
D* Count of parameter values  
D SLCNT          17    20I 0 INZ(0)  
D* Length of one string  
D SLSTL          21    24I 0 INZ(0)
```

MQCFST - PCF string parameter

The MQCFST structure describes a string parameter in a message that is a command or a response to a command. In either case, the format name in the message descriptor is MQFMT_ADMIN.

The MQCFST structure can also be used for user-defined message data. In this case the message descriptor *Format* field is MQFMT_PCF (see “Message descriptor for a PCF command” on page 11). Also in this case, not all of the fields in the structure

are meaningful. The supplied initial values can be used for most fields, but the application must set the *StrucLength*, *StringLength*, and *String* fields to the values appropriate to the data.

The structure ends with a variable-length character string; see the *String* field below for further details.

See “Usage notes” on page 418 for further information on how to use the structure.

Fields

Type (MQLONG)

Structure type.

This indicates that the structure is an MQCFST structure describing a string parameter. The value must be:

MQCFT_STRING

Structure defining a string.

StrucLength (MQLONG)

Structure length.

This is the length in bytes of the MQCFST structure, including the string at the end of the structure (the *String* field). The length must be a multiple of four, and must be sufficient to contain the string; any bytes between the end of the string and the length defined by the *StrucLength* field are not significant.

The following constant gives the length of the *fixed* part of the structure, that is the length excluding the *String* field:

MQCFST_STRUC_LENGTH_FIXED

Length of fixed part of command format string-parameter structure.

Parameter (MQLONG)

Parameter identifier.

This identifies the parameter whose value is contained in the structure. The values that can occur in this field depend on the value of the *Command* field in the MQCFH structure; see “MQCFH - PCF header” on page 418 for details.

The parameter is from the following groups of parameters:

- MQCA_*
- MQCACF_*
- MQCAMO_*
- MQCACH_*

CodedCharSetId (MQLONG)

Coded character set identifier.

This specifies the coded character set identifier of the data in the *String* field. The following special value can be used:

MQCCSI_DEFAULT

Default character set identifier.

The string data is in the character set defined by the *CodedCharSetId* field in the MQ header structure that precedes the MQCFH structure, or by the *CodedCharSetId* field in the MQMD if the MQCFH structure is at the start of the message.

StringLength (MQLONG)

Length of string.

This is the length in bytes of the data in the *String* field; it must be zero or greater. This length need not be a multiple of four.

String (MQCHAR*StringLength*)

String value.

This is the value of the parameter identified by the *Parameter* field:

- In MQFMT_ADMIN command messages, if the specified string is shorter than the standard length of the parameter, the omitted characters are assumed to be blanks. If the specified string is longer than the standard length, it is an error.
- In MQFMT_ADMIN response messages, string parameters may be returned padded with blanks to the standard length of the parameter.
- In MQFMT_EVENT messages, trailing blanks may be omitted from string parameters (that is, the string may be shorter than the standard length of the parameter).

The value of *StringLength* depends on whether, when the specified string is shorter than the standard length, padding blanks have been added to the string. If this is the case, the value of *StringLength* is the sum of the actual length of the string plus the padded blanks.

The string can contain any characters that are in the character set defined by *CodedCharSetId*, and that are valid for the parameter identified by *Parameter*.

Note: When the queue manager reads an MQCFST structure in an MQFMT_ADMIN message from the command input queue, the queue manager processes the string as though it had been specified on an MQI call. This means that within the string, the first null and the characters following it (up to the end of the string) are treated as blanks.

In responses and all other cases, a null character in the string is treated as normal data, and does not act as a delimiter for the string. This means that when a receiving application reads a MQFMT_PCF, MQFMT_EVENT, or MQFMT_ADMIN message, the receiving application receives all of the data specified by the sending application.

The way that this field is declared depends on the programming language:

- For the C programming language, the field is declared as an array with one element. Storage for the structure must be allocated dynamically, and pointers used to address the fields within it.
- For the COBOL, PL/I, and System/390 assembler programming languages, the field is omitted from the structure declaration. When an instance of the structure is declared, the user must include MQCFST in a larger structure, and declare additional field(s) following MQCFST, to represent the *String* field as required.

Language declarations

This structure is available in the following languages:

C language declaration

```
typedef struct tagMQCFST {  
    MQLONG Type;           /* Structure type */  
    MQLONG StrucLength;    /* Structure length */  
    MQLONG Parameter;      /* Parameter identifier */  
    MQLONG CodedCharSetId; /* Coded character set identifier */  
    MQLONG StringLength;   /* Length of string */  
    MQCHAR String[1];      /* String value - first  
                           character */  
} MQCFST;
```

COBOL language declaration

```
**  MQCFST structure  
10 MQCFST.  
**    Structure type  
15 MQCFST-TYPE          PIC S9(9) BINARY.  
**    Structure length  
15 MQCFST-STRUCLENGTH  PIC S9(9) BINARY.  
**    Parameter identifier  
15 MQCFST-PARAMETER    PIC S9(9) BINARY.  
**    Coded character set identifier  
15 MQCFST-CODEDCHARSETID PIC S9(9) BINARY.  
**    Length of string  
15 MQCFST-STRINGLENGTH  PIC S9(9) BINARY.
```

PL/I language declaration (z/OS only)

```
dcl  
1 MQCFST based,  
 3 Type      fixed bin(31), /* Structure type */  
 3 StrucLength fixed bin(31), /* Structure length */  
 3 Parameter  fixed bin(31), /* Parameter identifier */  
 3 CodedCharSetId fixed bin(31), /* Coded character set identifier */  
 3 StringLength fixed bin(31); /* Length of string */
```

System/390 assembler-language declaration (z/OS only)

MQCFST	DSECT
MQCFST_TYPE	DS F Structure type
MQCFST_STRUCLENGTH	DS F Structure length
MQCFST_PARAMETER	DS F Parameter identifier
MQCFST_CODEDCHARSETID	DS F Coded character set identifier
*	
MQCFST_STRINGLENGTH	DS F Length of string
MQCFST_LENGTH	EQU *-MQCFST Length of structure
	ORG MQCFST
MQCFST_AREA	DS CL(MQCFST_LENGTH)

Visual Basic language declaration (Windows only)

```
Type MQCFST  
    Type As Long          ' Structure type  
    StrucLength As Long   ' Structure length  
    Parameter As Long     ' Parameter identifier  
    CodedCharSetId As Long ' Coded character set identifier  
    StringLength As Long   ' Length of string  
End Type
```

```
Global MQCFST_DEFAULT As MQCFST
```

RPG language declaration (iSeries only)

```
D* MQCFST Structure  
D*  
D* Structure type  
D STTYP             1    4I 0 INZ(4)  
D* Structure length  
D STLEN             5    8I 0 INZ(20)
```

```
D* Parameter identifier
D  STPRM          9      12I 0 INZ(0)
D* Coded character set identifier
D  STCSI          13      16I 0 INZ(0)
D* Length of string
D  STSTL          17      20I 0 INZ(0)
D*
```

Chapter 5. PCF example

This is an example of how Programmable Command Formats can be used in a program for administration of WebSphere MQ queues.

Inquire local queue attributes

A C language program is listed here that uses WebSphere MQ for Windows, V6.0. It is given as an example of using PCFs and has been limited to a simple case. This program will be of most use as an example if you are considering the use of PCFs to manage your WebSphere MQ environment.

The program, once compiled, will inquire of the default queue manager about a subset of the attributes for all local queues defined to it. It then produces an output file, SAVEQMGR.TST, in the directory from which it was run. This file is of a format suitable for use with RUNMQSC.

Program listing

```
/*=====
/*
/* This is a program to inquire of the default queue manager about the */
/* local queues defined to it. */
/*
/* The program takes this information and appends it to a file */
/* SAVEQMGR.TST which is of a format suitable for RUNMQSC. It could, */
/* therefore, be used to recreate or clone a queue manager. */
/*
/* It is offered as an example of using Programmable Command Formats (PCFs) */
/* as a method for administering a queue manager.
/*
=====*/
/* Include standard libraries */
#include <memory.h>
#include <stdio.h>

/* Include MQSeries headers */
#include <cmqc.h>
#include <cmqcf.h>
#include <cmqxc.h>

typedef struct LocalQParms {
    MQCHAR48    QName;
    MQLONG      QType;
    MQCHAR64    QDesc;
    MQLONG      InhibitPut;
    MQLONG      DefPriority;
    MQLONG      DefPersistence;
    MQLONG      InhibitGet;
    MQCHAR48    ProcessName;
    MQLONG      MaxQDepth;
    MQLONG      MaxMsgLength;
    MQLONG      BackoutThreshold;
    MQCHAR48    BackoutReqQName;
```

```

        MQLONG      Shareability;
        MQLONG      DefInputOpenOption;
        MQLONG      HardenGetBackout;
        MQLONG      MsgDeliverySequence;
        MQLONG      RetentionInterval;
        MQLONG      DefinitionType;
        MQLONG      Usage;
        MQLONG      OpenInputCount;
        MQLONG      OpenOutputCount;
        MQLONG      CurrentQDepth;
        MQCHAR12     CreationDate;
        MQCHAR8      CreationTime;
        MQCHAR48     InitiationQName;
        MQLONG      TriggerControl;
        MQLONG      TriggerType;
        MQLONG      TriggerMsgPriority;
        MQLONG      TriggerDepth;
        MQCHAR64     TriggerData;
        MQLONG      Scope;
        MQLONG      QDepthHighLimit;
        MQLONG      QDepthLowLimit;
        MQLONG      QDepthMaxEvent;
        MQLONG      QDepthHighEvent;
        MQLONG      QDepthLowEvent;
        MQLONG      QServiceInterval;
        MQLONG      QServiceIntervalEvent;
    } LocalQParms;

void ProcessStringParm( MQCFST *pPCFString, LocalQParms *DefnLQ );

void ProcessIntegerParm( MQCFIN *pPCFInteger, LocalQParms *DefnLQ );

int AddToFileQLOCAL( LocalQParms DefnLQ );

void MQParmCpy( char *target, char *source, int length );

void PutMsg( MQHCONN   hConn          /* Connection to queue manager      */
            , MQCHAR8   MsgFormat      /* Format of user data to be put in msg   */
            , MQHOBJ    hQName         /* handle of queue to put the message to   */
            , MQCHAR48   QName          /* name of queue to put the message to   */
            , MQBYTE    *UserMsg        /* The user data to be put in the message */
            , MQLONG    UserMsgLen     /*                           */ );
/* */

void GetMsg( MQHCONN   hConn          /* handle of queue manager      */
            , MQLONG    MQParm         /* Options to specify nature of get   */
            , MQHOBJ    hQName         /* handle of queue to read from   */
            , MQCHAR48   QName          /* name of queue to read from   */
            , MQBYTE    *UserMsg        /* Input/Output buffer containing msg */
            , MQLONG    ReadBufferLen  /* Length of supplied buffer   */
            );
/* */

MQHOBJ OpenQ( MQHCONN   hConn
            , MQCHAR48   QName
            , MQLONG    OpenOpts
            );
/* */

int main( int argc, char *argv[] )
{
    MQCHAR48           QMgrName;      /* Name of connected queue mgr      */
    MQHCONN            hConn;         /* handle to connected queue mgr   */

```

```

MQOD          ObjDesc;      /* */
MQLONG        OpenOpts;     /* */
MQLONG        CompCode;    /* MQ API completion code */
MQLONG        Reason;      /* Reason qualifying above */
                         /* */
MQHOBJ       hAdminQ;      /* handle to output queue */
MQHOBJ       hReplyQ;     /* handle to input queue */
                         /* */
MQLONG        AdminMsgLen; /* Length of user message buffer */
MQBYTE       *pAdminMsg;   /* Ptr to outbound data buffer */
MQCFH        *pPCFHeader; /* Ptr to PCF header structure */
MQCFST       *pPCFString; /* Ptr to PCF string parm block */
MQCFIN       *pPCFInteger; /* Ptr to PCF integer parm block */
MQLONG        *pPCFType;   /* Type field of PCF message parm */
LocalQParms  DefnLQ;      /* */
                         /* */
char          ErrorReport[40]; /* */
MQCHAR8      MsgFormat;   /* Format of inbound message */
short         Index;       /* Loop counter */
                         /* */

/* Connect to default queue manager */
memset( QMgrName, '\0', sizeof( QMgrName ) );
MQCONN( QMgrName           /* I : use default queue manager */
       , &hConn            /* O : queue manager handle */
       , &CompCode          /* O : Completion code */
       , &Reason            /* O : Reason qualifying CompCode */
       );

if ( CompCode != MQCC_OK ) {
    printf( "MQCONN failed for %s, CC=%d RC=%d\n"
           , QMgrName
           , CompCode
           , Reason
           );
    exit( -1 );
} /* endif */

/* Open all the required queues */
hAdminQ = OpenQ( hConn, "SYSTEM.ADMIN.COMMAND.QUEUE\0", MQOO_OUTPUT );
hReplyQ = OpenQ( hConn, "SAVEQMGR.REPLY.QUEUE\0", MQOO_INPUT_EXCLUSIVE );

/* **** */
/* Put a message to the SYSTEM.ADMIN.COMMAND.QUEUE to inquire all */
/* the local queues defined on the queue manager. */
/* */
/* The request consists of a Request Header and a parameter block */
/* used to specify the generic search. The header and the parameter */
/* block follow each other in a contiguous buffer which is pointed */
/* to by the variable pAdminMsg. This entire buffer is then put to */
/* the queue. */
/* */
/* The command server, (use STRMQCSV to start it), processes the */
/* SYSTEM.ADMIN.COMMAND.QUEUE and puts a reply on the application */
/* ReplyToQ for each defined queue. */
/* **** */

/* Set the length for the message buffer */
AdminMsgLen = MQCFH_STRUC_LENGTH
              + MQCFST_STRUC_LENGTH_FIXED + MQ_Q_NAME_LENGTH
              + MQCFIN_STRUC_LENGTH
              ;

```

```

/* -----
/* Set pointers to message data buffers
/*
/* pAdminMsg points to the start of the message buffer
/*
/* pPCFHeader also points to the start of the message buffer. It is
/* used to indicate the type of command we wish to execute and the
/* number of parameter blocks following in the message buffer.
/*
/* pPCFString points into the message buffer immediately after the
/* header and is used to map the following bytes onto a PCF string
/* parameter block. In this case the string is used to indicate the
/* nameof the queue we want details about, * indicating all queues.
/*
/* pPCFInteger points into the message buffer immediately after the
/* string block described above. It is used to map the following
/* bytes onto a PCF integer parameter block. This block indicates
/* the type of queue we wish to receive details about, thereby
/* qualifying the generic search set up by passing the previous
/* string parameter.
/*
/* Note that this example is a generic search for all attributes of
/* all local queues known to the queue manager. By using different,
/* or more, parameter blocks in the request header it is possible
/* to narrow the search.
/* -----
*/
pAdminMsg = (MQBYTE *)malloc( AdminMsgLen );
pPCFHeader = (MQCFH *)pAdminMsg;
pPCFString = (MQCFST *)(pAdminMsg
+ MQCFH_STRUC_LENGTH
);
pPCFInteger = (MQCFIN *)(
pAdminMsg
+ MQCFH_STRUC_LENGTH
+ MQCFST_STRUC_LENGTH_FIXED + MQ_Q_NAME_LENGTH
);

/* Setup request header */
pPCFHeader->Type = MQCFT_COMMAND;
pPCFHeader->StrucLength = MQCFH_STRUC_LENGTH;
pPCFHeader->Version = MQCFH_VERSION_1;
pPCFHeader->Command = MQCMD_INQUIRE_Q;
pPCFHeader->MsgSeqNumber = MQCFC_LAST;
pPCFHeader->Control = MQCFC_LAST;
pPCFHeader->ParameterCount = 2;

/* Setup parameter block */
pPCFString->Type = MQCFT_STRING;
pPCFString->StrucLength = MQCFST_STRUC_LENGTH_FIXED + MQ_Q_NAME_LENGTH;
pPCFString->Parameter = MQCA_Q_NAME;
pPCFString->CodedCharSetId = MQCCSI_DEFAULT;
pPCFString->StringLength = MQ_Q_NAME_LENGTH;
memset( pPCFString->String, ' ', MQ_Q_NAME_LENGTH );
memcpy( pPCFString->String, "*", 1 );

/* Setup parameter block */
pPCFInteger->Type = MQCFT_INTEGER;
pPCFInteger->StrucLength = MQCFIN_STRUC_LENGTH;
pPCFInteger->Parameter = MQIA_Q_TYPE;
pPCFInteger->Value = MQQT_LOCAL;

PutMsg( hConn /* Queue manager handle */ )

```

```

        , MQFMT_ADMIN           /* Format of message          */
        , hAdminQ                /* Handle of command queue      */
        , "SYSTEM.ADMIN.COMMAND.QUEUE\0"
        , (MQBYTE *)pAdminMsg    /* Data part of message to put   */
        , AdminMsgLen            /*                                */
    );

free( pAdminMsg );

/*
 * ****
 * Get and process the replies received from the command server onto      */
 * the applications ReplyToQ.                                              */
 */
/* There will be one message per defined local queue.                      */
*/
/* The last message will have the Control field of the PCF header       */
/* set to MQCFC_LAST. All others will be MQCFC_NOT_LAST.                  */
*/
/* An individual Reply message consists of a header followed by a         */
/* number a parameters, the exact number, type and order will depend     */
/* upon the type of request.                                               */
*/
/* ----- */
/*
/* The message is retrieved into a buffer pointed to by pAdminMsg.        */
/* This buffer as been allocated to be large enough to hold all the      */
/* parameters for a local queue definition.                               */
*/
/* pPCFHeader is then allocated to point also to the beginning of        */
/* the buffer and is used to access the PCF header structure. The        */
/* header contains several fields. The one we are specifically             */
/* interested in is the ParameterCount. This tells us how many           */
/* parameters follow the header in the message buffer. There is          */
/* one parameter for each local queue attribute known by the            */
/* queue manager.                                                       */
*/
/* At this point we do not know the order or type of each parameter      */
/* block in the buffer, the first MQLONG of each block defines its        */
/* type; they may be parameter blocks containing either strings or       */
/* integers.                                                               */
*/
/* pPCFType is used initially to point to the first byte beyond the      */
/* known parameter block. Initially then, it points to the first byte      */
/* after the PCF header. Subsequently it is incremented by the length       */
/* of the identified parameter block and therefore points at the          */
/* next. Looking at the value of the data pointed to by pPCFType we       */
/* can decide how to process the next group of bytes, either as a          */
/* string, or an integer.                                                 */
*/
/* In this way we parse the message buffer extracting the values of       */
/* each of the parameters we are interested in.                            */
*/
/* ****
*/
/* AdminMsgLen is to be set to the length of the expected reply          */
/* message. This structure is specific to Local Queues.                   */
AdminMsgLen = MQCFH_STRUC_LENGTH
            + (MQCFST_STRUC_LENGTH_FIXED * 12)
            + (MQCFIN_STRUC_LENGTH * 30)
            + MQ_Q_NAME_LENGTH
            + MQ_Q_DESC_LENGTH
            + MQ_PROCESS_NAME_LENGTH
            + MQ_Q_NAME_LENGTH
            + MQ_CREATION_DATE_LENGTH
            + MQ_CREATION_TIME_LENGTH
            + MQ_Q_NAME_LENGTH

```

```

        + MQ_TRIGGER_DATA_LENGTH
        + MQ_Q_NAME_LENGTH
        + MQ_Q_NAME_LENGTH
        + MQ_Q_MGR_NAME_LENGTH
        + MQ_Q_NAME_LENGTH
        ;

/* Set pointers to message data buffers */
pAdminMsg = (MQBYTE *)malloc( AdminMsgLen );

do {

    GetMsg( hConn                  /* Queue manager handle */
            , MQGMO_WAIT           /* Parameters on Get */
            , hReplyQ               /* Get queue handle */
            , "SAVEQMGR.REPLY.QUEUE\0"
            , (MQBYTE *)pAdminMsg   /* pointer to message area */
            , AdminMsgLen           /* length of get buffer */
            );

    /* Examine Header */
    pPCFHeader = (MQCFH *)pAdminMsg;

    /* Examine first parameter */
    pPCFType = (MLONG *)(pAdminMsg + MQCFH_STRUC_LENGTH);

    Index = 1;

    while ( Index <= pPCFHeader->ParameterCount ) {

        /* Establish the type of each parameter and allocate */
        /* a pointer of the correct type to reference it. */
        switch ( *pPCFType ) {
        case MQCFT_INTEGER:
            pPCFInteger = (MQCFIN *)pPCFType;
            ProcessIntegerParm( pPCFInteger, &DefnLQ );
            Index++;
            /* Increment the pointer to the next parameter by the */
            /* length of the current parm. */
            pPCFType = (MLONG *)((MQBYTE *)pPCFType
                                + pPCFInteger->StrucLength
                                );
            break;
        case MQCFT_STRING:
            pPCFString = (MQCFST *)pPCFType;
            ProcessStringParm( pPCFString, &DefnLQ );
            Index++;
            /* Increment the pointer to the next parameter by the */
            /* length of the current parm. */
            pPCFType = (MLONG *)((MQBYTE *)pPCFType
                                + pPCFString->StrucLength
                                );
            break;
        } /* endswitch */
    } /* endwhile */

    /* **** */
    /* Message parsed, append to output file */
    /* **** */
    AddToFileQLocal( DefnLQ );

    /* **** */
    /* Finished processing the current message, do the next one. */
    /* **** */

```

```

} while ( pPCFHeader->Control == MQFCF_NOT_LAST ); /* enddo */

free( pAdminMsg );

/* **** */
/* Processing of the local queues complete */
/* **** */

}

void ProcessStringParm( MQCFST *pPCFString, LocalQParms *DefnLQ )
{
    switch ( pPCFString->Parameter ) {
    case MQCA_Q_NAME:
        MQParmCpy( DefnLQ->QName, pPCFString->String, 48 );
        break;
    case MQCA_Q_DESC:
        MQParmCpy( DefnLQ->QDesc, pPCFString->String, 64 );
        break;
    case MQCA_PROCESS_NAME:
        MQParmCpy( DefnLQ->ProcessName, pPCFString->String, 48 );
        break;
    case MQCA_BACKOUT_REQ_Q_NAME:
        MQParmCpy( DefnLQ->BackoutReqQName, pPCFString->String, 48 );
        break;
    case MQCA_CREATION_DATE:
        MQParmCpy( DefnLQ->CreationDate, pPCFString->String, 12 );
        break;
    case MQCA_CREATION_TIME:
        MQParmCpy( DefnLQ->CreationTime, pPCFString->String, 8 );
        break;
    case MQCA_INITIATION_Q_NAME:
        MQParmCpy( DefnLQ->InitiationQName, pPCFString->String, 48 );
        break;
    case MQCA_TRIGGER_DATA:
        MQParmCpy( DefnLQ->TriggerData, pPCFString->String, 64 );
        break;
    } /* endswitch */
}

void ProcessIntegerParm( MQCFIN *pPCFInteger, LocalQParms *DefnLQ )
{
    switch ( pPCFInteger->Parameter ) {
    case MQIA_Q_TYPE:
        DefnLQ->QType = pPCFInteger->Value;
        break;
    case MQIA_INHIBIT_PUT:
        DefnLQ->InhibitPut = pPCFInteger->Value;
        break;
    case MQIA_DEF_PRIORITY:
        DefnLQ->DefPriority = pPCFInteger->Value;
        break;
    case MQIA_DEF_PERSISTENCE:
        DefnLQ->DefPersistence = pPCFInteger->Value;
        break;
    case MQIA_INHIBIT_GET:
        DefnLQ->InhibitGet = pPCFInteger->Value;
        break;
    case MQIA_SCOPE:
        DefnLQ->Scope = pPCFInteger->Value;
        break;
    case MQIA_MAX_Q_DEPTH:
        DefnLQ->MaxQDepth = pPCFInteger->Value;
        break;
    }
}

```

```

case MQIA_MAX_MSG_LENGTH:
    DefnLQ->MaxMsgLength = pPCFInteger->Value;
    break;
case MQIA_BACKOUT_THRESHOLD:
    DefnLQ->BackoutThreshold = pPCFInteger->Value;
    break;
case MQIA_SHAREABILITY:
    DefnLQ->Shareability = pPCFInteger->Value;
    break;
case MQIA_DEF_INPUT_OPEN_OPTION:
    DefnLQ->DefInputOpenOption = pPCFInteger->Value;
    break;
case MQIA_HARDEN_GET_BACKOUT:
    DefnLQ->HardenGetBackout = pPCFInteger->Value;
    break;
case MQIA_MSG_DELIVERY_SEQUENCE:
    DefnLQ->MsgDeliverySequence = pPCFInteger->Value;
    break;
case MQIA_RETENTION_INTERVAL:
    DefnLQ->RetentionInterval = pPCFInteger->Value;
    break;
case MQIA_DEFINITION_TYPE:
    DefnLQ->DefinitionType = pPCFInteger->Value;
    break;
case MQIA_USAGE:
    DefnLQ->Usage = pPCFInteger->Value;
    break;
case MQIA_OPEN_INPUT_COUNT:
    DefnLQ->OpenInputCount = pPCFInteger->Value;
    break;
case MQIA_OPEN_OUTPUT_COUNT:
    DefnLQ->OpenOutputCount = pPCFInteger->Value;
    break;
case MQIA_CURRENT_Q_DEPTH:
    DefnLQ->CurrentQDepth = pPCFInteger->Value;
    break;
case MQIA_TRIGGER_CONTROL:
    DefnLQ->TriggerControl = pPCFInteger->Value;
    break;
case MQIA_TRIGGER_TYPE:
    DefnLQ->TriggerType = pPCFInteger->Value;
    break;
case MQIA_TRIGGER_MSG_PRIORITY:
    DefnLQ->TriggerMsgPriority = pPCFInteger->Value;
    break;
case MQIA_TRIGGER_DEPTH:
    DefnLQ->TriggerDepth = pPCFInteger->Value;
    break;
case MQIA_Q_DEPTH_HIGH_LIMIT:
    DefnLQ->QDepthHighLimit = pPCFInteger->Value;
    break;
case MQIA_Q_DEPTH_LOW_LIMIT:
    DefnLQ->QDepthLowLimit = pPCFInteger->Value;
    break;
case MQIA_Q_DEPTH_MAX_EVENT:
    DefnLQ->QDepthMaxEvent = pPCFInteger->Value;
    break;
case MQIA_Q_DEPTH_HIGH_EVENT:
    DefnLQ->QDepthHighEvent = pPCFInteger->Value;
    break;
case MQIA_Q_DEPTH_LOW_EVENT:
    DefnLQ->QDepthLowEvent = pPCFInteger->Value;
    break;
case MQIA_Q_SERVICE_INTERVAL:
    DefnLQ->QServiceInterval = pPCFInteger->Value;
    break;
case MQIA_Q_SERVICE_INTERVAL_EVENT:

```

```

        DefnLQ->QServiceIntervalEvent = pPCFInteger->Value;
        break;
    } /* endswitch */
}

/*
 * -----
 * This process takes the attributes of a single local queue and adds them
 * to the end of a file, SAVEQMGR.TST, which can be found in the current
 * directory.
 *
 * The file is of a format suitable for subsequent input to RUNMQSC.
 *
 * -----
int AddToFileQLocal( LocalQParms DefnLQ )
{
    char    ParmBuffer[120]; /* Temporary buffer to hold for output to file */
    FILE   *fp;             /* Pointer to a file */

    /* Append these details to the end of the current SAVEQMGR.TST file */
    fp = fopen( "SAVEQMGR.TST", "a" );

    sprintf( ParmBuffer, "DEFINE QLOCAL ('%s') REPLACE +\n", DefnLQ.QName );
    fputs( ParmBuffer, fp );

    sprintf( ParmBuffer, "      DESCRIPTOR(%s) +\n" , DefnLQ.QDesc );
    fputs( ParmBuffer, fp );

    if ( DefnLQ.InhibitPut == MQQA_PUT_ALLOWED ) {
        sprintf( ParmBuffer, "      PUT(ENABLED) +\n" );
        fputs( ParmBuffer, fp );
    } else {
        sprintf( ParmBuffer, "      PUT(DISABLED) +\n" );
        fputs( ParmBuffer, fp );
    } /* endif */

    sprintf( ParmBuffer, "      DEFPRTY(%d) +\n", DefnLQ.DefPriority );
    fputs( ParmBuffer, fp );

    if ( DefnLQ.DefPersistence == MQPER_PERSISTENT ) {
        sprintf( ParmBuffer, "      DEFPSIST(YES) +\n" );
        fputs( ParmBuffer, fp );
    } else {
        sprintf( ParmBuffer, "      DEFPSIST(NO) +\n" );
        fputs( ParmBuffer, fp );
    } /* endif */

    if ( DefnLQ.InhibitGet == MQQA_GET_ALLOWED ) {
        sprintf( ParmBuffer, "      GET(ENABLED) +\n" );
        fputs( ParmBuffer, fp );
    } else {
        sprintf( ParmBuffer, "      GET(DISABLED) +\n" );
        fputs( ParmBuffer, fp );
    } /* endif */

    sprintf( ParmBuffer, "      MAXDEPTH(%d) +\n", DefnLQ.MaxQDepth );
    fputs( ParmBuffer, fp );

    sprintf( ParmBuffer, "      MAXMSGLEN(%d) +\n", DefnLQ.MaxMsgLength );
    fputs( ParmBuffer, fp );

    if ( DefnLQ.Shareability == MQQA_SHAREABLE ) {
        sprintf( ParmBuffer, "      SHARE +\n" );
        fputs( ParmBuffer, fp );
    } else {
        sprintf( ParmBuffer, "      NOSHARE +\n" );

```

```

        fputs( ParmBuffer, fp );
    } /* endif */

    if ( DefnLQ.DefInputOpenOption == MQOO_INPUT_SHARED ) {
        sprintf( ParmBuffer, "      DEFSOPT(SHARED) +\n" );
        fputs( ParmBuffer, fp );
    } else {
        sprintf( ParmBuffer, "      DEFSOPT(EXCL) +\n" );
        fputs( ParmBuffer, fp );
    } /* endif */

    if ( DefnLQ.MsgDeliverySequence == MQMDS_PRIORITY ) {
        sprintf( ParmBuffer, "      MSGDLVSQ(PRIORITY) +\n" );
        fputs( ParmBuffer, fp );
    } else {
        sprintf( ParmBuffer, "      MSGDLVSQ(FIFO) +\n" );
        fputs( ParmBuffer, fp );
    } /* endif */

    if ( DefnLQ.HardenGetBackout == MQQA_BACKOUT_HARDENED ) {
        sprintf( ParmBuffer, "      HARDENBO +\n" );
        fputs( ParmBuffer, fp );
    } else {
        sprintf( ParmBuffer, "      NOHARDENBO +\n" );
        fputs( ParmBuffer, fp );
    } /* endif */

    if ( DefnLQ.Usage == MQUS_NORMAL ) {
        sprintf( ParmBuffer, "      USAGE(NORMAL) +\n" );
        fputs( ParmBuffer, fp );
    } else {
        sprintf( ParmBuffer, "      USAGE(XMIT) +\n" );
        fputs( ParmBuffer, fp );
    } /* endif */

    if ( DefnLQ.TriggerControl == MQTC_OFF ) {
        sprintf( ParmBuffer, "      NOTRIGGER +\n" );
        fputs( ParmBuffer, fp );
    } else {
        sprintf( ParmBuffer, "      TRIGGER +\n" );
        fputs( ParmBuffer, fp );
    } /* endif */

    switch ( DefnLQ.TriggerType ) {
    case MQTT_NONE:
        sprintf( ParmBuffer, "      TRIGTYPE(NONE) +\n" );
        fputs( ParmBuffer, fp );
        break;
    case MQTT_FIRST:
        sprintf( ParmBuffer, "      TRIGTYPE(FIRST) +\n" );
        fputs( ParmBuffer, fp );
        break;
    case MQTT_EVERY:
        sprintf( ParmBuffer, "      TRIGTYPE(EVERY) +\n" );
        fputs( ParmBuffer, fp );
        break;
    case MQTT_DEPTH:
        sprintf( ParmBuffer, "      TRIGTYPE(DEPTH) +\n" );
        fputs( ParmBuffer, fp );
        break;
    } /* endswitch */

    sprintf( ParmBuffer, "      TRIGDPTH(%d) +\n", DefnLQ.TriggerDepth );
    fputs( ParmBuffer, fp );

    sprintf( ParmBuffer, "      TRIGMPRI(%d) +\n", DefnLQ.TriggerMsgPriority);
    fputs( ParmBuffer, fp );

```

```

sprintf( ParmBuffer, "      TRIGDATA('%s') +\n", DefnLQ.TriggerData );
fputs( ParmBuffer, fp );

sprintf( ParmBuffer, "      PROCESS('%s') +\n", DefnLQ.ProcessName );
fputs( ParmBuffer, fp );

sprintf( ParmBuffer, "      INITQ('%s') +\n", DefnLQ.InitiationQName );
fputs( ParmBuffer, fp );

sprintf( ParmBuffer, "      RETINTVL(%d) +\n", DefnLQ.RetentionInterval );
fputs( ParmBuffer, fp );

sprintf( ParmBuffer, "      BOTHRESH(%d) +\n", DefnLQ.BackoutThreshold );
fputs( ParmBuffer, fp );

sprintf( ParmBuffer, "      BOQNAME('%s') +\n", DefnLQ.BackoutReqQName );
fputs( ParmBuffer, fp );

if ( DefnLQ.Scope == MQSCO_Q_MGR ) {
    sprintf( ParmBuffer, "      SCOPE(QMGR) +\n" );
    fputs( ParmBuffer, fp );
} else {
    sprintf( ParmBuffer, "      SCOPE(CELL) +\n" );
    fputs( ParmBuffer, fp );
} /* endif */

sprintf( ParmBuffer, "      QDEPTHHI(%d) +\n", DefnLQ.QDepthHighLimit );
fputs( ParmBuffer, fp );

sprintf( ParmBuffer, "      QDEPTHLO(%d) +\n", DefnLQ.QDepthLowLimit );
fputs( ParmBuffer, fp );

if ( DefnLQ.QDepthMaxEvent == MQEVR_ENABLED ) {
    sprintf( ParmBuffer, "      QDPMAXEV(ENABLED) +\n" );
    fputs( ParmBuffer, fp );
} else {
    sprintf( ParmBuffer, "      QDPMAXEV(DISABLED) +\n" );
    fputs( ParmBuffer, fp );
} /* endif */

if ( DefnLQ.QDepthHighEvent == MQEVR_ENABLED ) {
    sprintf( ParmBuffer, "      QDPHIEV(ENABLED) +\n" );
    fputs( ParmBuffer, fp );
} else {
    sprintf( ParmBuffer, "      QDPHIEV(DISABLED) +\n" );
    fputs( ParmBuffer, fp );
} /* endif */

if ( DefnLQ.QDepthLowEvent == MQEVR_ENABLED ) {
    sprintf( ParmBuffer, "      QDPLOEV(ENABLED) +\n" );
    fputs( ParmBuffer, fp );
} else {
    sprintf( ParmBuffer, "      QDPLOEV(DISABLED) +\n" );
    fputs( ParmBuffer, fp );
} /* endif */

sprintf( ParmBuffer, "      QSVCINT(%d) +\n", DefnLQ.QServiceInterval );
fputs( ParmBuffer, fp );

switch ( DefnLQ.QServiceIntervalEvent ) {
case MQQSIE_OK:
    sprintf( ParmBuffer, "      QSVCIEV(OK)\n" );
    fputs( ParmBuffer, fp );
    break;
case MQQSIE_NONE:

```

```

        sprintf( ParmBuffer, "      QSVCIEV(NONE)\n" );
        fputs( ParmBuffer, fp );
        break;
    case MQQSIE_HIGH:
        sprintf( ParmBuffer, "      QSVCIEV(HIGH)\n" );
        fputs( ParmBuffer, fp );
        break;
    } /* endswitch */

    sprintf( ParmBuffer, "\n" );
    fputs( ParmBuffer, fp );

    fclose(fp);

}

/*
 */
/* The queue manager returns strings of the maximum length for each */
/* specific parameter, padded with blanks. */
/*
/* We are interested in only the nonblank characters so will extract them */
/* from the message buffer, and terminate the string with a null, \0. */
/*
*/
void MQParmCpy( char *target, char *source, int length )
{
    int    counter=0;

    while ( counter < length && source[counter] != ' ' ) {
        target[counter] = source[counter];
        counter++;
    } /* endwhile */

    if ( counter < length) {
        target[counter] = '\0';
    } /* endif */
}

```

Part 2. Message Queueing Administration Interface

Chapter 6. Introduction to the WebSphere MQ

Administration Interface (MQAI)	463
MQAI concepts and terminology	463
Use of the MQAI	464
How do I use the MQAI?	465
Overview	465
Building your MQAI application	466

Chapter 7. Using data bags

Types of data bag	467
Creating and deleting data bags	467
Deleting data bags	468
Types of data item	468
Adding data items to bags	469
Adding an inquiry command to a bag	469
Filtering and querying data items	469
Changing information within a bag	470
Counting data items	471
Deleting data items	472
Deleting data items from a bag using the mqDeleteItem call	472
Clearing a bag using the mqClearBag call	472
Truncating a bag using the mqTruncateBag call	473
Inquiring within data bags	473
System items	473

Chapter 8. Configuring WebSphere MQ using

mqExecute	475
Sending administration commands to the command server	475
Example code	476
Hints and tips for configuring WebSphere MQ	477

Chapter 9. Exchanging data between applications

Converting bags and buffers	479
Putting and receiving data bags	480
Sending PCF messages to a specified queue	480
Receiving PCF messages from a specified queue	480

Chapter 10. MQAI reference

mqAddBag	483
Syntax	484
Parameters	484
Usage notes	485
C language invocation	485
Visual Basic invocation	485
mqAddByteString	485
Syntax	485
Parameters	486
Usage notes	487
C language invocation	487
Visual Basic invocation	487
mqAddByteStringFilter	487
Syntax	487

Parameters	487
Usage notes	489
C language invocation	489
Visual Basic invocation	489
mqAddInquiry	489
Syntax	489
Parameters	490
Usage notes	490
C language invocation	490
Visual Basic invocation	491
Supported INQUIRE command codes	491
mqAddInteger	491
Syntax	491
Parameters	491
Usage notes	492
C language invocation	492
Visual Basic invocation	493
mqAddInteger64	493
Syntax	493
Parameters	493
Usage notes	494
C language invocation	494
Visual Basic invocation	494
mqAddIntegerFilter	494
Syntax	494
Parameters	495
Usage notes	496
C language invocation	496
Visual Basic invocation	496
mqAddString	496
Syntax	496
Parameters	496
Usage notes	498
C language invocation	498
Visual Basic invocation	498
mqAddStringFilter	498
Syntax	498
Parameters	498
Usage notes	500
C language invocation	500
Visual Basic invocation	500
mqBagToBuffer	500
Syntax	500
Parameters	501
Usage notes	502
C language invocation	502
Visual Basic invocation	502
mqBufferToBag	503
Syntax	503
Parameters	503
Usage notes	504
C language invocation	504
Visual Basic invocation	504
mqClearBag	504
Syntax	504
Parameters	504

Usage notes	505
C language invocation	505
Visual Basic invocation	505
mqCountItems	505
Syntax.	505
Parameters	505
Usage notes	506
C language invocation	506
Visual Basic invocation	506
mqCreateBag	507
Syntax.	507
Parameters	507
Usage notes	510
C language invocation	510
Visual Basic invocation	510
mqDeleteBag	511
Syntax.	511
Parameters	511
Usage notes	511
C language invocation	511
Visual Basic invocation	511
mqDeleteItem	512
Syntax.	512
Parameters	512
Usage notes	513
C language invocation	514
Visual Basic invocation	514
mqExecute	514
Syntax.	514
Parameters	514
Usage notes	517
C language invocation	517
Visual Basic invocation	518
mqGetBag	518
Syntax.	518
Parameters	518
Usage notes	520
C language invocation	520
Visual Basic invocation	520
mqInquireBag	520
Syntax.	520
Parameters	520
C language invocation	522
Visual Basic invocation	522
mqInquireByteString	523
Syntax.	523
Parameters	523
C language invocation	525
Visual Basic invocation	525
mqInquireByteStringFilter	525
Syntax.	525
Parameters	526
C language invocation	528
Visual Basic invocation	528
mqInquireInteger	528
Syntax.	528
Parameters	529
C language invocation	530
Visual Basic invocation	530
mqInquireInteger64	531
Syntax.	531
Parameters	531
C language invocation	532
Visual Basic invocation	533
mqInquireIntegerFilter	533
Syntax.	533
Parameters	533
C language invocation	535
Visual Basic invocation	535
mqInquireItemInfo	535
Syntax.	535
Parameters	535
C language invocation	537
Visual Basic invocation	538
mqInquireString	538
Syntax.	538
Parameters	538
C language invocation	540
Visual Basic invocation	541
mqInquireStringFilter	541
Syntax.	541
Parameters	541
C language invocation	543
Visual Basic invocation	544
mqPad	544
Syntax.	544
Parameters	544
Usage notes	545
C language invocation	545
mqPutBag	545
Syntax.	545
Parameters	545
C language invocation	547
Visual Basic invocation	547
mqSetByteString	547
Syntax.	547
Parameters	547
C language invocation	549
Visual Basic invocation	549
mqSetByteStringFilter	550
Syntax.	550
Parameters	550
C language invocation	552
Visual Basic invocation	552
mqSetInteger	553
Syntax.	553
Parameters	553
C language invocation	555
Visual Basic invocation	555
mqSetInteger64	555
Syntax.	555
Parameters	555
C language invocation	557
Visual Basic invocation	557
mqSetIntegerFilter	557
Syntax.	557
Parameters	558
C language invocation	559
Visual Basic invocation	560
mqSetString	560
Syntax.	560
Parameters	560

Usage notes	562
C language invocation	562
Visual Basic invocation	563
mqSetStringFilter	563
Syntax.	563
Parameters	563
Usage notes	565
C language invocation	565
Visual Basic invocation	566
mqTrim	566
Syntax.	566
Parameters	566
Usage notes	567
C language invocation	567
mqTruncateBag	567
Syntax.	567
Parameters	567
Usage notes	568
C language invocation	568
Visual Basic invocation	568
MQAI Selectors	568
User selectors	569
System selectors	569
Chapter 11. Examples of using the MQAI	571
Creating a local queue (amqsaicq.c)	571
Inquiring about queues and printing information (amqsailq.c)	575
Displaying events using an event monitor (amqsaiem.c)	580
Chapter 12. Advanced topics	589
Indexing	589
Data conversion	590
Use of the message descriptor	591

Chapter 6. Introduction to the WebSphere MQ Administration Interface (MQAI)

This topic describes:

- The main WebSphere MQ Administration Interface (MQAI) concepts and terminology
- When the MQAI can be used
- How to use the MQAI

MQAI concepts and terminology

The MQAI is a programming interface to WebSphere MQ, using the C language and also Visual Basic for Windows. It is available on platforms other than z/OS. It performs administration tasks on a WebSphere MQ queue manager using *data bags*. Data bags allow you to handle properties (or parameters) of objects in a way that is easier than using the other administration interface, Programmable Command Formats (PCFs). The MQAI offers easier manipulation of PCFs than using the MQGET and MQPUT calls. For more information about data bags, see Chapter 7, "Using data bags," on page 467. For more information about PCFs, see part 1 of this book.

The data bag contains zero or more *data items*. These are ordered within the bag as they are placed into the bag. This is called the *insertion order*. Each data item contains a *selector* that identifies the data item and a *value* of that data item that can be either an integer, a 64-bit integer, an integer filter, a string, a string filter, a byte string, a byte string filter, or a handle of another bag.

There are two types of selector; *user selectors* and *system selectors*. These are described in "MQAI Selectors" on page 568. The selectors are usually unique, but it is possible to have multiple values for the same selector. In this case, an *index* identifies the particular occurrence of selector that is required. Indexes are described in "Indexing" on page 589.

A hierarchy of the above concepts is shown in Figure 1 on page 464.

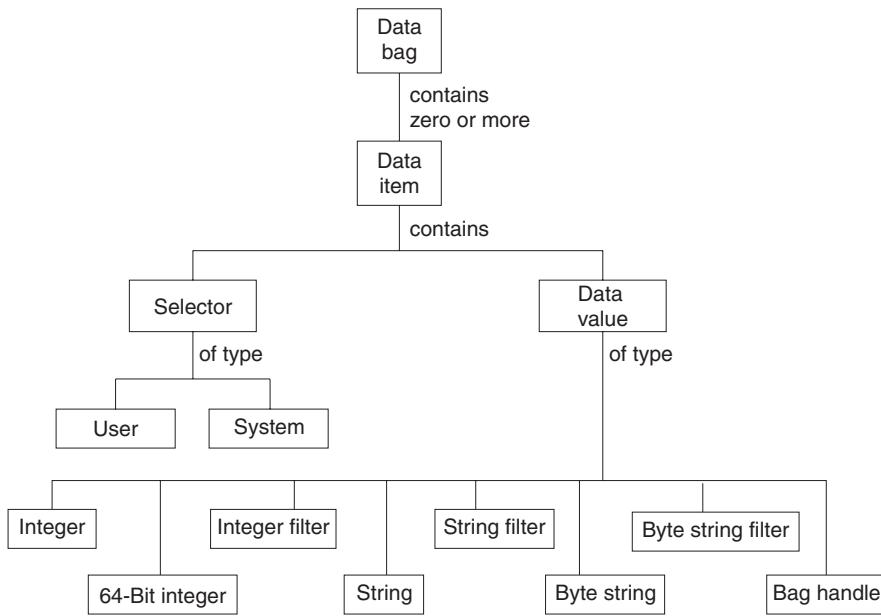


Figure 1. Hierarchy of MQAI concepts

Use of the MQAI

You can use the MQAI to::

- Implement self-administering applications and administration tools. For example, the Active Directory Services provided on Windows uses the MQAI. For more information about the Active Directory Service Interface, see the *WebSphere MQ Using the Component Object Model Interface* book.
- Simplify the use of PCF messages. The MQAI is an easy way to administer WebSphere MQ; you do not have to write your own PCF messages and thus avoid the problems associated with complex data structures.
- Handle error conditions more easily. It is difficult to get return codes back from the WebSphere MQ script (MQSC) commands, but the MQAI makes it easier for the program to handle error conditions.

How do I use the MQAI?

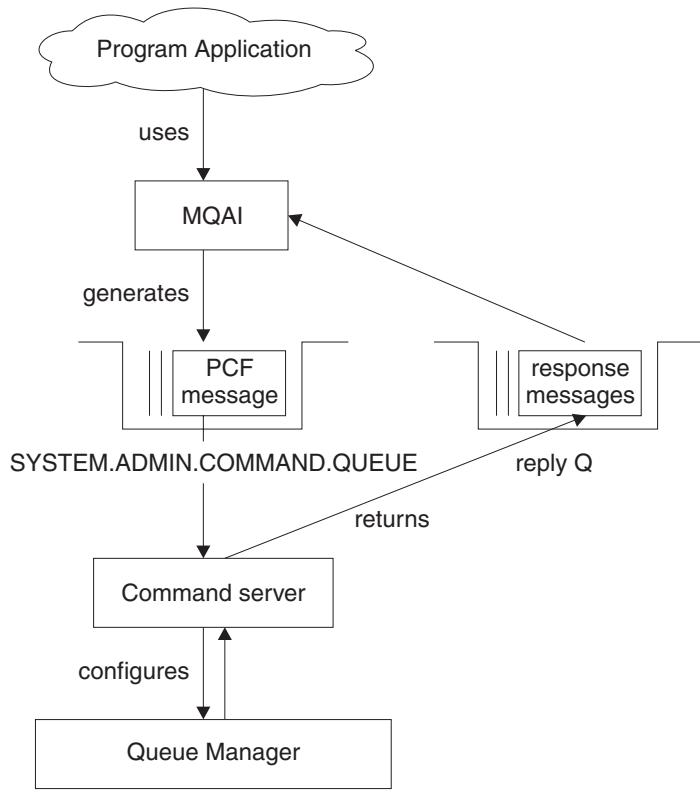


Figure 2. How the MQAI administers WebSphere MQ

The MQAI provides easier programming access to PCF messages. To pass parameters in programs that are written using MQI calls, the PCF message must contain the command and details of the string or integer data. To do this, several statements are needed in your program for every structure, and memory space must be allocated.

On the other hand, programs written using the MQAI pass parameters into the data bag and only one statement is required for each structure. The data bag removes the need for the programmer to handle arrays and allocate storage, and provides some isolation from the details of PCF.

The MQAI administers WebSphere MQ by sending PCF messages to the command server and waiting for a response as shown in Figure 2.

Overview

The following instructions give a brief overview of 1) what you do with the MQAI, and 2) how you use the MQAI. Further details are contained in the rest of this book.

To use the MQAI to administer WebSphere MQ:

1. Decide on the task you want to carry out (for example, Change Queue).

2. Use part 1 of this book as a reference to the commands and responses sent between a WebSphere MQ systems management application program and a WebSphere MQ queue manager. For example, look up the Change, Create and Copy Queues command in this book.
3. Choose the values of the selectors for the required parameters and any optional parameters that you want to set.
4. Create a data bag using the mqCreateBag call and enter values for each of these selectors using the mqAdd* calls. This is described in Chapter 7, “Using data bags,” on page 467.
5. Ensure the command server is running.
6. Using the mqExecute call, send the message to the command server and wait for a response. This is described in Chapter 8, “Configuring WebSphere MQ using mqExecute,” on page 475.

To use the MQAI to exchange data between applications:

- The sender must:
 1. Create a data bag intended to send the data using mqCreateBag. See “Creating and deleting data bags” on page 467.
 2. Add the data to be sent in the bag using mqAddInteger or mqAddString. See “Adding data items to bags” on page 469.
 3. Use the mqPutBag call to convert the data in the bag into a PCF message and put the message onto the required queue. See “Putting and receiving data bags” on page 480.
- The receiver must:
 1. Create a data bag intended to receive the data using mqCreateBag. See “Creating and deleting data bags” on page 467.
 2. Use the mqGetBag call to get the PCF message from the queue and recreate a bag from the PCF message. See “Putting and receiving data bags” on page 480.

Using the MQAI is discussed in more detail in the topics that follow.

Building your MQAI application

To build your application using the MQAI, you link to the same libraries as you do for WebSphere MQ. For information on how to build your WebSphere MQ applications, see the *WebSphere MQ Application Programming Guide*.

Chapter 7. Using data bags

A data bag is a means of handling properties (or parameters) of objects using the MQAI. This topic discusses the configuration of data bags. It describes:

- The different types of bag and their uses
- How to create and delete data bags
- Types of data item
- How to add data items to data bags
- How to change information within a data bag
- How to count data items within a data bag
- How to delete data items
- How to inquire within data bags
- System items

Types of data bag

You can choose the type of data bag that you want to create depending on the task that you wish to perform:

user bag

A simple bag used for user data.

administration bag

A bag created for data used to administer WebSphere MQ objects by sending administration messages to a command server. The administration bag automatically implies certain options as described in “Creating and deleting data bags.”

command bag

A bag also created for commands for administering WebSphere MQ objects. However, unlike the administration bag, the command bag does not automatically imply certain options although these options are available. Again, these options are discussed in “Creating and deleting data bags.”

group bag

A bag used to hold a set of grouped data items. Group bags cannot be used for administering WebSphere MQ objects.

In addition, the **system bag** is created by the MQAI when a reply message is returned from the command server and placed into a user’s output bag. A system bag cannot be modified by the user.

Creating and deleting data bags

To use the MQAI, you first create a data bag using the mqCreateBag call. As input to this call, you supply one or more options to control the creation of the bag.

The *Options* parameter of the MQCreateBag call lets you choose whether to create a user bag, a command bag, a group bag, or an administration bag.

To create a user bag, a command bag, or a group bag, you can choose one or more further options to:

- Use the list form when there are two or more adjacent occurrences of the same selector in a bag.
- Reorder the data items as they are added to a PCF message to ensure that the parameters are in their correct order.
- Check the values of user selectors for items that you add to the bag.

Administration bags automatically imply these options.

A data bag is identified by its handle. The bag handle is returned from `mqCreateBag` and must be supplied on all other calls that use the data bag.

For a full description of the `mqCreateBag` call, see “`mqCreateBag`” on page 507.

Deleting data bags

Any data bag that is created by the user must also be deleted using the `mqDeleteBag` call. For example, if a bag is created in the user code, it must also be deleted in the user code.

System bags are created and deleted automatically by the MQAI. For more information about this, see “[Sending administration commands to the command server](#)” on page 475. User code cannot delete a system bag.

For a full description of the `mqDeleteBag` call, see “`mqDeleteBag`” on page 511.

Types of data item

Here are the types of data item available within the MQAI:

- Integer
- 64-bit integer
- Integer filter
- Character-string
- String filter
- Byte string
- Byte string filter
- Bag handle

When you have created a data bag, you can populate it with integer or character-string items. You can inquire about all three types of item.

Note: You cannot insert bag handles.

These data items can be user or system items. User items contain user data such as attributes of objects that are being administered. System items should be used for more control over the messages generated: for example, the generation of message headers. For more information about system items, see “[System items](#)” on page 473.

Adding data items to bags

The MQAI lets you add integer items, 64-bit integer items, integer filter items, character-string items, string filter, byte string items, and byte string filter items to bags and this is shown in Figure 3. The items are identified by a selector. Usually one selector identifies one item only, but this is not always the case. If a data item with the specified selector is already present in the bag, an additional instance of that selector is added to the end of the bag.



Figure 3. Adding data items

Add data items to a bag using the mqAdd* calls:

- To add integer items, use the mqAddInteger call as described in “mqAddInteger” on page 491
 - To add 64-bit integer items, use the mqAddInteger64 call as described in “mqAddInteger64” on page 493
 - To add integer filter items, use the mqAddIntegerFilter call as described in “mqAddIntegerFilter” on page 494
 - To add character-string items, use the mqAddString call as described in “mqAddString” on page 496
 - To add string filter items, use the mqAddStringFilter call as described in “mqAddStringFilter” on page 498
 - To add byte string items, use the mqAddByteString call as described in “mqAddByteString” on page 485
 - To add byte string filter items, use the mqAddByteStringFilter call as described in “mqAddByteStringFilter” on page 487
- ..

Adding an inquiry command to a bag

The mqAddInquiry call is used to add an inquiry command to a bag. The call is specifically for administration purposes, so it can be used with administration bags only. It lets you specify the selectors of attributes on which you want to inquire from WebSphere MQ.

For a full description of the mqAddInquiry call, see “mqAddInquiry” on page 489.

Filtering and querying data items

When using the MQAI to inquire about the attributes of WebSphere MQ objects, you can control the data that is returned to your program in two ways.

1. You can *filter* the data that is returned using the mqAddInteger and mqAddString calls. This approach lets you specify a *Selector* and *ItemValue* pair, for example:

```
mqAddInteger(inputbag, MQIA_Q_TYPE, MQQT_LOCAL)
```

This example specifies that the queue type (*Selector*) must be local (*ItemValue*) and this specification must match the attributes of the object (in this case, a queue) about which you are inquiring.

Other attributes that can be filtered correspond to the PCF Inquire* commands that can be found in part 1 of this book. For example, to inquire about the attributes of a channel, see the Inquire Channel command in this book. The “Required parameters” and “Optional parameters” of the Inquire Channel command identify the selectors that you can use for filtering.

2. You can *query* particular attributes of an object using the mqAddInquiry call. This specifies the selector in which you are interested. If you do not specify the selector, all attributes of the object are returned.

Here is an example of filtering and querying the attributes of a queue:

```
/* Request information about all queues */
mqAddString(adminbag, MQCA_Q_NAME, "*")

/* Filter attributes so that local queues only are returned */
mqAddInteger(adminbag, MQIA_Q_TYPE, MQQT_LOCAL)

/* Query the names and current depths of the local queues */
mqAddInquiry(adminbag, MQCA_Q_NAME)
mqAddInquiry(adminbag, MQIA_CURRENT_Q_DEPTH)

/* Send inquiry to the command server and wait for reply */
mqExecute(MQCMD_INQUIRE_Q, ...)
```

For more examples of filtering and querying data items, see Chapter 11, “Examples of using the MQAI,” on page 571.

Changing information within a bag

The MQAI lets you change information within a bag using the mqSet* calls. You can:

1. Modify data items within a bag. The index allows an individual instance of a parameter to be replaced by identifying the occurrence of the item to be modified (see Figure 4).

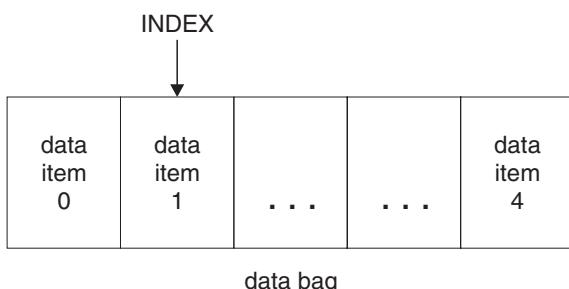


Figure 4. Modifying a single data item

2. Delete all existing occurrences of the specified selector and add a new occurrence to the end of the bag. (See Figure 5.) A special index value allows *all* instances of a parameter to be replaced.

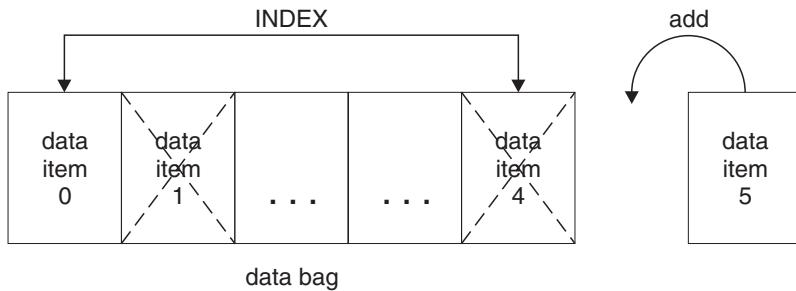


Figure 5. Modifying all data items

Note: The index preserves the insertion order within the bag but can affect the indices of other data items.

The mqSetInteger call lets you modify integer items within a bag. The mqSetInteger64 call lets you modify 64-bit integer items. The mqSetIntegerFilter call lets you modify integer filter items. The mqSetString call lets you modify character-string items. The mqSetStringFilter call lets you modify string filter items. The mqSetByteString call lets you modify byte string items. The mqSetByteStringFilter call lets you modify byte string filter items. Alternatively, you can use these calls to delete all existing occurrences of the specified selector and add a new occurrence at the end of the bag. The data item can be a user item or a system item.

For a full description of these calls, see:

- “mqSetInteger” on page 553
- “mqSetInteger64” on page 555
- “mqSetIntegerFilter” on page 557
- “mqSetString” on page 560
- “mqSetStringFilter” on page 563
- “mqSetByteString” on page 547
- “mqSetByteStringFilter” on page 550

Counting data items

The mqCountItems call counts the number of user items, system items, or both, that are stored in a data bag, and returns this number. For example, `mqCountItems(Bag, 7, ...)`, returns the number of items in the bag with a selector of 7. It can count items by individual selector, by user selectors, by system selectors, or by all selectors.

Note: This call counts the number of data items, not the number of unique selectors in the bag. A selector can occur multiple times, so there may be fewer unique selectors in the bag than data items.

For a full description of the mqCountItems call, see “mqCountItems” on page 505.

Deleting data items

You can delete items from bags in a number of ways. You can:

- Remove one or more user items from a bag,
- Delete *all* user items from a bag, that is, *clear* a bag,
- Delete user items from the end of a bag, that is, *truncate* a bag.

Deleting data items from a bag using the mqDeleteItem call

The mqDeleteItem call removes one or more user items from a bag. The index is used to delete either:

1. A single occurrence of the specified selector. (See Figure 6.)

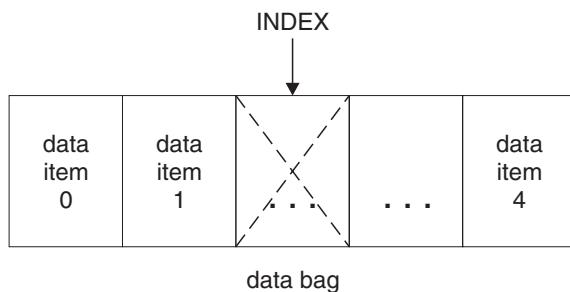


Figure 6. Deleting a single data item

or

2. All occurrences of the specified selector. (See Figure 7.)

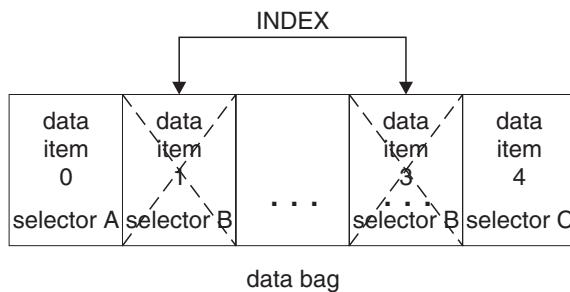


Figure 7. Deleting all data items

Note: The index preserves the insertion order within the bag but can affect the indices of other data items. For example, the mqDeleteItem call does not preserve the index values of the data items that follow the deleted item because the indices are reorganized to fill the gap that remains from the deleted item.

For a full description of the mqDeleteItem call, see “mqDeleteItem” on page 512.

Clearing a bag using the mqClearBag call

The mqClearBag call removes all user items from a user bag and resets system items to their initial values. System bags contained within the bag are also deleted.

For a full description of the mqClearBag call, see “mqClearBag” on page 504.

Truncating a bag using the mqTruncateBag call

The mqTruncateBag call reduces the number of user items in a user bag by deleting the items from the end of the bag, starting with the most recently added item. For example, it can be used when using the same header information to generate more than one message.

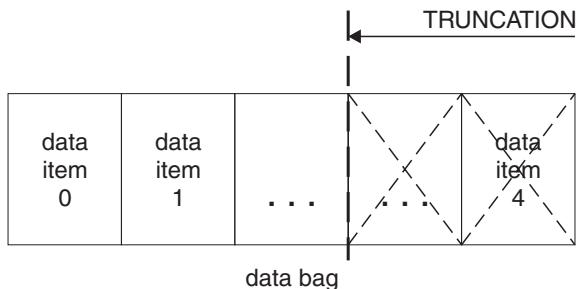


Figure 8. Truncating a bag

For a full description of the mqTruncateBag call, see “mqTruncateBag” on page 567.

Inquiring within data bags

You can inquire about:

- The value of an integer item using the mqInquireInteger call. See “mqInquireInteger” on page 528.
- The value of a 64-bit integer item using the mqInquireInteger64 call. See “mqInquireInteger64” on page 531.
- The value of an integer filter item using the mqInquireIntegerFilter call. See “mqInquireIntegerFilter” on page 533.
- The value of a character-string item using the mqInquireString call. See “mqInquireString” on page 538.
- The value of a string filter item using the mqInquireStringFilter call. See “mqInquireStringFilter” on page 541.
- The value of a byte string item using the mqInquireByteString call. See “mqInquireByteString” on page 523.
- The value of a byte string filter item using the mqInquireByteStringFilter call. See “mqInquireByteStringFilter” on page 525.
- The value of a bag handle using the mqInquireBag call. See “mqInquireBag” on page 520.

You can also inquire about the type (integer, 64-bit integer, integer filter, character string, string filter, byte string, byte string filter or bag handle) of a specific item using the mqInquireItemInfo call. See “mqInquireItemInfo” on page 535.

System items

System items can be used for:

- The generation of PCF headers. System items can control the PCF command identifier, control options, message sequence number, and command type.
- Data conversion. System items handle the character-set identifier for the character-string items in the bag.

Like all data items, system items consist of a selector and a value. For information about these selectors and what they are for, see “MQAI Selectors” on page 568.

System items are unique. One or more system items can be identified by a system selector. There is only one occurrence of each system selector.

Most system items can be modified (see “Changing information within a bag” on page 470), but the bag-creation options cannot be changed by the user. You cannot delete system items. (See “Deleting data items” on page 472.)

Chapter 8. Configuring WebSphere MQ using mqExecute

After you have created and populated your data bag, you can send an administration command message to the command server of a queue manager and wait for any response messages. The easiest way to do this is by using the mqExecute call. This handles the exchange with the command server and returns responses in a bag.

Sending administration commands to the command server

The mqExecute call sends an administration command message as a nonpersistent message and waits for any responses. Responses are returned in a response bag. These might contain information about attributes relating to several WebSphere MQ objects or a series of PCF error response messages, for example. Therefore, the response bag could contain a return code only or it could contain *nested bags*.

Response messages are placed into system bags that are created by the system. For example, for inquiries about the names of objects, a system bag is created to hold those object names and the bag is inserted into the user bag. Handles to these bags are then inserted into the response bag and the nested bag can be accessed by the selector MQHA_BAG_HANDLE. The system bag stays in storage, if it is not deleted, until the response bag is deleted.

The concept of *nesting* is shown in Figure 9.

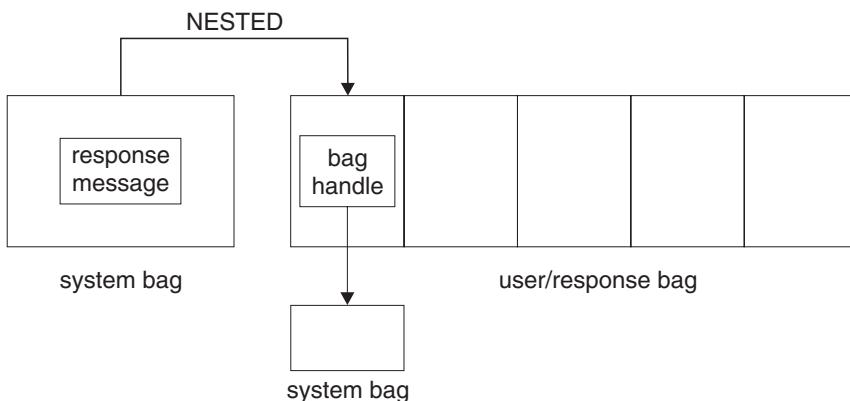


Figure 9. Nesting

As input to the mqExecute call, you must supply:

- An MQI connection handle.
- The command to be executed. This should be one of the MQCMD_* values.

Note: If this value is not recognized by the MQAI, the value is still accepted. However, if the mqAddInquiry call was used to insert values into the bag, this parameter must be an INQUIRE command recognized by the MQAI. That is, the parameter should be of the form MQCMD_INQUIRE_*.

- Optionally, a handle of the bag containing options that control the processing of the call. This is also where you can specify the maximum time in milliseconds that the MQAI should wait for each reply message.

- A handle of the administration bag that contains details of the administration command to be issued.
- A handle of the response bag that receives the reply messages.

The following are optional:

- An object handle of the queue where the administration command is to be placed.

If no object handle is specified, the administration command is placed on the SYSTEM.ADMIN.COMMAND.QUEUE belonging to the currently connected queue manager. This is the default.

- An object handle of the queue where reply messages are to be placed.

You can choose to place the reply messages on a dynamic queue that is created automatically by the MQAI. The queue created exists for the duration of the call only, and is deleted by the MQAI on exit from the mqExecute call.

Example code

Here are some example uses of the mqExecute call.

The example shown in figure Figure 10 creates a local queue (with a maximum message length of 100 bytes) on a queue manager:

```
/* Create a bag for the data you want in your PCF message */
mqCreateBag(MQCBO_ADMIN_BAG, &hbagRequest)

/* Create a bag to be filled with the response from the command server */
mqCreateBag(MQCBO_ADMIN_BAG, &hbagResponse)

/* Create a queue */
/* Supply queue name */
mqAddString(hbagRequest, MQCA_Q_NAME, "QBERT")

/* Supply queue type */
mqAddString(hbagRequest, MQIA_Q_TYPE, MQQT_LOCAL)

/* Maximum message length is an optional parameter */
mqAddString(hbagRequest, MQIA_MAX_MSG_LENGTH, 100)

/* Ask the command server to create the queue */
mqExecute(MQCMD_CREATE_Q, hbagRequest, hbagResponse)

/* Tidy up memory allocated */
mqDeleteBag(hbagRequest)
mqDeleteBag(hbagResponse)
```

Figure 10. Using mqExecute to create a local queue

The example shown in figure Figure 11 on page 477 inquires about all attributes of a particular queue. The mqAddInquiry call identifies all WebSphere MQ object attributes of a queue to be returned by the Inquire parameter on mqExecute.

```

/* Create a bag for the data you want in your PCF message */
mqCreateBag(MQCBO_ADMIN_BAG, &hbagRequest)

/* Create a bag to be filled with the response from the command server */
mqCreateBag(MQCBO_ADMIN_BAG, &hbagResponse)

/* Inquire about a queue by supplying its name */
/* (other parameters are optional) */
mqAddString(hbagRequest, MQCA_Q_NAME, "QBERT")

/* Request the command server to inquire about the queue */
mqExecute(MQCMD_INQUIRE_Q, hbagRequest, hbagResponse)

/* If it worked, the attributes of the queue are returned */
/* in a system bag within the response bag */
mqInquireBag(hbagResponse, MQHA_BAG_HANDLE, 0, &hbagAttributes)

/* Inquire the name of the queue and its current depth */
mqInquireString(hbagAttributes, MQCA_Q_NAME, &stringAttribute)
mqInquireString(hbagAttributes, MQIA_CURRENT_Q_DEPTH, &integerAttribute)

/* Tidy up memory allocated */
mqDeleteBag(hbagRequest)
mqDeleteBag(hbagResponse)

```

Figure 11. Using mqExecute to inquire about queue attributes

Using mqExecute is the simplest way of administering WebSphere MQ, but lower-level calls, mqBagToBuffer and mqBufferToBag, can be used. For more information about the use of these calls, see Chapter 9, “Exchanging data between applications,” on page 479.

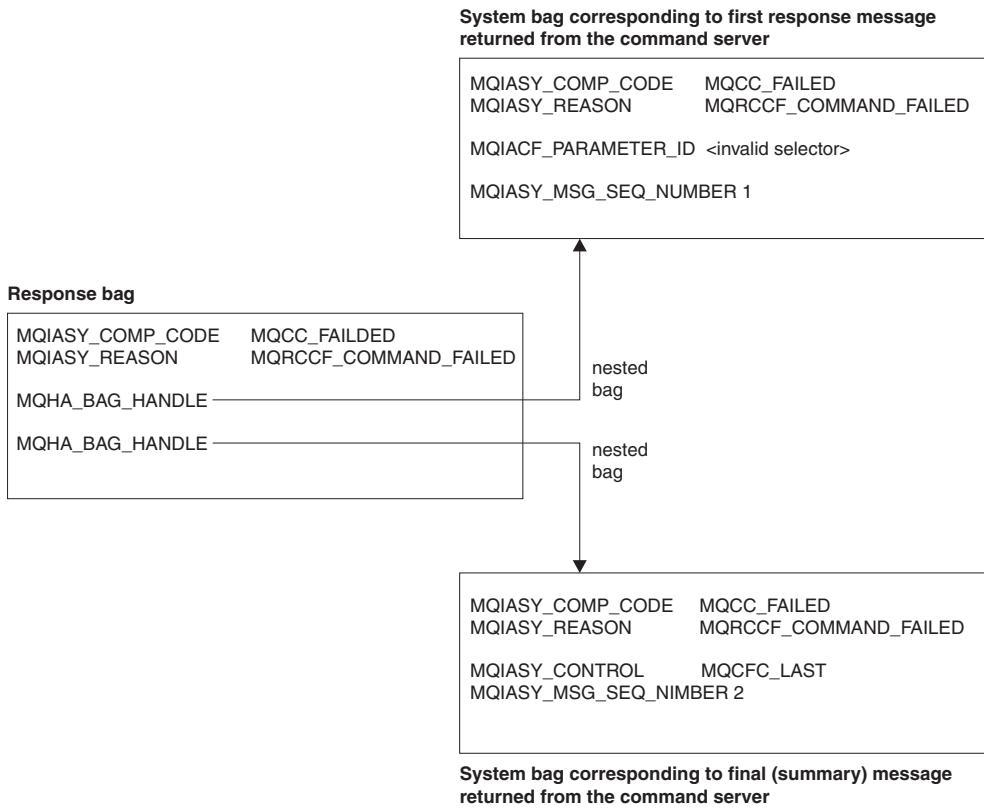
For sample programs, see Chapter 11, “Examples of using the MQAI,” on page 571.

Hints and tips for configuring WebSphere MQ

The MQAI uses PCF messages to send administration commands to the command server rather than dealing directly with the command server itself. Here are some tips for configuring WebSphere MQ using the MQAI:

- Character strings in WebSphere MQ are blank padded to a fixed length. Using C, null-terminated strings can normally be supplied as input parameters to WebSphere MQ programming interfaces.
- To clear the value of a string attribute, set it to a single blank rather than an empty string.
- It is recommended that you know in advance the attributes that you want to change and that you inquire on just those attributes. This is because the number of attributes that can be returned by the Inquire Queue (Response) command is higher than the number of attributes that can be changed using the Change Queue command. (See part 1 of this book for details of these commands.) Therefore, you are not recommended to attempt to modify all the attributes that you inquire.
- If an MQAI call fails, some detail of the failure is returned to the response bag. Further detail can then be found in a nested bag that can be accessed by the selector MQHA_BAG_HANDLE. For example, if an mqExecute call fails with a reason code of MQRCCF_COMMAND_FAILED, this information is returned in the response bag. However, a possible reason for this reason code is that a selector specified was not valid for the type of command message and this detail of information is found in a nested bag that can be accessed via a bag handle.

The following diagram shows this:



Chapter 9. Exchanging data between applications

The MQAI can also be used to exchange data between applications. The application data is sent in PCF format and packed and unpacked by the MQAI. If your message data consists of integers and character strings, you can use the MQAI to take advantage of WebSphere MQ built-in data conversion for PCF data. This avoids the need to write data-conversion exits. To exchange data, the sender must first create the message and send it to the receiving application. Then, the receiver must read the message and extract the data. This can be done in two ways:

1. Converting bags and buffers, that is, using the mqBagToBuffer and mqBufferToBag calls.
2. Putting and getting bags, that is, using the mqPutBag and mqGetBag calls to send and receive PCF messages.

Both of these options are described in this topic.

Note: You cannot convert a bag containing nested bags into a message.

Converting bags and buffers

To send data between applications, firstly the message data is placed in a bag. Then, the data in the bag is converted into a PCF message using the mqBagToBuffer call. The PCF message is sent to the required queue using the MQPUT call. This is shown in Figure 12. For a full description of the mqBagToBuffer call, see “mqBagToBuffer” on page 500.

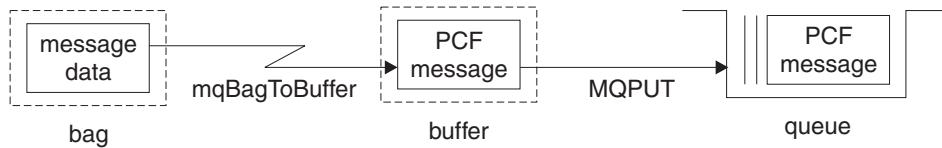


Figure 12. Converting bags to PCF messages

To receive data, the message is received into a buffer using the MQGET call. The data in the buffer is then converted into a bag using the mqBufferToBag call, providing the buffer contains a valid PCF message. This is shown in Figure 13. For a full description of the mqBufferToBag call, see “mqBufferToBag” on page 503.

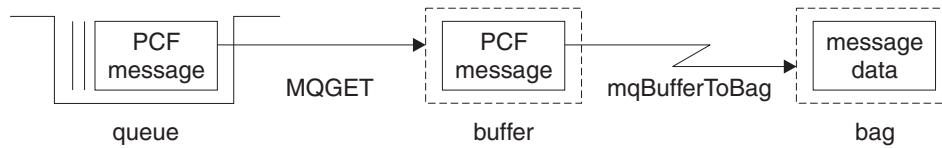


Figure 13. Converting PCF messages to bag form

Putting and receiving data bags

Data can also be sent between applications by putting and getting data bags using the mqPutBag and mqGetBag calls. This lets the MQAI handle the buffer rather than the application. The mqPutBag call converts the contents of the specified bag into a PCF message and sends the message to the specified queue and the mqGetBag call removes the message from the specified queue and converts it back into a data bag. Therefore, the mqPutBag call is the equivalent of the mqBagToBuffer call followed by MQPUT, and the mqGetBag is the equivalent of the MQGET call followed by mqBufferToBag.

Note: If you choose to use the mqGetBag call, the PCF details within the message must be correct; if they are not, an appropriate error results and the PCF message is not returned.

Sending PCF messages to a specified queue

To send a message to a specified queue, the mqPutBag call converts the contents of the specified bag into a PCF message and sends the message to the specified queue. The contents of the bag are left unchanged after the call.

As input to this call, you must supply:

- An MQI connection handle.
- An object handle for the queue on which the message is to be placed.
- A message descriptor. For more information about the message descriptor, see the *WebSphere MQ Application Programming Reference*.
- Put Message Options using the MQPMO structure. For more information about the MQPMO structure, see the *WebSphere MQ Application Programming Reference*.
- The handle of the bag to be converted to a message.

Note: If the bag contains an administration message and the mqAddInquiry call was used to insert values into the bag, the value of the MQIASY_COMMAND data item must be an INQUIRE command recognized by the MQAI.

For a full description of the mqPutBag call, see “mqPutBag” on page 545.

Receiving PCF messages from a specified queue

To receive a message from a specified queue, the mqGetBag call gets a PCF message from a specified queue and converts the message data into a data bag.

As input to this call, you must supply:

- An MQI connection handle.
- An object handle of the queue from which the message is to be read.
- A message descriptor. Within the MQMD structure, the Format parameter must be MQFMT_ADMIN, MQFMT_EVENT, or MQFMT_PCF.

Note: If the message is received within a unit of work (that is, with the MQGMO_SYNCPOINT option) and the message has an unsupported format, the unit of work can be backed out. The message is then reinstated on the queue and can be retrieved using the MQGET call

instead of the mqGetBag call. For more information about the message descriptor, see the *WebSphere MQ Application Programming Reference*.

- Get Message Options using the MQGMO structure. For more information about the MQGMO structure, see the *WebSphere MQ Application Programming Reference*.
- The handle of the bag to contain the converted message.

For a full description of the mqGetBag call, see “mqGetBag” on page 518.

Chapter 10. MQAI reference

This topic contains reference information for the MQAI.

There are two types of selector: *user selector* and *system selector*. These are described in “MQAI Selectors” on page 568.

There are three types of call:

- Data-bag manipulation calls for configuring data bags:
 - “mqAddBag” on page 484
 - “mqAddByteString” on page 485
 - “mqAddByteStringFilter” on page 487
 - “mqAddInquiry” on page 489
 - “mqAddInteger” on page 491
 - “mqAddInteger64” on page 493
 - “mqAddIntegerFilter” on page 494
 - “mqAddString” on page 496
 - “mqAddStringFilter” on page 498
 - “mqClearBag” on page 504
 - “mqCountItems” on page 505
 - “mqCreateBag” on page 507
 - “mqDeleteBag” on page 511
 - “mqDeleteItem” on page 512
 - “mqInquireBag” on page 520
 - “mqInquireByteString” on page 523
 - “mqInquireByteStringFilter” on page 525
 - “mqInquireInteger” on page 528
 - “mqInquireInteger64” on page 531
 - “mqInquireIntegerFilter” on page 533
 - “mqInquireItemInfo” on page 535
 - “mqInquireString” on page 538
 - “mqInquireStringFilter” on page 541
 - “mqSetByteString” on page 547
 - “mqSetByteStringFilter” on page 550
 - “mqSetInteger” on page 553
 - “mqSetInteger64” on page 555
 - “mqSetIntegerFilter” on page 557
 - “mqSetString” on page 560
 - “mqSetStringFilter” on page 563
 - “mqTruncateBag” on page 567
- Command calls for sending and receiving administration commands and PCF messages:
 - “mqBagToBuffer” on page 500
 - “mqBufferToBag” on page 503

- “mqExecute” on page 514
- “mqGetBag” on page 518
- “mqPutBag” on page 545
- Utility calls for handling blank-padded and null-terminated strings:
 - “mqPad” on page 544
 - “mqTrim” on page 566

These calls are described in alphabetical order in the following sections.

mqAddBag

Note: The mqAddBag call can be used with user bags only; you cannot add nested bags to administration or command bags. You can only nest group bags.

The mqAddBag call nests a bag in another bag.

Syntax

mqAddBag (*Bag*, *Selector*, *ItemValue*, *CompCode*, *Reason*)

Parameters

***Bag* (MQHBAG) – input**

Bag handle into which the item is to be added.

The bag must be a user bag. This means that it must have been created using the MQCBO_USER_BAG option on the mqCreateBag call. If the bag was not created in this way, MQRC_WRONG_BAG_TYPE results.

***Selector* (MQLONG) – input**

Selector identifying the item to be nested.

If the selector is less than zero (that is, a system selector), MQRC_SELECTOR_OUT_OF_RANGE results.

If the selector is zero or greater (that is, a user selector) and the bag was created with the MQCBO_CHECK_SELECTORS option, the selector must be in the range MQGA_FIRST through MQGA_LAST; if not, again MQRC_SELECTOR_OUT_OF_RANGE results.

If MQCBO_CHECK_SELECTORS was not specified, the selector can be any value of zero or greater.

If the call is creating a second or later occurrence of a selector that is already in the bag, the datatype of this occurrence must be the same as the datatype of the first occurrence; MQRC_INCONSISTENT_ITEM_TYPE results if it is not.

***ItemValue* (MQHBAG) – input**

The bag which is to be nested.

If the bag is not a group bag, MQRC_BAG_WRONG_TYPE results. If an attempt is made to add a bag to itself, MQRC_HBAG_ERROR results.

***CompCode* (MQLONG) – output**

Completion code.

***Reason* (MQLONG) – output**

Reason code qualifying *CompCode*.

The following reason codes indicate error conditions that can be returned from the mqAddBag call:

MQRC_BAG_WRONG_TYPE

Wrong type of bag for intended use (either Bag or ItemValue).

MQRC_HBAG_ERROR

Bag handle not valid.

MQRC_INCONSISTENT_ITEM_TYPE

Datatype of this occurrence of selector differs from datatype of first occurrence.

MQRC_SELECTOR_OUT_OF_RANGE

Selector not within valid range for call.

MQRC_STORAGE_NOT_AVAILABLE

Insufficient storage available.

Usage notes

If a bag with the specified selector is already present in the bag, an additional instance of that selector is added to the end of the bag. The new instance is not necessarily adjacent to the existing instance.

C language invocation

```
mqAddBag (Bag, Selector, ItemValue, &CompCode, &Reason)
```

Declare the parameters as follows:

```
MQHBAG    Bag;      /* Bag handle */
MQLONG    Selector; /* Selector */
MQHBAG    ItemValue; /* Nested bag handle */
MQLONG    CompCode; /* Completion code */
MQLONG    Reason;   /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqAddGroup Bag, Selector, ItemValue, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag      As Long 'Bag handle'
Dim Selector As Long 'Selector'
Dim ItemValue As Long 'Nested bag handle'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'
```

mqAddByteString

The mqAddByteString call adds a byte string identified by a user selector to the end of a specified bag.

Syntax

```
mqAddByteString (Bag, Selector, BufferLength, Buffer, CompCode, Reason)
```

Parameters

Bag (**MQHBAG**) – input

Handle of the bag to be modified.

This value must be the handle of a bag created by the user, not the handle of a system bag. MQRC_SYSTEM_BAG_NOT_ALTERABLE results if the value you specify relates to a system bag.

Selector (**MQLONG**) – input

Selector identifying the item to be added to the bag.

If the selector is less than zero (that is, a system selector), MQRC_SELECTOR_OUT_OF_RANGE results.

If the selector is zero or greater (that is, a user selector), and the bag was created with the MQCBO_CHECK_SELECTORS option or as an administration bag (MQCBO_ADMIN_BAG), the selector must be in the range MQBA_FIRST through MQBA_LAST. MQRC_SELECTOR_OUT_OF_RANGE results if it is not in the correct range.

If MQCBO_CHECK_SELECTORS was not specified, the selector can be any value zero or greater.

If the call is creating a second or later occurrence of a selector that is already in the bag, the datatype of this occurrence must be the same as the datatype of the first occurrence; MQRC_INCONSISTENT_ITEM_TYPE results if it is not.

BufferLength (**MQLONG**) – input

The length in bytes of the string contained in the *Buffer* parameter. The value must be zero or greater.

Buffer (**MQBYTE** × *BufferLength*) – input

Buffer containing the byte string.

The length is given by the *BufferLength* parameter. If zero is specified for *BufferLength*, the null pointer can be specified for the address of the *Buffer* parameter. In all other cases, a valid (nonnull) address must be specified for the *Buffer* parameter.

CompCode (**MQLONG**) – output

Completion code.

Reason (**MQLONG**) – output

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqAddByteString call:

MQRC_BUFFER_ERROR

Buffer parameter not valid (invalid parameter address or buffer not completely accessible).

MQRC_BUFFER_LENGTH_ERROR

Buffer length not valid.

MQRC_HBAG_ERROR

Bag handle not valid.

MQRC_INCONSISTENT_ITEM_TYPE

Datatype of this occurrence of selector differs from datatype of first occurrence.

MQRC_SELECTOR_OUT_OF_RANGE

Selector not within valid range for call.

MQRC_STORAGE_NOT_AVAILABLE

Insufficient storage available.

MQRC_SYSTEM_BAG_NOT_ALTERABLE

System bag cannot be altered or deleted.

Usage notes

1. If a data item with the specified selector is already present in the bag, an additional instance of that selector is added to the end of the bag. The new instance is not necessarily adjacent to the existing instance.
2. This call cannot be used to add a system selector to a bag.

C language invocation

```
mqAddByteString (hBag, Selector, BufferLength, Buffer, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG Bag;           /* Bag handle */
MQLONG Selector;     /* Selector */
MQLONG BufferLength; /* Buffer length */
PMQBYTE Buffer;       /* Buffer containing item value */
MQLONG CompCode;     /* Completion code */
MQLONG Reason;        /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqAddByteString Bag, Selector, BufferLength, Buffer, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag      As Long 'Bag handle'
Dim Selector As Long 'Selector'
Dim BufferLength As Long 'Buffer length'
Dim Buffer    As Byte 'Buffer containing item value'
Dim CompCode  As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'
```

mqAddByteStringFilter

The mqAddByteStringFilter call adds a byte string filter identified by a user selector to the end of a specified bag.

Syntax

```
mqAddByteStringFilter (Bag, Selector, BufferLength, Buffer, Operator, CompCode, Reason)
```

Parameters

Bag (MQHBAG) – input

Handle of the bag to be modified.

This value must be the handle of a bag created by the user, not the handle of a system bag. MQRC_SYSTEM_BAG_NOT_ALTERABLE results if the value you specify relates to a system bag.

Selector (MQLONG) – input

Selector identifying the item to be added to the bag.

If the selector is less than zero (that is, a system selector), MQRC_SELECTOR_OUT_OF_RANGE results.

If the selector is zero or greater (that is, a user selector), and the bag was created with the MQCBO_CHECK_SELECTORS option or as an administration bag (MQCBO_ADMIN_BAG), the selector must be in the range MQBA_FIRST through MQBA_LAST. MQRC_SELECTOR_OUT_OF_RANGE results if it is not in the correct range.

If MQCBO_CHECK_SELECTORS was not specified, the selector can be any value zero or greater.

If the call is creating a second or later occurrence of a selector that is already in the bag, the datatype of this occurrence must be the same as the datatype of the first occurrence; MQRC_INCONSISTENT_ITEM_TYPE results if it is not.

BufferLength (MQLONG) – input

The length in bytes of the condition byte string contained in the *Buffer* parameter. The value must be zero or greater.

Buffer (MQBYTE × BufferLength) – input

Buffer containing the condition byte string.

The length is given by the *BufferLength* parameter. If zero is specified for *BufferLength*, the null pointer can be specified for the address of the *Buffer* parameter. In all other cases, a valid (nonnull) address must be specified for the *Buffer* parameter.

Operator (MQLONG) – input

The byte string filter operator to be placed in the bag. Valid operators are of the form MQCFOP_*

CompCode (MQLONG) – output

Completion code.

Reason (MQLONG) – output

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqAddByteStringFilter call:

MQRC_BUFFER_ERROR

Buffer parameter not valid (invalid parameter address or buffer not completely accessible).

MQRC_BUFFER_LENGTH_ERROR

Buffer length not valid.

MQRC_FILTER_OPERATOR_ERROR

Filter operator not valid.

MQRC_HBAG_ERROR

Bag handle not valid.

MQRC_INCONSISTENT_ITEM_TYPE

Datatype of this occurrence of selector differs from datatype of first occurrence.

MQRC_SELECTOR_OUT_OF_RANGE

Selector not within valid range for call.

MQRC_STORAGE_NOT_AVAILABLE

Insufficient storage available.

MQRC_SYSTEM_BAG_NOT_ALTERABLE

System bag cannot be altered or deleted.

Usage notes

1. If a data item with the specified selector is already present in the bag, an additional instance of that selector is added to the end of the bag. The new instance is not necessarily adjacent to the existing instance.
2. This call cannot be used to add a system selector to a bag.

C language invocation

```
mqAddByteStringFilter (hBag, Selector, BufferLength, Buffer, Operator,
&CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG hBag;           /* Bag handle */
MQLONG Selector;      /* Selector */
MQLONG BufferLength;  /* Buffer length */
PMQBYTE Buffer;        /* Buffer containing item value */
MQLONG Operator;       /* Operator */
PMQLONG CompCode;     /* Completion code */
PMQLONG Reason;        /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqAddByteStringFilter Bag, Selector, BufferLength, Buffer, Operator, CompCode,
Reason
```

Declare the parameters as follows:

```
Dim Bag          As Long 'Bag handle'
Dim Selector    As Long 'Selector'
Dim BufferLength As Long 'Buffer length'
Dim Buffer      As String 'Buffer containing item value'
Dim Operator    As Long 'Operator'
Dim CompCode    As Long 'Completion code'
Dim Reason      As Long 'Reason code qualifying CompCode'
```

mqAddInquiry

Note: The mqAddInquiry call can be used with administration bags only; it is specifically for administration purposes.

The mqAddInquiry call adds a selector to an administration bag. The selector refers to a WebSphere MQ object attribute that is to be returned by a PCF INQUIRE command. The value of the Selector parameter specified on this call is added to the end of the bag, as the value of a data item that has the selector value MQIACF_INQUIRY.

Syntax

mqAddInquiry (Bag, Selector, CompCode, Reason)

Parameters

Bag (**MQHBAG**) – **input**

Bag handle.

The bag must be an administration bag; that is, it must have been created with the MQCBO_ADMIN_BAG option on the mqCreateBag call. If the bag was not created this way, MQRC_BAG_WRONG_TYPE results.

Selector (**MQLONG**) – **input**

Selector of the WebSphere MQ object attribute that is to be returned by the appropriate INQUIRE administration command.

CompCode (**MQLONG**) – **output**

Completion code.

Reason (**MQLONG**) – **output**

Reason code qualifying *CompCode*.

The following reason codes indicate error conditions that can be returned from the mqAddInquiry call:

MQRC_BAG_WRONG_TYPE

Wrong type of bag for intended use.

MQRC_HBAG_ERROR

Bag handle not valid.

MQRC_SELECTOR_OUT_OF_RANGE

Selector not within valid range for call.

MQRC_STORAGE_NOT_AVAILABLE

Insufficient storage available.

MQRC_SYSTEM_BAG_NOT_ALTERABLE

System bag cannot be altered or deleted.

Usage notes

1. When the administration message is generated, the MQAI constructs an integer list with the MQIACF_*_ATTRS or MQIACH_*_ATTRS selector that is appropriate to the Command value specified on the mqExecute, mqPutBag, or mqBagToBuffer call. It then adds the values of the attribute selectors specified by the mqAddInquiry call.
2. If the Command value specified on the mqExecute, mqPutBag, or mqBagToBuffer call is not recognized by the MQAI, MQRC_INQUIRY_COMMAND_ERROR results. Instead of using the mqAddInquiry call, this can be overcome by using the mqAddInteger call with the appropriate MQIACF_*_ATTRS or MQIACH_*_ATTRS selector and the ItemValue parameter of the selector being inquired.

C language invocation

```
mqAddInquiry (Bag, Selector, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG Bag;           /* Bag handle */
MQLONG Selector;     /* Selector */
MQLONG CompCode;     /* Completion code */
MQLONG Reason;       /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

`mqAddInquiry Bag, Selector, CompCode, Reason`

Declare the parameters as follows:

```
Dim Bag      As Long 'Bag handle'  
Dim Selector As Long 'Selector'  
Dim CompCode As Long 'Completion code'  
Dim Reason   As Long 'Reason code qualifying CompCode'
```

Supported INQUIRE command codes

- MQCMD_INQUIRE_AUTH_INFO
- MQCMD_INQUIRE_AUTH_RECS
- MQCMD_INQUIRE_AUTH_SERVICE
- MQCMD_INQUIRE_CF_STRUC
- MQCMD_INQUIRE_CHANNEL
- MQCMD_INQUIRE_CHANNEL_STATUS
- MQCMD_INQUIRE_CLUSTER_Q_MGR
- MQCMD_INQUIRE_CONNECTION
- MQCMD_INQUIRE_LISTENER
- MQCMD_INQUIRE_LISTENER_STATUS
- MQCMD_INQUIRE_NAMELIST
- MQCMD_INQUIRE_PROCESS
- MQCMD_INQUIRE_Q
- MQCMD_INQUIRE_Q_MGR
- MQCMD_INQUIRE_Q_MGR_STATUS
- MQCMD_INQUIRE_Q_STATUS
- MQCMD_INQUIRE_SECURITY

For an example that demonstrates the use of supported INQUIRE command codes, see “Inquiring about queues and printing information (amqsailq.c)” on page 575.

mqAddInteger

The mqAddInteger call adds an integer item identified by a user selector to the end of a specified bag.

Syntax

`mqAddInteger (Bag, Selector, ItemValue, CompCode, Reason)`

Parameters

Bag (MQHBAG) – input

Handle of the bag to be modified.

This must be the handle of a bag created by the user, not the handle of a system bag. MQRC_SYSTEM_BAG_NOT_ALTERABLE results if the value you specify identifies a system bag.

Selector (MQLONG) – input

Selector identifying the item to be added to the bag.

If the selector is less than zero (that is, a system selector), MQRC_SELECTOR_OUT_OF_RANGE results.

If the selector is zero or greater (that is, a user selector) and the bag was created with the MQCBO_CHECK_SELECTORS option or as an administration bag (MQCBO_ADMIN_BAG), the selector must be in the range MQIA_FIRST through MQIA_LAST; if not, again MQRC_SELECTOR_OUT_OF_RANGE results.

If MQCBO_CHECK_SELECTORS was not specified, the selector can be any value of zero or greater.

If the call is creating a second or later occurrence of a selector that is already in the bag, the datatype of this occurrence must be the same as the datatype of the first occurrence; MQRC_INCONSISTENT_ITEM_TYPE results if it is not.

ItemValue (MQLONG) – input

The integer value to be placed in the bag.

CompCode (MQLONG) – output

Completion code.

Reason (MQLONG) – output

Reason code qualifying *CompCode*.

The following reason codes indicate error conditions that can be returned from the mqAddInteger call:

MQRC_HBAG_ERROR

Bag handle not valid.

MQRC_INCONSISTENT_ITEM_TYPE

Datatype of this occurrence of selector differs from datatype of first occurrence.

MQRC_SELECTOR_OUT_OF_RANGE

Selector not within valid range for call.

MQRC_STORAGE_NOT_AVAILABLE

Insufficient storage available.

MQRC_SYSTEM_BAG_NOT_ALTERABLE

System bag cannot be altered or deleted.

Usage notes

1. If a data item with the specified selector is already present in the bag, an additional instance of that selector is added to the end of the bag. The new instance is not necessarily adjacent to the existing instance.
2. This call cannot be used to add a system selector to a bag.

C language invocation

```
mqAddInteger (Bag, Selector, ItemValue, &CompCode, &Reason)
```

Declare the parameters as follows:

```
MQHBAG Bag;      /* Bag handle */
MQLONG Selector; /* Selector */
MQLONG ItemValue; /* Integer value */
MQLONG CompCode; /* Completion code */
MQLONG Reason;   /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqAddInteger Bag, Selector, ItemValue, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag      As Long 'Bag handle'  
Dim Selector As Long 'Selector'  
Dim ItemValue As Long 'Integer value'  
Dim CompCode As Long 'Completion code'  
Dim Reason   As Long 'Reason code qualifying CompCode'
```

mqAddInteger64

The mqAddInteger64 call adds a 64-bit integer item identified by a user selector to the end of a specified bag.

Syntax

mqAddInteger64 (*Bag*, *Selector*, *ItemValue*, *CompCode*, *Reason*)

Parameters

Bag (MQHBAG) – input

Handle of the bag to be modified.

This must be the handle of a bag created by the user, not the handle of a system bag. MQRC_SYSTEM_BAG_NOT_ALTERABLE results if the value you specify identifies a system bag.

Selector (MQLONG) – input

Selector identifying the item to be added to the bag.

If the selector is less than zero (that is, a system selector), MQRC_SELECTOR_OUT_OF_RANGE results.

If the selector is zero or greater (that is, a user selector) and the bag was created with the MQCBO_CHECK_SELECTORS option or as an administration bag (MQCBO_ADMIN_BAG), the selector must be in the range MQIA_FIRST through MQIA_LAST; if not, again MQRC_SELECTOR_OUT_OF_RANGE results.

If MQCBO_CHECK_SELECTORS was not specified, the selector can be any value of zero or greater.

If the call is creating a second or later occurrence of a selector that is already in the bag, the datatype of this occurrence must be the same as the datatype of the first occurrence; MQRC_INCONSISTENT_ITEM_TYPE results if it is not.

ItemValue (MQINT64) – input

The 64-bit integer value to be placed in the bag.

CompCode (MQLONG) – output

Completion code.

Reason (MQLONG) – output

Reason code qualifying *CompCode*.

The following reason codes indicate error conditions that can be returned from the mqAddInteger64 call:

MQRC_HBAG_ERROR

Bag handle not valid.

MQRC_INCONSISTENT_ITEM_TYPE

Datatype of this occurrence of selector differs from datatype of first occurrence.

MQRC_SELECTOR_OUT_OF_RANGE

Selector not within valid range for call.

MQRC_STORAGE_NOT_AVAILABLE

Insufficient storage available.

MQRC_SYSTEM_BAG_NOT_ALTERABLE

System bag cannot be altered or deleted.

Usage notes

1. If a data item with the specified selector is already present in the bag, an additional instance of that selector is added to the end of the bag. The new instance is not necessarily adjacent to the existing instance.
2. This call cannot be used to add a system selector to a bag.

C language invocation

```
mqAddInteger64 (Bag, Selector, ItemValue, &CompCode, &Reason)
```

Declare the parameters as follows:

```
MQHBAG Bag; /* Bag handle */
MLONG Selector; /* Selector */
MQINT64 ItemValue; /* Integer value */
MLONG CompCode; /* Completion code */
MLONG Reason; /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqAddInteger64 Bag, Selector, ItemValue, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag As Long 'Bag handle'
Dim Selector As Long 'Selector'
Dim Item Value As Long 'Integer value'
Dim CompCode As Long 'Completion code'
Dim Reason As Long 'Reason code qualifying CompCode'
```

mqAddIntegerFilter

The mqAddIntegerFilter call adds an integer filter identified by a user selector to the end of a specified bag.

Syntax

```
mqAddIntegerFilter (Bag, Selector, ItemValue, Operator, CompCode, Reason)
```

Parameters

Bag (**MQHBAG**) – **input**

Handle of the bag to be modified.

This must be the handle of a bag created by the user, not the handle of a system bag. MQRC_SYSTEM_BAG_NOT_ALTERABLE results if the value you specify identifies a system bag.

Selector (**MQLONG**) – **input**

Selector identifying the item to be added to the bag.

If the selector is less than zero (that is, a system selector), MQRC_SELECTOR_OUT_OF_RANGE results.

If the selector is zero or greater (that is, a user selector) and the bag was created with the MQCBO_CHECK_SELECTORS option or as an administration bag (MQCBO_ADMIN_BAG), the selector must be in the range MQIA_FIRST through MQIA_LAST; if not, again MQRC_SELECTOR_OUT_OF_RANGE results.

If MQCBO_CHECK_SELECTORS was not specified, the selector can be any value of zero or greater.

If the call is creating a second or later occurrence of a selector that is already in the bag, the datatype of this occurrence must be the same as the datatype of the first occurrence; MQRC_INCONSISTENT_ITEM_TYPE results if it is not.

ItemValue (**MQLONG**) – **input**

The integer condition value to be placed in the bag.

Operator (**MQLONG**) – **input**

The integer filter operator to be placed in the bag. Valid operators take the form MQCFOP_*

CompCode (**MQLONG**) – **output**

Completion code.

Reason (**MQLONG**) – **output**

Reason code qualifying *CompCode*.

The following reason codes indicate error conditions that can be returned from the mqAddIntegerFilter call:

MQRC_FILTER_OPERATOR_ERROR

Filter operator not valid.

MQRC_HBAG_ERROR

Bag handle not valid.

MQRC_INCONSISTENT_ITEM_TYPE

Datatype of this occurrence of selector differs from datatype of first occurrence.

MQRC_SELECTOR_OUT_OF_RANGE

Selector not within valid range for call.

MQRC_STORAGE_NOT_AVAILABLE

Insufficient storage available.

MQRC_SYSTEM_BAG_NOT_ALTERABLE

System bag cannot be altered or deleted.

Usage notes

1. If a data item with the specified selector is already present in the bag, an additional instance of that selector is added to the end of the bag. The new instance is not necessarily adjacent to the existing instance.
2. This call cannot be used to add a system selector to a bag.

C language invocation

```
mqAddIntegerFilter (Bag, Selector, ItemValue, Operator, &CompCode, &Reason)
```

Declare the parameters as follows:

```
MQHBAG Bag; /* Bag handle */
MLONG Selector; /* Selector */
MLONG ItemValue; /* Integer value */
MLONG Operator; /* Item operator */
MLONG CompCode; /* Completion code */
MLONG Reason; /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqAddIntegerFilter Bag, Selector, ItemValue, Operator, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag As Long 'Bag handle'
Dim Selector As Long 'Selector'
Dim ItemValue As Long 'Integer value'
Dim Operator As Long 'Item Operator'
Dim CompCode As Long 'Completion code'
Dim Reason As Long 'Reason code qualifying CompCode'
```

mqAddString

The mqAddString call adds a character data item identified by a user selector to the end of a specified bag.

Syntax

```
mqAddString (Bag, Selector, BufferLength, Buffer, CompCode, Reason)
```

Parameters

Bag (MQHBAG) – input

Handle of the bag to be modified.

This value must be the handle of a bag created by the user, not the handle of a system bag. MQRC_SYSTEM_BAG_NOT_ALTERABLE results if the value you specify relates to a system bag.

Selector (MLONG) – input

Selector identifying the item to be added to the bag.

If the selector is less than zero (that is, a system selector), MQRC_SELECTOR_OUT_OF_RANGE results.

If the selector is zero or greater (that is, a user selector), and the bag was created with the MQCBO_CHECK_SELECTORS option or as an administration

bag (MQCBO_ADMIN_BAG), the selector must be in the range MQCA_FIRST through MQCA_LAST. MQRC_SELECTOR_OUT_OF_RANGE results if it is not in the correct range.

If MQCBO_CHECK_SELECTORS was not specified, the selector can be any value zero or greater.

If the call is creating a second or later occurrence of a selector that is already in the bag, the datatype of this occurrence must be the same as the datatype of the first occurrence; MQRC_INCONSISTENT_ITEM_TYPE results if it is not.

BufferLength (MQLONG) – input

The length in bytes of the string contained in the *Buffer* parameter. The value must be zero or greater, or the special value MQBL_NULL_TERMINATED:

- If MQBL_NULL_TERMINATED is specified, the string is delimited by the first null encountered in the string. The null is not added to the bag as part of the string.
- If MQBL_NULL_TERMINATED is not specified, *BufferLength* characters are inserted into the bag, even if null characters are present. Nulls do not delimit the string.

Buffer (MQCHAR × *BufferLength*) – input

Buffer containing the character string.

The length is given by the *BufferLength* parameter. If zero is specified for *BufferLength*, the null pointer can be specified for the address of the *Buffer* parameter. In all other cases, a valid (nonnull) address must be specified for the *Buffer* parameter.

CompCode (MQLONG) – output

Completion code.

Reason (MQLONG) – output

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqAddString call:

MQRC_BUFFER_ERROR

Buffer parameter not valid (invalid parameter address or buffer not completely accessible).

MQRC_BUFFER_LENGTH_ERROR

Buffer length not valid.

MQRC_CODED_CHAR_SET_ID_ERROR

Bag CCSID is MQCCSI_EMBEDDED.

MQRC_HBAG_ERROR

Bag handle not valid.

MQRC_INCONSISTENT_ITEM_TYPE

Datatype of this occurrence of selector differs from datatype of first occurrence.

MQRC_SELECTOR_OUT_OF_RANGE

Selector not within valid range for call.

MQRC_STORAGE_NOT_AVAILABLE

Insufficient storage available.

MQRC_SYSTEM_BAG_NOT_ALTERABLE

System bag cannot be altered or deleted.

Usage notes

1. If a data item with the specified selector is already present in the bag, an additional instance of that selector is added to the end of the bag. The new instance is not necessarily adjacent to the existing instance.
2. This call cannot be used to add a system selector to a bag.
3. The Coded Character Set ID associated with this string is copied from the current CCSID of the bag.

C language invocation

```
mqAddString (hBag, Selector, BufferLength, Buffer, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG hBag;           /* Bag handle */
MQLONG Selector;      /* Selector */
MQLONG BufferLength;  /* Buffer length */
PMQCHAR Buffer;        /* Buffer containing item value */
MQLONG CompCode;       /* Completion code */
MQLONG Reason;         /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqAddString Bag, Selector, BufferLength, Buffer, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag      As Long 'Bag handle'
Dim Selector As Long 'Selector'
Dim BufferLength As Long 'Buffer length'
Dim Buffer    As String 'Buffer containing item value'
Dim CompCode  As Long 'Completion code'
Dim Reason    As Long 'Reason code qualifying CompCode'
```

mqAddStringFilter

The mqAddStringFilter call adds a string filter identified by a user selector to the end of a specified bag.

Syntax

```
mqAddStringFilter (Bag, Selector, BufferLength, Buffer, Operator, CompCode, Reason)
```

Parameters

Bag (MQHBAG) – input

Handle of the bag to be modified.

This value must be the handle of a bag created by the user, not the handle of a system bag. MQRC_SYSTEM_BAG_NOT_ALTERABLE results if the value you specify relates to a system bag.

Selector (MQLONG) – input

Selector identifying the item to be added to the bag.

If the selector is less than zero (that is, a system selector), MQRC_SELECTOR_OUT_OF_RANGE results.

If the selector is zero or greater (that is, a user selector), and the bag was created with the MQCBO_CHECK_SELECTORS option or as an administration bag (MQCBO_ADMIN_BAG), the selector must be in the range MQCA_FIRST through MQCA_LAST. MQRC_SELECTOR_OUT_OF_RANGE results if it is not in the correct range.

If MQCBO_CHECK_SELECTORS was not specified, the selector can be any value zero or greater.

If the call is creating a second or later occurrence of a selector that is already in the bag, the datatype of this occurrence must be the same as the datatype of the first occurrence; MQRC_INCONSISTENT_ITEM_TYPE results if it is not.

BufferLength (MQLONG) – input

The length in bytes of the character condition string contained in the *Buffer* parameter. The value must be zero or greater, or the special value MQBL_NULL_TERMINATED:

- If MQBL_NULL_TERMINATED is specified, the string is delimited by the first null encountered in the string. The null is not added to the bag as part of the string.
- If MQBL_NULL_TERMINATED is not specified, *BufferLength* characters are inserted into the bag, even if null characters are present. Nulls do not delimit the string.

Buffer (MQCHAR × *BufferLength*) – input

Buffer containing the character condition string.

The length is given by the *BufferLength* parameter. If zero is specified for *BufferLength*, the null pointer can be specified for the address of the *Buffer* parameter. In all other cases, a valid (nonnull) address must be specified for the *Buffer* parameter.

Operator (MQLONG) – input

The string filter operator to be placed in the bag. Valid operators are of the form MQCFOP_*

CompCode (MQLONG) – output

Completion code.

Reason (MQLONG) – output

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqAddStringFilter call:

MQRC_BUFFER_ERROR

Buffer parameter not valid (invalid parameter address or buffer not completely accessible).

MQRC_BUFFER_LENGTH_ERROR

Buffer length not valid.

MQRC_CODED_CHAR_SET_ID_ERROR

Bag CCSID is MQCCSI_EMBEDDED.

MQRC_FILTER_OPERATOR_ERROR

Filter operator not valid.

MQRC_HBAG_ERROR

Bag handle not valid.

MQRC_INCONSISTENT_ITEM_TYPE

Datatype of this occurrence of selector differs from datatype of first occurrence.

MQRC_SELECTOR_OUT_OF_RANGE

Selector not within valid range for call.

MQRC_STORAGE_NOT_AVAILABLE

Insufficient storage available.

MQRC_SYSTEM_BAG_NOT_ALTERABLE

System bag cannot be altered or deleted.

Usage notes

1. If a data item with the specified selector is already present in the bag, an additional instance of that selector is added to the end of the bag. The new instance is not necessarily adjacent to the existing instance.
2. This call cannot be used to add a system selector to a bag.
3. The Coded Character Set ID associated with this string is copied from the current CCSID of the bag.

C language invocation

```
mqAddStringFilter (hBag, Selector, BufferLength, Buffer, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG hBag;           /* Bag handle */
MQLONG Selector;       /* Selector */
MQLONG BufferLength;   /* Buffer length */
PMQCHAR Buffer;         /* Buffer containing item value */
MQLONG Operator;        /* Operator */
MQLONG CompCode;        /* Completion code */
MQLONG Reason;          /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqAddStringFilter Bag, Selector, BufferLength, Buffer, Operator, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag      As Long 'Bag handle'
Dim Selector As Long 'Selector'
Dim BufferLength As Long 'Buffer length'
Dim Buffer    As String 'Buffer containing item value'
Dim Operator  As Long 'Item operator'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'
```

mqBagToBuffer

The mqBagToBuffer call converts the bag into a PCF message in the supplied buffer.

Syntax

mqBagToBuffer (*OptionsBag*, *DataBag*, *BufferLength*, *Buffer*, *DataLength*, *CompCode*, *Reason*)

Parameters

OptionsBag (MQHBAG) – input

Handle of the bag containing options that control the processing of the call.
This is a reserved parameter; the value must be MQHB_NONE.

DataBag (MQHBAG) – input

The handle of the bag to convert.

If the bag contains an administration message and mqAddInquiry was used to insert values into the bag, the value of the MQIASY_COMMAND data item must be an INQUIRE command that is recognized by the MQAI; MQRC_INQUIRY_COMMAND_ERROR results if it is not.

If the bag contains nested system bags,
MQRC_NESTED_BAG_NOT_SUPPORTED results.

BufferLength (MQLONG) – input

Length in bytes of the buffer supplied.

If the buffer is too small to accommodate the message generated,
MQRC_BUFFER_LENGTH_ERROR results.

Buffer (MQBYTE × *BufferLength*) – output

The buffer to hold the message.

DataLength (MQLONG) – output

The length in bytes of the buffer required to hold the entire bag. If the buffer is not long enough, the contents of the buffer are undefined but the *DataLength* is returned.

CompCode (MQLONG) – output

Completion code.

Reason (MQLONG) – output

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqBagToBuffer call:

MQRC_BAG_WRONG_TYPE

Input data bag is a group bag.

MQRC_BUFFER_ERROR

Buffer parameter not valid (invalid parameter address or buffer not accessible).

MQRC_BUFFER_LENGTH_ERROR

Buffer length not valid or buffer too small. (Required length returned in *DataLength*.)

MQRC_DATA_LENGTH_ERROR

DataLength parameter not valid (invalid parameter address).

MQRC_HBAG_ERROR

Bag handle not valid.

MQRC_INQUIRY_COMMAND_ERROR

mqAddInquiry used with a command code that is not recognized as an INQUIRE command.

MQRC_NESTED_BAG_NOT_SUPPORTED

Input data bag contains one or more nested system bags.

MQRC_OPTIONS_ERROR

Options bag contains unsupported data items or a supported option has an invalid value.

MQRC_PARAMETER_MISSING

An administration message requires a parameter that is not present in the bag.

Note: This reason code occurs for bags created with the MQCBO_ADMIN_BAG or MQCBO_REORDER_AS_REQUIRED options only.

MQRC_SELECTOR_WRONG_TYPE

mqAddString or mqSetString was used to add the MQIACF_INQUIRY selector to the bag.

MQRC_STORAGE_NOT_AVAILABLE

Insufficient storage available.

Usage notes

1. The PCF message is generated with an encoding of MQENC_NATIVE for the numeric data.
2. The buffer that holds the message can be null if the BufferLength is zero. This is useful if you use the mqBagToBuffer call to calculate the size of buffer necessary to convert your bag.

C language invocation

```
mqBagToBuffer (OptionsBag, DataBag, BufferLength, Buffer, &DataLength,  
&CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG OptionsBag; /* Options bag handle */  
MQHBAG DataBag; /* Data bag handle */  
MQLONG BufferLength; /* Buffer length */  
MQBYTE Buffer[n]; /* Buffer to contain PCF */  
MQLONG DataLength; /* Length of PCF returned in buffer */  
MQLONG CompCode; /* Completion code */  
MQLONG Reason; /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqBagToBuffer OptionsBag, DataBag, BufferLength, Buffer, DataLength,  
CompCode, Reason
```

Declare the parameters as follows:

```
Dim OptionsBag As Long 'Options bag handle'  
Dim DataBag As Long 'Data bag handle'  
Dim BufferLength As Long 'Buffer length'  
Dim Buffer As Long 'Buffer to contain PCF'  
Dim DataLength As Long 'Length of PCF returned in buffer'  
Dim CompCode As Long 'Completion code'  
Dim Reason As Long 'Reason code qualifying CompCode'
```

mqBufferToBag

The mqBufferToBag call converts the supplied buffer into bag form.

Syntax

mqBufferToBag (OptionsBag, BufferLength, Buffer, DataBag, CompCode, Reason)

Parameters

OptionsBag (**MQHBAG**) – **input**

Handle of the bag containing options that control the processing of the call.
This is a reserved parameter; the value must be MQHB_NONE.

BufferLength (**MQLONG**) – **input**

Length in bytes of the buffer.

Buffer (**MQBYTE** × *BufferLength*) – **input**

Pointer to the buffer containing the message to be converted.

Databag (**MQHBAG**) – **input/output**

Handle of the bag to receive the message. The MQAI performs an mqClearBag call on the bag before placing the message in the bag.

CompCode (**MQLONG**) – **output**

Completion code.

Reason (**MQLONG**) – **output**

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqBufferToBag call:

MQRC_BAG_CONVERSION_ERROR

Data could not be converted into a bag. This indicates a problem with the format of the data to be converted into a bag (for example, the message is not a valid PCF).

MQRC_BUFFER_ERROR

Buffer parameter not valid (invalid parameter address or buffer not accessible).

MQRC_BUFFER_LENGTH_ERROR

Buffer length not valid.

MQRC_HBAG_ERROR

Bag handle not valid.

MQRC_INCONSISTENT_ITEM_TYPE

Datatype of second occurrence of selector differs from datatype of first occurrence.

MQRC_OPTIONS_ERROR

Options bag contains unsupported data items, or a supported option has a value that is not valid.

MQRC_SELECTOR_OUT_OF_RANGE

Selector not within valid range for call.

MQRC_STORAGE_NOT_AVAILABLE

Insufficient storage available.

MQRC_SYSTEM_BAG_NOT_ALTERABLE
System bag cannot be altered or deleted.

Usage notes

The buffer must contain a valid PCF message. The encoding of numeric data in the buffer must be MQENC_NATIVE.

The Coded Character Set ID of the bag is unchanged by this call.

C language invocation

```
mqBufferToBag (OptionsBag, BufferLength, Buffer, DataBag,  
&CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG OptionsBag; /* Options bag handle */  
MQLONG BufferLength; /* Buffer length */  
MQBYTE Buffer[n]; /* Buffer containing PCF */  
MQHBAG DataBag; /* Data bag handle */  
MQLONG CompCode; /* Completion code */  
MQLONG Reason; /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqBufferToBag OptionsBag, BufferLength, Buffer, DataBag,  
CompCode, Reason
```

Declare the parameters as follows:

```
Dim OptionsBag As Long 'Options bag handle'  
Dim BufferLength As Long 'Buffer length'  
Dim Buffer As Long 'Buffer containing PCF'  
Dim DataBag As Long 'Data bag handle'  
Dim CompCode As Long 'Completion code'  
Dim Reason As Long 'Reason code qualifying CompCode'
```

mqClearBag

The mqClearBag call deletes all user items from the bag, and resets system items to their initial values.

Syntax

mqClearBag (*Bag*, *CompCode*, *Reason*)

Parameters

Bag (MQHBAG) – input

Handle of the bag to be cleared. This must be the handle of a bag created by the user, not the handle of a system bag.

MQRC_SYSTEM_BAG_NOT_ALTERABLE results if you specify the handle of a system bag.

CompCode (MQLONG) – output

Completion code.

Reason (MQLONG) – output
Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqClearBag call:

MQRC_HBAG_ERROR
Bag handle not valid.

MQRC_SYSTEM_BAG_NOT_ALTERABLE
System bag cannot be altered or deleted.

Usage notes

1. If the bag contains system bags, they are also deleted.
2. The call cannot be used to clear system bags.

C language invocation

```
mqClearBag (Bag, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG    Bag;          /* Bag handle */  
MQLONG    CompCode;     /* Completion code */  
MQLONG    Reason;       /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqClearBag Bag, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag      As Long 'Bag handle'  
Dim CompCode As Long 'Completion code'  
Dim Reason   As Long 'Reason code qualifying CompCode'
```

mqCountItems

The mqCountItems call returns the number of occurrences of user items, system items, or both, that are stored in a bag with the same specific selector.

Syntax

```
mqCountItems (Bag, Selector, ItemCount, CompCode, Reason)
```

Parameters

Bag (MQHBAG) – input

Handle of the bag whose items are to be counted. This can be a user bag or a system bag.

Selector (MQLONG) – input

Selector of the data items to count.

If the selector is less than zero (a system selector), the selector must be one that is supported by the MQAI. MQRC_SELECTOR_NOT_SUPPORTED results if it is not.

If the specified selector is not present in the bag, the call succeeds and zero is returned for *ItemCount*.

The following special values can be specified for *Selector*:

MQSEL_ALL_SELECTORS

All user and system items are to be counted.

MQSEL_ALL_USER_SELECTORS

All user items are to be counted; system items are excluded from the count.

MQSEL_ALL_SYSTEM_SELECTORS

All system items are to be counted; user items are excluded from the count.

***ItemCount* (MQLONG) – output**

Number of items of the specified type in the bag (can be zero).

***CompCode* (MQLONG) – output**

Completion code.

***Reason* (MQLONG) – output**

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqCountItems call:

MQRC_HBAG_ERROR

Bag handle not valid.

MQRC_ITEM_COUNT_ERROR

ItemCount parameter not valid (invalid parameter address).

MQRC_SELECTOR_NOT_SUPPORTED

Specified system selector not supported by the MQAI.

MQRC_SELECTOR_OUT_OF_RANGE

Selector not within valid range for call.

Usage notes

This call counts the number of data items, not the number of unique selectors in the bag. A selector can occur multiple times, so there may be fewer unique selectors in the bag than data items.

C language invocation

```
mqCountItems (Bag, Selector, &ItemCount, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG Bag;           /* Bag handle */
MQLONG Selector;      /* Selector */
MQLONG ItemCount;     /* Number of items */
MQLONG CompCode;       /* Completion code */
MQLONG Reason;         /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqCountItems Bag, Selector, ItemCount, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag;      As Long 'Bag handle'  
Dim Selector As Long 'Selector'  
Dim ItemCount As Long 'Number of items'  
Dim CompCode As Long 'Completion code'  
Dim Reason   As Long 'Reason code qualifying CompCode'
```

mqCreateBag

The mqCreateBag call creates a new bag.

Syntax

mqCreateBag (*Options*, *Bag*, *CompCode*, *Reason*)

Parameters

Options (MQLONG) – input

Options for creation of the bag.

The following are valid:

MQCBO_ADMIN_BAG

Specifies that the bag is for administering WebSphere MQ objects. MQCBO_ADMIN_BAG automatically implies the MQCBO_LIST_FORM_ALLOWED, MQCBO_REORDER_AS_REQUIRED, and MQCBO_CHECK_SELECTORS options.

Administration bags are created with the MQIASY_TYPE system item set to MQCFT_COMMAND.

MQCBO_COMMAND_BAG

Specifies that the bag is a command bag. This is an alternative to the administration bag (MQCBO_ADMIN_BAG) and MQRC_OPTIONS_ERROR results if both are specified.

A command bag is processed in the same way as a user bag except that the value of the MQIASY_TYPE system item is set to MQCFT_COMMAND when the bag is created.

The command bag is also created for administering objects but they are not used to send administration messages to a command server as an administration bag is. The bag options assume the following default values:

- MQCBO_LIST_FORM_INHIBITED
- MQCBO_DO_NOT_REORDER
- MQCBO_DO_NOT_CHECK_SELECTORS

Therefore, the MQAI will not change the order of data items or create lists within a message as with administration bags.

MQCBO_GROUP_BAG

Specifies that the bag is a group bag. This means that the bag is used to hold a set of grouped items. Group bags cannot be used for the administration of WebSphere MQ objects. The bag options assume the following default values:

- MQCBO_LIST_FORM_ALLOWED

- MQCBO_REORDER_AS_REQUIRED
- MQCBO_DO_NOT_CHECK_SELECTORS

Therefore, the MQAI may change the order of data items or create lists within a bag of grouped items.

Group bags are created with two system selectors:
MQIASY_BAG_OPTIONS and MQIASY_CODED_CHAR_SET_ID.

If a group bag is nested in a bag in which
MQCBO_CHECK_SELECTORS was specified, the group bag to be
nested has its selectors checked at that point whether or not
MQCBO_CHECK_SELECTORS was specified when the group bag was
created.

MQCBO_USER_BAG

Specifies that the bag is a user bag. This is the default bag-type option.
User bags can also be used for the administration of WebSphere MQ
objects, but the MQCBO_LIST_FORM_ALLOWED and
MQCBO_REORDER_AS_REQUIRED options should be specified to
ensure correct generation of the administration messages.

User bags are created with the MQIASY_TYPE system item set to
MQCFT_USER.

For user bags, one or more of the following options can be specified:

MQCBO_LIST_FORM_ALLOWED

Specifies that the MQAI is allowed to use the more compact
list form in the message sent whenever there are two or more
adjacent occurrences of the same selector in the bag. However,
this option does not allow the items to be reordered. Therefore,
if the occurrences of the selector are not adjacent in the bag,
and MQCBO_REORDER_AS_REQUIRED is not specified, the
MQAI cannot use the list form for that particular selector.

If the data items are character strings, these strings must have
the same Character Set ID as well as the same selector, in order
to be compacted into list form. If the list form is used, the
shorter strings are padded with blanks to the length of the
longest string.

This option should be specified if the message to be sent is an
administration message but MQCBO_ADMIN_BAG is not
specified.

Note: MQCBO_LIST_FORM_ALLOWED does not imply that
the MQAI will definitely use the list form. The MQAI
considers various factors in deciding whether to use the
list form.

MQCBO_LIST_FORM_INHIBITED

Specifies that the MQAI is not allowed to use the list form in
the message sent, even if there are adjacent occurrences of the
same selector in the bag. This is the default list-form option.

MQCBO_REORDER_AS_REQUIRED

Specifies that the MQAI is allowed to change the order of the
data items in the message sent. This option does not affect the
order of the items in the sending bag.

This means that you can insert items into a data bag in any order; that is, the items do not need to be inserted in the way that they must appear in the PCF message, because the MQAI can reorder these items as required.

If the message is a user message, the order of the items in the receiving bag will be the same as the order of the items in the message; this may be different from the order of the items in the sending bag.

If the message is an administration message, the order of the items in the receiving bag will be determined by the message received.

This option should be specified if the message to be sent is an administration message but MQCBO_ADMIN is not specified.

MQCBO_DO_NOT_REORDER

Specifies that the MQAI is not allowed to change the order of data items in the message sent. Both the message sent and the receiving bag contain the items in the same order as they occur in the sending bag. This is the default ordering option.

MQCBO_CHECK_SELECTORS

Specifies that user selectors (selectors that are zero or greater) should be checked to ensure that the selector is consistent with the datatype implied by the mqAddInteger, mqAddInteger64, mqAddIntegerFilter, mqAddString, mqAddStringFilter, mqAddByteString, mqAddByteStringFilter, mqSetInteger, mqSetInteger64, mqSetIntegerFilter, mqSetString, mqSetStringFilter, mqSetByteString, or mqSetByteStringFilter call:

- For the integer, 64-bit integer, and integer filter calls, the selector must be in the range MQIA_FIRST through MQIA_LAST.
- For the string and string filter calls, the selector must be in the range MQCA_FIRST through MQCA_LAST.
- For byte string and byte string filter calls, the selector must be in the range MQBA_FIRST through MQBA_LAST
- For group bag calls, the selector must be in the range MQGA_FIRST through MQGA_LAST
- For the handle calls, the selector must be in the range MQHA_FIRST through MQHA_LAST.

The call fails if the selector is outside the valid range. Note that system selectors (selectors less than zero) are always checked, and if a system selector is specified, it must be one that is supported by the MQAI.

MQCBO_DO_NOT_CHECK_SELECTORS

Specifies that user selectors (selectors that are zero or greater) should not be checked. This option allows any selector that is zero or positive to be used with any call. This is the default selectors option. Note that system selectors (selectors less than zero) are always checked.

MQCBO_NONE

Specifies that all options should have their default values. This

is provided to aid program documentation, and should not be specified with any of the options that has a nonzero value.

The following list summarizes the default option values:

- MQCBO_USER_BAG
 - MQCBO_LIST_FORM_INHIBITED
 - MQCBO_DO_NOT_REORDER
 - MQCBO_DO_NOT_CHECK_SELECTORS

Bag (MQHBAG) – output

The handle of the bag created by the call.

CompCode (MQLONG) – output

Completion code.

Reason (MQLONG) – output

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqCreateBag call:

MQRC_HBAG_ERROR

Bag handle not valid (invalid parameter address or the parameter location is read-only).

MQRC_OPTIONS_ERROR

Options not valid or not consistent.

MQRC_STORAGE_NOT_AVAILABLE

Insufficient storage available.

Usage notes

Any options used for creating your bag are contained in a system item within the bag when it is created.

C language invocation

```
mqCreateBag (Options, &Bag, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQLONG Options;      /* Bag options */  
MQHBAG Bag;          /* Bag handle */  
MQLONG CompCode;     /* Completion code */  
MQLONG Reason;       /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqCreateBag Options, Bag, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Options As Long 'Bag options'  
Dim Bag As Long 'Bag handle'  
Dim CompCode As Long 'Completion code'  
Dim Reason As Long 'Reason code qualifying CompCode'
```

mqDeleteBag

The mqDeleteBag call deletes the specified bag.

Syntax

mqDeleteBag (Bag, CompCode, Reason)

Parameters

Bag (MQHBAG) – input/output

The handle of the bag to be deleted. This must be the handle of a bag created by the user, not the handle of a system bag.

MQRC_SYSTEM_BAG_NOT_DELETEABLE results if you specify the handle of a system bag. The handle is reset to MQHB_UNUSABLE_HBAG.

If the bag contains system-generated bags, they are also deleted.

CompCode (MQLONG) – output

Completion code.

Reason (MQLONG) – output

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqDeleteBag call:

MQRC_HBAG_ERROR

Bag handle not valid, or invalid parameter address, or parameter location is read only.

MQRC_SYSTEM_BAG_NOT_DELETEABLE

System bag cannot be deleted.

Usage notes

1. Delete any bags created with mqCreateBag.
2. Nested bags are deleted automatically when the containing bag is deleted.

C language invocation

```
mqDeleteBag (&Bag, CompCode, Reason);
```

Declare the parameters as follows:

```
MQHBAG    Bag;          /* Bag handle */  
MQLONG   CompCode;      /* Completion code */  
MQLONG   Reason;        /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqDeleteBag Bag, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag;      As Long 'Bag handle'  
Dim CompCode As Long 'Completion code'  
Dim Reason   As Long 'Reason code qualifying CompCode'
```

mqDeleteItem

The mqDeleteItem call removes one or more user items from a bag.

Syntax

mqDeleteItem (Bag, Selector, ItemIndex, CompCode, Reason)

Parameters

Hbag (**MQHBAG**) – input

Handle of the bag to be modified.

This must be the handle of a bag created by the user, and not the handle of a system bag; MQRC_SYSTEM_BAG_NOT_ALTERABLE results if it is a system bag.

Selector (**MQLONG**) – input

Selector identifying the user item to be deleted.

If the selector is less than zero (that is, a system selector), MQRC_SELECTOR_OUT_OF_RANGE results.

The following special values are valid:

MQSEL_ANY_SELECTOR

The item to be deleted is a user item identified by the *ItemIndex* parameter, the index relative to the set of items that contains both user and system items.

MQSEL_ANY_USER_SELECTOR

The item to be deleted is a user item identified by the *ItemIndex* parameter, the index relative to the set of user items.

If an explicit selector value is specified, but the selector is not present in the bag, the call succeeds if MQIND_ALL is specified for *ItemIndex*, and fails with reason code MQRC_SELECTOR_NOT_PRESENT if MQIND_ALL is not specified.

ItemIndex (**MQLONG**) – input

Index of the data item to be deleted.

The value must be zero or greater, or one of the following special values:

MQIND_NONE

This specifies that there must be one occurrence only of the selector in the bag. If there is more than one occurrence, MQRC_SELECTOR_NOT_UNIQUE results. If MQIND_NONE is specified with one of the MQSEL_XXX_SELECTOR values, MQRC_INDEX_ERROR results.

MQIND_ALL

This specifies that all occurrences of the selector in the bag are to be deleted. If MQIND_ALL is specified with one of the MQSEL_XXX_SELECTOR values, MQRC_INDEX_ERROR results. If MQIND_ALL is specified when the selector is not present within the bag, the call succeeds.

If MQSEL_ANY_SELECTOR is specified for the *Selector* parameter, the *ItemIndex* parameter is the index relative to the set of items that

contains both user items and system items, and must be zero or greater. If ItemIndex identifies a system selector MQRC_SYSTEM_ITEM_NOT_DELETABLE results. If MQSEL_ANY_USER_SELECTOR is specified for the Selector parameter, the ItemIndex parameter is the index relative to the set of user items, and must be zero or greater.

If an explicit selector value is specified, ItemIndex is the index relative to the set of items that have that selector value, and can be MQIND_NONE, MQIND_ALL, zero, or greater.

If an explicit index is specified (that is, not MQIND_NONE or MQIND_ALL) and the item is not present in the bag, MQRC_INDEX_NOT_PRESENT results.

CompCode (MQLONG) – output
Completion code.

Reason (MQLONG) – output
Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqDeleteItem call:

MQRC_HBAG_ERROR

Bag handle not valid.

MQRC_INDEX_ERROR

MQIND_NONE or MQIND_ALL specified with one of the MQSEL_ANY_XXX_SELECTOR values.

MQRC_INDEX_NOT_PRESENT

No item with the specified index is present within the bag.

MQRC_SELECTOR_NOT_PRESENT

No item with the specified selector is present within the bag.

MQRC_SELECTOR_NOT_UNIQUE

MQIND_NONE specified when more than one occurrence of the specified selector is present in the bag.

MQRC_SELECTOR_OUT_OF_RANGE

Selector not within valid range for call.

MQRC_STORAGE_NOT_AVAILABLE

Insufficient storage available.

MQRC_SYSTEM_BAG_NOT_ALTERABLE

System bag is read only and cannot be altered.

MQRC_SYSTEM_ITEM_NOT_DELETABLE

System item is read only and cannot be deleted.

Usage notes

1. Either a single occurrence of the specified selector can be removed, or all occurrences of the specified selector.
2. The call cannot remove system items from the bag, or remove items from a system bag. However, the call can remove the handle of a system bag from a user bag. This way, a system bag can be deleted.

C language invocation

```
mqDeleteItem (Bag, Selector, ItemIndex, &CompCode, &Reason)
```

Declare the parameters as follows:

```
MQHBAG Hbag;           /* Bag handle */
MQLONG Selector;      /* Selector */
MQLONG ItemIndex;     /* Index of the data item */
MQLONG CompCode;      /* Completion code */
MQLONG Reason;        /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqDeleteItem Bag, Selector, ItemIndex, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag      As Long 'Bag handle'
Dim Selector As Long 'Selector'
Dim ItemIndex As Long 'Index of the data item'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'
```

mqExecute

The mqExecute call sends an administration command message and waits for the reply (if expected).

Syntax

```
mqExecute (Hconn, Command, OptionsBag, AdminBag, ResponseBag, AdminQ, ResponseQ, CompCode, Reason)
```

Parameters

Hconn (MQHCONN) – input
MQI Connection handle.

This is returned by a preceding MQCONN call issued by the application.

Command (MQLONG) – input
The command to be executed.

This should be one of the MQCMD_* values. If it is a value that is not recognized by the MQAI servicing the mqExecute call, the value is still accepted. However, if mqAddInquiry was used to insert values in the bag, the Command parameter must be an INQUIRE command recognized by the MQAI; MQRC_INQUIRY_COMMAND_ERROR results if it is not.

OptionsBag (MQHBAG) – input
Handle of a bag containing options that affect the operation of the call.

This must be the handle returned by a preceding mqCreateBag call or the following special value:

MQHB_NONE
No options bag; all options assume their default values.

Only the options listed below can be present in the options bag (MQRC_OPTIONS_ERROR results if other data items are present).

The appropriate default value is used for each option that is not present in the bag. The following option can be specified:

MQIACF_WAIT_INTERVAL

This data item specifies the maximum time in milliseconds that the MQAI should wait for each reply message. The time interval must be zero or greater, or the special value MQWI_UNLIMITED; the default is thirty seconds. The mqExecute call completes either when all of the reply messages are received or when the specified wait interval expires without the expected reply message having been received.

Note: The time interval is an approximate quantity.

If the MQIACF_WAIT_INTERVAL data item has the wrong datatype, or there is more than one occurrence of that selector in the options bag, or the value of the data item is not valid, MQRC_WAIT_INTERVAL_ERROR results.

AdminBag (MQHBAG) – input

Handle of the bag containing details of the administration command to be issued.

All user items placed in the bag are inserted into the administration message that is sent. It is the application's responsibility to ensure that only valid parameters for the command are placed in the bag.

If the value of the MQIASY_TYPE data item in the command bag is not MQCFT_COMMAND, MQRC_COMMAND_TYPE_ERROR results. If the bag contains nested system bags, MQRC_NESTED_BAG_NOT_SUPPORTED results.

ResponseBag (MQHBAG) – input

Handle of the bag where reply messages are placed.

The MQAI performs an mqClearBag call on the bag before placing reply messages in the bag. To retrieve the reply messages, the selector, MQIACF_CONVERT_RESPONSE, can be specified.

Each reply message is placed into a separate system bag, whose handle is then placed in the response bag. Use the mqInquireBag call with selector MQHA_BAG_HANDLE to determine the handles of the system bags within the reply bag, and those bags can then be inquired to determine their contents.

If some but not all of the expected reply messages are received, MQCC_WARNING with MQRC_NO_MSG_AVAILABLE results. If none of the expected reply messages is received, MQCC_FAILED with MQRC_NO_MSG_AVAILABLE results.

Group bags cannot be used as response bags.

AdminQ (MQHOBJ) – input

Object handle of the queue on which the administration message is to be placed.

This handle was returned by a preceding MQOPEN call issued by the application. The queue must be open for output.

The following special value can be specified:

MQHO_NONE

This indicates that the administration message should be placed on the SYSTEM.ADMIN.COMMAND.QUEUE belonging to the currently

connected queue manager. If MQHO_NONE is specified, the application need not use MQOPEN to open the queue.

ResponseQ

Object handle of the queue on which reply messages are placed.

This handle was returned by a preceding MQOPEN call issued by the application. The queue must be open for input and for inquiry.

The following special value can be specified:

MQHO_NONE

This indicates that the reply messages should be placed on a dynamic queue created automatically by the MQAI. The queue is created by opening SYSTEM.DEFAULT.MODEL.QUEUE, that must therefore have suitable characteristics. The queue created exists for the duration of the call only, and is deleted by the MQAI on exit from the mqExecute call.

CompCode

Completion code.

Reason

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqExecute call:

MQRC_*

Anything from the MQINQ, MQPUT, MQGET, or MQOPEN calls.

MQRC_BAG_WRONG_TYPE

Input data bag is a group bag.

MQRC_CMD_SERVER_NOT_AVAILABLE

The command server that processes administration commands is not available.

MQRC_COMMAND_TYPE_ERROR

The value of the MQIASY_TYPE data item in the request bag is not MQCFT_COMMAND.

MQRC_HBAG_ERROR

Bag handle not valid.

MQRC_INQUIRY_COMMAND_ERROR

mqAddInteger call used with a command code that is not a recognized INQUIRE command.

MQRC_NESTED_BAG_NOT_SUPPORTED

Input data bag contains one or more nested system bags.

MQRC_NO_MSG_AVAILABLE

Some reply messages received, but not all. Reply bag contains system-generated bags for messages that were received.

MQRC_NO_MSG_AVAILABLE

No reply messages received during the specified wait interval.

MQRC_OPTIONS_ERROR

Options bag contains unsupported data items, or a supported option has a value which is not valid.

MQRC_PARAMETER_MISSING

Administration message requires a parameter which is not present in

the bag. This reason code occurs for bags created with the MQCBO_ADMIN_BAG or MQCBO_REORDER_AS_REQUIRED options only.

MQRC_SELECTOR_NOT_UNIQUE

Two or more instances of a selector exist within the bag for a mandatory parameter that permits one instance only.

MQRC_SELECTOR_WRONG_TYPE

mqAddString or mqSetString was used to add the MQIACF_INQUIRY selector to the bag.

MQRC_STORAGE_NOT_AVAILABLE

Insufficient storage available.

MQRCCF_COMMAND_FAILED

Command failed; details of failure are contained in system-generated bags within the reply bag.

Usage notes

1. If no *AdminQ* is specified, the MQAI checks to see if the command server is active before sending the administration command message. However, if the command server is not active, the MQAI does not start it. If you are sending a large number of administration command messages, you are recommended to open the SYSTEM.ADMIN.COMMAND.QUEUE yourself and pass the handle of the administration queue on each administration request.
2. Specifying the MQHO_NONE value in the *ResponseQ* parameter simplifies the use of the mqExecute call, but if mqExecute is issued repeatedly by the application (for example, from within a loop), the response queue will be created and deleted repeatedly. In this situation, it is better for the application itself to open the response queue prior to any mqExecute call, and close it after all mqExecute calls have been issued.
3. If the administration command results in a message being sent with a message type of MQMT_REQUEST, the call waits for the period of time given by the MQIACF_WAIT_INTERVAL data item in the options bag.
4. If an error occurs during the processing of the call, the response bag may contain some data from the reply message, but the data will usually be incomplete.

C language invocation

```
mqExecute (Hconn, Command, OptionsBag, AdminBag, ResponseBag,  
AdminQ, ResponseQ, CompCode, Reason);
```

Declare the parameters as follows:

```
MQHCONN Hconn;           /* MQI connection handle */  
MQLONG   Command;        /* Command to be executed */  
MQHBAG   OptionsBag;    /* Handle of a bag containing options */  
MQHBAG   AdminBag;      /* Handle of administration bag containing  
/* details of administration command */  
MQHBAG   ResponseBag;   /* Handle of bag for response messages */  
MQHOBJ   AdminQ;         /* Handle of administration queue for  
administration messages */  
MQHOBJ   ResponseQ;     /* Handle of response queue for response  
messages */  
MQLONG   pCompCode;       /* Completion code */  
MQLONG   pReason;         /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqExecute (Hconn, Command, OptionsBag, AdminBag, ResponseBag,  
AdminQ, ResponseQ, CompCode, Reason);
```

Declare the parameters as follows:

```
Dim HConn      As Long 'MQI connection handle'  
Dim Command    As Long 'Command to be executed'  
Dim OptionsBag As Long 'Handle of a bag containing options'  
Dim AdminBag   As Long 'Handle of command bag containing details of  
                        administration command'  
Dim ResponseBag As Long 'Handle of bag for reply messages'  
Dim AdminQ     As Long 'Handle of command queue for  
                        administration messages'  
Dim ResponseQ   As Long 'Handle of response queue for reply messages'  
Dim CompCode    As Long 'Completion code'  
Dim Reason      As Long 'Reason code qualifying CompCode'
```

mqGetBag

The mqGetBag call removes a message from the specified queue and converts the message data into a data bag.

Syntax

mqGetBag (Hconn, Hobj, MsgDesc, GetMsgOpts, Bag, CompCode, Reason)

Parameters

Hconn (**MQHCONN**) – **input**
MQI connection handle.

Hobj (**MQHOBJ**) – **input**
Object handle of the queue from which the message is to be retrieved. This handle was returned by a preceding MQOPEN call issued by the application. The queue must be open for input.

MsgDesc (**MQMD**) – **input/output**
Message descriptor (for more information, see the *WebSphere MQ Application Programming Reference*).

If the *Format* field in the message has a value other than MQFMT_ADMIN, MQFMT_EVENT, or MQFMT_PCF, MQRC_FORMAT_NOT_SUPPORTED results.

If, on entry to the call, the *Encoding* field in the application's MQMD has a value other than MQENC_NATIVE and MQGMO_CONVERT is specified, MQRC_ENCODING_NOT_SUPPORTED results. Also, if MQGMO_CONVERT is not specified, the value of the *Encoding* parameter must be the retrieving application's MQENC_NATIVE; if not, again MQRC_ENCODING_NOT_SUPPORTED results.

GetMsgOpts (**MQGMO**) – **input/output**
Get-message options (for more information, see the *WebSphere MQ Application Programming Guide*).

MQGMO_ACCEPT_TRUNCATED_MSG cannot be specified; MQRC_OPTIONS_ERROR results if it is. MQGMO_LOCK and MQGMO_UNLOCK are not supported in a 16-bit or 32-bit Window environment. MQGMO_SET_SIGNAL is supported in a 32-bit Window environment only.

***Bag* (MQHBAG) – input/output**

Handle of a bag into which the retrieved message is placed. The MQAI performs an mqClearBag call on the bag before placing the message in the bag.

MQHB_NONE

Gets the retrieved message. This provides a means of deleting messages from the queue.

If an option of MQGMO_BROWSE_* is specified, this value sets the browse cursor to the selected message; it is not deleted in this case.

***CompCode* (MQLONG) – output**

Completion code.

***Reason* (MQLONG) – output**

Reason code qualifying *CompCode*.

The following reason codes indicating warning and error conditions can be returned from the mqGetBag call:

MQRC_*

Anything from the MQGET call or bag manipulation.

MQRC_BAG_CONVERSION_ERROR

Data could not be converted into a bag.

This indicates a problem with the format of the data to be converted into a bag (for example, the message is not a valid PCF).

If the message was retrieved destructively from the queue (that is, not browsing the queue), this reason code indicates that it has been discarded.

MQRC_BAG_WRONG_TYPE

Input data bag is a group bag.

MQRC_ENCODING_NOT_SUPPORTED

Encoding not supported; the *Encoding* field of the MQMD must be MQENC_NATIVE.

MQRC_FORMAT_NOT_SUPPORTED

Format not supported; the *Format* name in the message is not MQFMT_ADMIN, MQFMT_EVENT, or MQFMT_PCF. If the message was retrieved destructively from the queue (that is, not browsing the queue), this reason code indicates that it has been discarded.

MQRC_HBAG_ERROR

Bag handle not valid.

MQRC_INCONSISTENT_ITEM_TYPE

Datatype of second occurrence of selector differs from datatype of first occurrence.

MQRC_SELECTOR_OUT_OF_RANGE

Selector not within valid range for call.

MQRC_STORAGE_NOT_AVAILABLE

Insufficient storage available.

MQRC_SYSTEM_BAG_NOT_ALTERABLE
System bag cannot be altered or deleted.

Usage notes

1. Only messages that have a supported format can be returned by this call. If the message has a format that is not supported, the message is discarded, and the call completes with an appropriate reason code.
2. If the message is retrieved within a unit of work (that is, with the MQGMO_SYNCPOINT option), and the message has an unsupported format, the unit of work can be backed out, reinstating the message on the queue. This allows the message to be retrieved by using the MQGET call in place of the mqGetBag call.

C language invocation

```
mqGetBag (hConn, hObj, &MsgDesc, &GetMsgOpts, hBag, CompCode, Reason);
```

Declare the parameters as follows:

```
MQHCONN hConn;           /* MQI connection handle */
MQHOBJ  hObj;            /* Object handle */
MQMD    MsgDesc;          /* Message descriptor */
MQGMO   GetMsgOpts;       /* Get-message options */
MQHBAG  hBag;             /* Bag handle */
MQLONG  CompCode;         /* Completion code */
MQLONG  Reason;           /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqGetBag (HConn, HObj, MsgDesc, GetMsgOpts, Bag, CompCode, Reason);
```

Declare the parameters as follows:

```
Dim HConn      As Long 'MQI connection handle'
Dim HObj       As Long 'Object handle'
Dim MsgDesc    As Long 'Message descriptor'
Dim GetMsgOpts As Long 'Get-message options'
Dim Bag        As Long 'Bag handle'
Dim CompCode   As Long 'Completion code'
Dim Reason     As Long 'Reason code qualifying CompCode'
```

mqInquireBag

The mqInquireBag call inquires the value of a bag handle that is present in the bag. The data item can be a user item or a system item.

Syntax

```
mqInquireBag (Bag, Selector, ItemIndex, ItemValue, CompCode, Reason)
```

Parameters

Bag (MQHBAG) – input

Bag handle to be inquired. The bag can be a user bag or a system bag.

Selector (MQLONG) – input

Selector identifying the item to be inquired.

If the selector is less than zero (that is, a system selector), the selector must be one that is supported by the MQAI; MQRC_SELECTOR_NOT_SUPPORTED results if it is not.

The specified selector must be present in the bag; MQRC_SELECTOR_NOT_PRESENT results if it is not.

The datatype of the item must agree with the datatype implied by the call; MQRC_SELECTOR_WRONG_TYPE results if it is not.

The following special values can be specified for Selector:

MQSEL_ANY_SELECTOR

The item to be inquired is a user or system item identified by the ItemIndex parameter.

MQSEL_ANY_USER_SELECTOR

The item to be inquired is a user item identified by the ItemIndex parameter.

MQSEL_ANY_SYSTEM_SELECTOR

The item to be inquired is a system item identified by the ItemIndex parameter.

ItemIndex (MQLONG) – input

Index of the data item to be inquired.

The value must be zero or greater, or the special value MQIND_NONE. If the value is less than zero and not MQIND_NONE, MQRC_INDEX_ERROR results. If the item is not already present in the bag, MQRC_INDEX_NOT_PRESENT results.

The following special value can be specified:

MQIND_NONE

This specifies that there must be one occurrence only of the selector in the bag. If there is more than one occurrence, MQRC_SELECTOR_NOT_UNIQUE results.

If MQSEL_ANY_SELECTOR is specified for the Selector parameter, the ItemIndex parameter is the index relative to the set of items that contains both user items and system items, and must be zero or greater.

If MQSEL_ANY_USER_SELECTOR is specified for the Selector parameter, the ItemIndex parameter is the index relative to the set of system items, and must be zero or greater.

If MQSEL_ANY_SYSTEM_SELECTOR is specified for the Selector parameter, the ItemIndex parameter is the index relative to the set of system items, and must be zero or greater.

If an explicit selector value is specified, the ItemIndex parameter is the index relative to the set of items that have that selector value and can be MQIND_NONE, zero, or greater.

ItemValue (MQHBAG) – output

Value of the item in the bag.

CompCode (MQLONG) – output

Completion code.

Reason (MQLONG) – output

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqInquireBag call:

MQRC_HBAG_ERROR

Bag handle not valid.

MQRC_INDEX_ERROR

Index not valid (index negative and not MQIND_NONE, or MQIND_NONE specified with one of the MQSEL_ANY_xxx SELECTOR values).

MQRC_INDEX_NOT_PRESENT

No item with the specified index is present within the bag for the selector given.

MQRC_ITEM_VALUE_ERROR

The ItemValue parameter is not valid (invalid parameter address).

MQRC_SELECTOR_NOT_PRESENT

No item with the specified selector is present within the bag.

MQRC_SELECTOR_NOT_SUPPORTED

Specified system selector not supported by the MQAI.

MQRC_SELECTOR_NOT_UNIQUE

MQIND_NONE specified when more than one occurrence of the specified selector is present within the bag.

MQRC_SELECTOR_OUT_OF_RANGE

Selector not within valid range for call.

MQRC_SELECTOR_WRONG_TYPE

Data item has wrong datatype for call.

MQRC_STORAGE_NOT_AVAILABLE

Insufficient storage available.

C language invocation

```
mqInquireBag (Bag, Selector, ItemIndex, &ItemValue, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG Bag;          /* Bag handle */
MLONG Selector;      /* Selector */
MLONG ItemIndex;     /* Index of the data item to be inquired */
MQHBAG ItemValue;    /* Value of item in the bag */
MLONG CompCode;      /* Completion code */
MLONG Reason;        /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqInquireBag (Bag, Selector, ItemIndex, ItemValue, CompCode, Reason)
```

Declare the parameters as follows:

```
Dim Bag      As Long 'Bag handle'
Dim Selector As Long 'Selector'
Dim ItemIndex As Long 'Index of the data item to be inquired'
Dim ItemValue As Long 'Value of item in the bag'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'
```

mqInquireByteString

The mqInquireByteString call requests the value of a byte string data item that is present in the bag. The data item can be a user item or a system item.

Syntax

`mqInquireByteString (Bag, Selector, ItemIndex, Bufferlength, Buffer, ByteStringLength, CompCode, Reason)`

Parameters

Bag (MQHBAG) – input

Handle of the bag to which the inquiry relates. The bag can be a user bag or a system bag.

Selector (MQLONG) – input

Selector of the item to which the inquiry relates.

If the selector is less than zero (that is, a system selector), the selector must be one that is supported by the MQAI; MQRC_SELECTOR_NOT_SUPPORTED results if it is not.

The specified selector must be present in the bag; MQRC_SELECTOR_NOT_PRESENT results if it is not.

The datatype of the item must be the same as the datatype implied by the call; MQRC_SELECTOR_WRONG_TYPE results if it is not.

The following special values can be specified for *Selector*:

MQSEL_ANY_SELECTOR

The item to be inquired about is a user or system item identified by *ItemIndex*.

MQSEL_ANY_USER_SELECTOR

The item to be inquired about is a user item identified by *ItemIndex*.

MQSEL_ANY_SYSTEM_SELECTOR

The item to be inquired about is a system item identified by *ItemIndex*.

ItemIndex (MQLONG) – input

Index of the data item to which the inquiry relates. The value must be zero or greater, or the special value MQIND_NONE. If the value is less than zero and not MQIND_NONE, MQRC_INDEX_ERROR results. If the item is not already present in the bag, MQRC_INDEX_NOT_PRESENT results. The following special value can be specified:

MQIND_NONE

This specifies that there must be one occurrence only of the selector in the bag. If there is more than one occurrence, MQRC_SELECTOR_NOT_UNIQUE results.

If MQSEL_ANY_SELECTOR is specified for the *Selector* parameter, *ItemIndex* is the index relative to the set of items that contains both user items and system items, and must be zero or greater.

If MQSEL_ANY_USER_SELECTOR is specified for the *Selector* parameter, *ItemIndex* is the index relative to the set of user items, and must be zero or greater.

If MQSEL_ANY_SYSTEM_SELECTOR is specified for *Selector*, *ItemIndex* is the index relative to the set of system items, and must be zero or greater.

If an explicit selector value is specified, *ItemIndex* is the index relative to the set of items that have that selector value, and can be MQIND_NONE, zero, or greater.

***BufferLength* (MQLONG) – input**

Length in bytes of the buffer to receive the byte string. Zero is a valid value.

***Buffer* (MQBYTE × *BufferLength*) – output**

Buffer to receive the byte string. The length is given by the *BufferLength* parameter. If zero is specified for *BufferLength*, the null pointer can be specified for the address of the *Buffer* parameter; in all other cases, a valid (nonnull) address must be specified for the *Buffer* parameter.

The string is padded with nulls to the length of the buffer. If the string is longer than the buffer, the string is truncated to fit; in this case *ByteStringLength* indicates the size of the buffer needed to accommodate the string without truncation.

***ByteStringLength* (MQLONG) – output**

The length in bytes of the string contained in the bag. If the *Buffer* parameter is too small, the length of the string returned is less than *ByteStringLength*.

***CompCode* (MQLONG) – output**

Completion code.

***Reason* (MQLONG) – output**

Reason code qualifying *CompCode*.

The following reason codes indicating error and warning conditions can be returned from the mqInquireByteString call:

MQRC_BUFFER_ERROR

Buffer parameter not valid (invalid parameter address or buffer not completely accessible).

MQRC_BUFFER_LENGTH_ERROR

Buffer length not valid.

MQRC_HBAG_ERROR

Bag handle not valid.

MQRC_INDEX_ERROR

Index not valid (index negative and not MQIND_NONE, or MQIND_NONE specified with one of the MQSEL_ANY_xxx_SELECTOR values).

MQRC_INDEX_NOT_PRESENT

No item with the specified index is present within the bag for the selector given.

MQRC_SELECTOR_NOT_PRESENT

No item with the specified selector is present within the bag.

MQRC_SELECTOR_NOT_SUPPORTED

Specified system selector not supported by the MQAI.

MQRC_SELECTOR_NOT_UNIQUE

MQIND_NONE specified when more than one occurrence of the specified selector is present in the bag.

MQRC_SELECTOR_OUT_OF_RANGE	Selector not within valid range for call.
MQRC_SELECTOR_WRONG_TYPE	Data item has wrong datatype for call.
MQRC_STORAGE_NOT_AVAILABLE	Insufficient storage available.
MQRC_STRING_LENGTH_ERROR	<i>ByteStringLength</i> parameter not valid (invalid parameter address).
MQRC_STRING_TRUNCATED	Data too long for output buffer and has been truncated.

C language invocation

```
mqInquireByteString (Bag, Selector, ItemIndex,
BufferLength, Buffer, &StringLength, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG Bag;           /* Bag handle */
MQLONG Selector;     /* Selector */
MQLONG ItemIndex;    /* Item index */
MQLONG BufferLength; /* Buffer length */
PMQBYTE Buffer;       /* Buffer to contain string */
MQLONG ByteStringLength; /* Length of byte string returned */
MQLONG CompCode;     /* Completion code */
MQLONG Reason;        /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqInquireByteString Bag, Selector, ItemIndex,
BufferLength, Buffer, StringLength, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag          As Long   'Bag handle'
Dim Selector     As Long   'Selector'
Dim ItemIndex    As Long   'Item index'
Dim BufferLength As Long   'Buffer length'
Dim Buffer       As Byte    'Buffer to contain string'
Dim ByteStringLength As Long 'Length of byte string returned'
Dim CompCode     As Long   'Completion code'
Dim Reason       As Long   'Reason code qualifying CompCode'
```

mqInquireByteStringFilter

The mqInquireByteStringFilter call requests the value and operator of a byte string filter item that is present in the bag. The data item can be a user item or a system item.

Syntax

```
mqInquireByteStringFilter (Bag, Selector, ItemIndex, Bufferlength, Buffer, ByteStringLength, Operator,
CompCode, Reason)
```

Parameters

Bag (MQHBAG) – input

Handle of the bag to which the inquiry relates. The bag can be a user bag or a system bag.

Selector (MQLONG) – input

Selector of the item to which the inquiry relates.

If the selector is less than zero (that is, a system selector), the selector must be one that is supported by the MQAI; MQRC_SELECTOR_NOT_SUPPORTED results if it is not.

The specified selector must be present in the bag; MQRC_SELECTOR_NOT_PRESENT results if it is not.

The datatype of the item must be the same as the datatype implied by the call; MQRC_SELECTOR_WRONG_TYPE results if it is not.

The following special values can be specified for *Selector*:

MQSEL_ANY_SELECTOR

The item to be inquired about is a user or system item identified by *ItemIndex*.

MQSEL_ANY_USER_SELECTOR

The item to be inquired about is a user item identified by *ItemIndex*.

MQSEL_ANY_SYSTEM_SELECTOR

The item to be inquired about is a system item identified by *ItemIndex*.

ItemIndex (MQLONG) – input

Index of the data item to which the inquiry relates. The value must be zero or greater, or the special value MQIND_NONE. If the value is less than zero and not MQIND_NONE, MQRC_INDEX_ERROR results. If the item is not already present in the bag, MQRC_INDEX_NOT_PRESENT results. The following special value can be specified:

MQIND_NONE

This specifies that there must be one occurrence only of the selector in the bag. If there is more than one occurrence, MQRC_SELECTOR_NOT_UNIQUE results.

If MQSEL_ANY_SELECTOR is specified for the *Selector* parameter, *ItemIndex* is the index relative to the set of items that contains both user items and system items, and must be zero or greater.

If MQSEL_ANY_USER_SELECTOR is specified for the *Selector* parameter, *ItemIndex* is the index relative to the set of user items, and must be zero or greater.

If MQSEL_ANY_SYSTEM_SELECTOR is specified for *Selector*, *ItemIndex* is the index relative to the set of system items, and must be zero or greater.

If an explicit selector value is specified, *ItemIndex* is the index relative to the set of items that have that selector value, and can be MQIND_NONE, zero, or greater.

BufferLength (MQLONG) – input

Length in bytes of the buffer to receive the condition byte string. Zero is a valid value.

Buffer (**MQBYTE** × *BufferLength*) – **output**

Buffer to receive the condition byte string. The length is given by the *BufferLength* parameter. If zero is specified for *BufferLength*, the null pointer can be specified for the address of the *Buffer* parameter; in all other cases, a valid (nonnull) address must be specified for the *Buffer* parameter.

The string is padded with blanks to the length of the buffer; the string is not null-terminated. If the string is longer than the buffer, the string is truncated to fit; in this case *ByteStringLength* indicates the size of the buffer needed to accommodate the string without truncation.

ByteStringLength (**MQLONG**) – **output**

The length in bytes of the condition string contained in the bag. If the *Buffer* parameter is too small, the length of the string returned is less than *StringLength*.

Operator (**MQLONG**) – **output**

Byte string filter operator in the bag.

CompCode (**MQLONG**) – **output**

Completion code.

Reason (**MQLONG**) – **output**

Reason code qualifying *CompCode*.

The following reason codes indicating error and warning conditions can be returned from the mqInquireByteStringFilter call:

MQRC_BUFFER_ERROR

Buffer parameter not valid (invalid parameter address or buffer not completely accessible).

MQRC_BUFFER_LENGTH_ERROR

Buffer length not valid.

MQRC_FILTER_OPERATOR_ERROR

Filter operator not valid.

MQRC_HBAG_ERROR

Bag handle not valid.

MQRC_INDEX_ERROR

Index not valid (index negative and not MQIND_NONE, or MQIND_NONE specified with one of the MQSEL_ANY_xxx_SELECTOR values).

MQRC_INDEX_NOT_PRESENT

No item with the specified index is present within the bag for the selector given.

MQRC_SELECTOR_NOT_PRESENT

No item with the specified selector is present within the bag.

MQRC_SELECTOR_NOT_SUPPORTED

Specified system selector not supported by the MQAI.

MQRC_SELECTOR_NOT_UNIQUE

MQIND_NONE specified when more than one occurrence of the specified selector is present in the bag.

MQRC_SELECTOR_OUT_OF_RANGE

Selector not within valid range for call.

MQRC_SELECTOR_WRONG_TYPE

Data item has wrong datatype for call.

MQRC_STORAGE_NOT_AVAILABLE

Insufficient storage available.

MQRC_STRING_LENGTH_ERROR

ByteStringLength parameter not valid (invalid parameter address).

MQRC_STRING_TRUNCATED

Data too long for output buffer and has been truncated.

C language invocation

```
mqInquireByteStringFilter (Bag, Selector, ItemIndex,  
BufferLength, Buffer, &ByteStringLength, &Operator, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG Bag;           /* Bag handle */  
MQLONG Selector;     /* Selector */  
MQLONG ItemIndex;    /* Item index */  
MQLONG BufferLength; /* Buffer length */  
PMQBYTE Buffer;       /* Buffer to contain string */  
MQLONG ByteStringLength; /* Length of string returned */  
MQLONG Operator;     /* Item operator */  
PMQLONG CompCode;    /* Completion code */  
PMQLONG Reason;      /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqInquireByteStringFilter Bag, Selector, ItemIndex,  
BufferLength, Buffer, ByteStringLength,  
Operator, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag          As Long   'Bag handle'  
Dim Selector    As Long   'Selector'  
Dim ItemIndex   As Long   'Item index'  
Dim BufferLength As Long   'Buffer length'  
Dim Buffer      As String 'Buffer to contain string'  
Dim ByteStringLength As Long 'Length of byte string returned'  
Dim Operator    As Long   'Operator'  
Dim CompCode    As Long   'Completion code'  
Dim Reason      As Long   'Reason code qualifying CompCode'
```

mqInquireInteger

The mqInquireInteger call requests the value of an integer data item that is present in the bag. The data item can be a user item or a system item.

Syntax

```
mqInquireInteger (Bag, Selector, ItemIndex, ItemValue, CompCode, Reason)
```

Parameters

Bag (MQHBAG) – input

Handle of the bag to which the inquiry relates. The bag can be a user bag or a system bag.

Selector (MQLONG) – input

Selector identifying the item to which the inquiry relates.

If the selector is less than zero (a system selector), the selector must be one that is supported by the MQAI; MQRC_SELECTOR_NOT_SUPPORTED results if it is not.

The specified selector must be present in the bag; MQRC_SELECTOR_NOT_PRESENT results if it is not.

The datatype of the item must agree with the datatype implied by the call; MQRC_SELECTOR_WRONG_TYPE results if it is not.

The following special values can be specified for *Selector*:

MQSEL_ANY_SELECTOR

The item to be inquired about is a user or system item identified by *ItemIndex*.

MQSEL_ANY_USER_SELECTOR

The item to be inquired about is a user item identified by *ItemIndex*.

MQSEL_ANY_SYSTEM_SELECTOR

The item to be inquired about is a system item identified by *ItemIndex*.

ItemIndex (MQLONG) – input

Index of the data item to which the inquiry relates. The value must be zero or greater, or the special value MQIND_NONE. If the value is less than zero and is not MQIND_NONE, MQRC_INDEX_ERROR results. If the item is not already present in the bag, MQRC_INDEX_NOT_PRESENT results. The following special value can be specified:

MQIND_NONE

This specifies that there must be one occurrence only of the selector in the bag. If there is more than one occurrence, MQRC_SELECTOR_NOT_UNIQUE results.

If MQSEL_ANY_SELECTOR is specified for *Selector*, *ItemIndex* is the index relative to the set of items that contains both user items and system items, and must be zero or greater.

If MQSEL_ANY_USER_SELECTOR is specified for *Selector*, *ItemIndex* is the index relative to the set of user items, and must be zero or greater.

If MQSEL_ANY_SYSTEM_SELECTOR is specified for *Selector*, *ItemIndex* is the index relative to the set of system items, and must be zero or greater.

If an explicit selector value is specified, *ItemIndex* is the index relative to the set of items that have that selector value, and can be MQIND_NONE, zero, or greater.

ItemValue (MQLONG) – output

The value of the item in the bag.

CompCode (MQLONG) – output

Completion code.

Reason (MQLONG) – output

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqInquireInteger call:

MQRC_HBAG_ERROR

Bag handle not valid.

MQRC_INDEX_ERROR

Index not valid (index negative and not MQIND_NONE, or MQIND_NONE specified with one of the MQSEL_ANY_xxx SELECTOR values).

MQRC_INDEX_NOT_PRESENT

No item with the specified index is present within the bag for the selector given.

MQRC_ITEM_VALUE_ERROR

ItemValue parameter not valid (invalid parameter address).

MQRC_SELECTOR_NOT_PRESENT

No item with the specified selector is present within the bag.

MQRC_SELECTOR_NOT_SUPPORTED

Specified system selector not supported by the MQAI.

MQRC_SELECTOR_NOT_UNIQUE

MQIND_NONE specified when more than one occurrence of the specified selector is present in the bag.

MQRC_SELECTOR_OUT_OF_RANGE

Selector not within valid range for call.

MQRC_SELECTOR_WRONG_TYPE

Data item has wrong datatype for call.

MQRC_STORAGE_NOT_AVAILABLE

Insufficient storage available.

C language invocation

```
mqInquireInteger (Bag, Selector, ItemIndex, &ItemValue,  
&CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG Bag;          /* Bag handle */  
MQLONG Selector;    /* Selector */  
MQLONG ItemIndex;   /* Item index */  
MQLONG ItemValue;   /* Item value */  
MQLONG CompCode;    /* Completion code */  
MQLONG Reason;      /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqInquireInteger Bag, Selector, ItemIndex, ItemValue,  
CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag      As Long 'Bag handle'  
Dim Selector As Long 'Selector'  
Dim ItemIndex As Long 'Item index'  
Dim ItemValue As Long 'Item value'  
Dim CompCode As Long 'Completion code'  
Dim Reason   As Long 'Reason code qualifying CompCode'
```

mqInquireInteger64

The mqInquireInteger64 call requests the value of a 64-bit integer data item that is present in the bag. The data item can be a user item or a system item.

Syntax

`mqInquireInteger64 (Bag, Selector, ItemIndex, ItemValue, CompCode, Reason)`

Parameters

Bag (MQHBAG) – input

Handle of the bag to which the inquiry relates. The bag can be a user bag or a system bag.

Selector (MQLONG) – input

Selector identifying the item to which the inquiry relates.

If the selector is less than zero (a system selector), the selector must be one that is supported by the MQAI; MQRC_SELECTOR_NOT_SUPPORTED results if it is not.

The specified selector must be present in the bag; MQRC_SELECTOR_NOT_PRESENT results if it is not.

The datatype of the item must agree with the datatype implied by the call; MQRC_SELECTOR_WRONG_TYPE results if it is not.

The following special values can be specified for *Selector*:

MQSEL_ANY_SELECTOR

The item to be inquired about is a user or system item identified by *ItemIndex*.

MQSEL_ANY_USER_SELECTOR

The item to be inquired about is a user item identified by *ItemIndex*.

MQSEL_ANY_SYSTEM_SELECTOR

The item to be inquired about is a system item identified by *ItemIndex*.

ItemIndex (MQLONG) – input

Index of the data item to which the inquiry relates. The value must be zero or greater, or the special value MQIND_NONE. If the value is less than zero and is not MQIND_NONE, MQRC_INDEX_ERROR results. If the item is not already present in the bag, MQRC_INDEX_NOT_PRESENT results. The following special value can be specified:

MQIND_NONE

This specifies that there must be one occurrence only of the selector in the bag. If there is more than one occurrence, MQRC_SELECTOR_NOT_UNIQUE results.

If MQSEL_ANY_SELECTOR is specified for *Selector*, *ItemIndex* is the index relative to the set of items that contains both user items and system items, and must be zero or greater.

If MQSEL_ANY_USER_SELECTOR is specified for *Selector*, *ItemIndex* is the index relative to the set of user items, and must be zero or greater.

If MQSEL_ANY_SYSTEM_SELECTOR is specified for *Selector*, *ItemIndex* is the index relative to the set of system items, and must be zero or greater.

If an explicit selector value is specified, *ItemIndex* is the index relative to the set of items that have that selector value, and can be MQIND_NONE, zero, or greater.

ItemValue (MQINT64) – **output**

The value of the item in the bag.

CompCode (MQLONG) – **output**

Completion code.

Reason (MQLONG) – **output**

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqInquireInteger64 call:

MQRC_HBAG_ERROR

Bag handle not valid.

MQRC_INDEX_ERROR

Index not valid (index negative and not MQIND_NONE, or MQIND_NONE specified with one of the MQSEL_ANY_xxx_SELECTOR values).

MQRC_INDEX_NOT_PRESENT

No item with the specified index is present within the bag for the selector given.

MQRC_ITEM_VALUE_ERROR

ItemValue parameter not valid (invalid parameter address).

MQRC_SELECTOR_NOT_PRESENT

No item with the specified selector is present within the bag.

MQRC_SELECTOR_NOT_SUPPORTED

Specified system selector not supported by the MQAI.

MQRC_SELECTOR_NOT_UNIQUE

MQIND_NONE specified when more than one occurrence of the specified selector is present in the bag.

MQRC_SELECTOR_OUT_OF_RANGE

Selector not within valid range for call.

MQRC_SELECTOR_WRONG_TYPE

Data item has wrong datatype for call.

MQRC_STORAGE_NOT_AVAILABLE

Insufficient storage available.

C language invocation

```
mqInquireInteger64 (Bag, Selector, ItemIndex, &ItemValue,  
&CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG Bag;           /* Bag handle */  
MQLONG Selector;     /* Selector */  
MQLONG ItemIndex;    /* Item index */  
MQINT64 ItemValue;   /* Item value */  
MQLONG CompCode;     /* Completion code */  
MQLONG Reason;       /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqInquireInteger64 Bag, Selector, ItemIndex, ItemValue,  
CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag      As Long 'Bag handle'  
Dim Selector As Long 'Selector'  
Dim ItemIndex As Long 'Item index'  
Dim ItemValue As Long 'Item value'  
Dim CompCode As Long 'Completion code'  
Dim Reason   As Long 'Reason code qualifying CompCode'
```

mqInquireIntegerFilter

The mqInquireIntegerFilter call requests the value and operator of an integer filter item that is present in the bag. The data item can be a user item or a system item.

Syntax

```
mqInquireIntegerFilter (Bag, Selector, ItemIndex, ItemValue, Operator, CompCode, Reason)
```

Parameters

Bag (MQHBAG) – input

Handle of the bag to which the inquiry relates. The bag can be a user bag or a system bag.

Selector (MQLONG) – input

Selector identifying the item to which the inquiry relates.

If the selector is less than zero (a system selector), the selector must be one that is supported by the MQAI; MQRC_SELECTOR_NOT_SUPPORTED results if it is not.

The specified selector must be present in the bag; MQRC_SELECTOR_NOT_PRESENT results if it is not.

The datatype of the item must agree with the datatype implied by the call; MQRC_SELECTOR_WRONG_TYPE results if it is not.

The following special values can be specified for *Selector*:

MQSEL_ANY_SELECTOR

The item to be inquired about is a user or system item identified by *ItemIndex*.

MQSEL_ANY_USER_SELECTOR

The item to be inquired about is a user item identified by *ItemIndex*.

MQSEL_ANY_SYSTEM_SELECTOR

The item to be inquired about is a system item identified by *ItemIndex*.

ItemIndex (MQLONG) – input

Index of the data item to which the inquiry relates. The value must be zero or greater, or the special value MQIND_NONE. If the value is less than zero and is not MQIND_NONE, MQRC_INDEX_ERROR results. If the item is not

already present in the bag, MQRC_INDEX_NOT_PRESENT results. The following special value can be specified:

MQIND_NONE

This specifies that there must be one occurrence only of the selector in the bag. If there is more than one occurrence, MQRC_SELECTOR_NOT_UNIQUE results.

If MQSEL_ANY_SELECTOR is specified for *Selector*, *ItemIndex* is the index relative to the set of items that contains both user items and system items, and must be zero or greater.

If MQSEL_ANY_USER_SELECTOR is specified for *Selector*, *ItemIndex* is the index relative to the set of user items, and must be zero or greater.

If MQSEL_ANY_SYSTEM_SELECTOR is specified for *Selector*, *ItemIndex* is the index relative to the set of system items, and must be zero or greater.

If an explicit selector value is specified, *ItemIndex* is the index relative to the set of items that have that selector value, and can be MQIND_NONE, zero, or greater.

ItemValue (MQLONG) – output

The condition value.

Operator (MQLONG) – output

Integer filter operator in the bag.

CompCode (MQLONG) – output

Completion code.

Reason (MQLONG) – output

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqInquireIntegerFilter call:

MQRC_FILTER_OPERATOR_ERROR

Filter operator not valid.

MQRC_HBAG_ERROR

Bag handle not valid.

MQRC_INDEX_ERROR

Index not valid (index negative and not MQIND_NONE, or MQIND_NONE specified with one of the MQSEL_ANY_xxx_SELECTOR values).

MQRC_INDEX_NOT_PRESENT

No item with the specified index is present within the bag for the selector given.

MQRC_ITEM_VALUE_ERROR

ItemValue parameter not valid (invalid parameter address).

MQRC_SELECTOR_NOT_PRESENT

No item with the specified selector is present within the bag.

MQRC_SELECTOR_NOT_SUPPORTED

Specified system selector not supported by the MQAI.

MQRC_SELECTOR_NOT_UNIQUE

MQIND_NONE specified when more than one occurrence of the specified selector is present in the bag.

MQRC_SELECTOR_OUT_OF_RANGE	Selector not within valid range for call.
MQRC_SELECTOR_WRONG_TYPE	Data item has wrong datatype for call.
MQRC_STORAGE_NOT_AVAILABLE	Insufficient storage available.

C language invocation

```
mqInquireIntegerFilter (Bag, Selector, ItemIndex, &ItemValue,
&Operator, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG Bag; /* Bag handle */
MQLONG Selector; /* Selector */
MQLONG ItemIndex; /* Item index */
MQLONG ItemValue; /* Item value */
MQLONG Operator; /* Item operator */
MQLONG CompCode; /* Completion code */
MQLONG Reason; /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqInquireIntegerFilter Bag, Selector, ItemIndex, ItemValue,
Operator, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag As Long 'Bag handle'
Dim Selector As Long 'Selector'
Dim ItemIndex As Long 'Item index'
Dim ItemValue As Long 'Item value'
Dim Operator As Long 'Item operator'
Dim CompCode As Long 'Completion code'
Dim Reason As Long 'Reason code qualifying CompCode'
```

mqInquireItemInfo

The mqInquireItemInfo call returns information about a specified item in a bag. The data item can be a user item or a system item.

Syntax

```
mqInquireItemInfo (Bag, Selector, ItemIndex, ItemType, OutSelector, CompCode, Reason)
```

Parameters

Bag (MQHBAG) – input

Handle of the bag to be inquired.

The bag can be a user bag or a system bag.

Selector (MQLONG) – input

Selector identifying the item to be inquired.

If the selector is less than zero (that is, a system selector), the selector must be one that is supported by the MQAI; MQRC_SELECTOR_NOT_SUPPORTED results if it is not.

The specified selector must be present in the bag; MQRC_SELECTOR_NOT_PRESENT results if it is not.

The following special values can be specified for Selector:

MQSEL_ANY_SELECTOR

The item to be inquired is a user or system item identified by the ItemIndex parameter.

MQSEL_ANY_USER_SELECTOR

The item to be inquired is a user item identified by the ItemIndex parameter.

MQSEL_ANY_SYSTEM_SELECTOR

The item to be inquired is a system item identified by the ItemIndex parameter.

ItemIndex (MQLONG) – input

Index of the data item to be inquired.

The item must be present within the bag; MQRC_INDEX_NOT_PRESENT results if it is not. The value must be zero or greater, or the following special value:

MQIND_NONE

This specifies that there must be one occurrence only of the selector in the bag. If there is more than one occurrence, MQRC_SELECTOR_NOT_UNIQUE results.

If MQSEL_ANY_SELECTOR is specified for the Selector parameter, the ItemIndex parameter is the index relative to the set of items that contains both user items and system items, and must be zero or greater.

If MQSEL_ANY_USER_SELECTOR is specified for the Selector parameter, the ItemIndex parameter is the index relative to the set of system items, and must be zero or greater.

If MQSEL_ANY_SYSTEM_SELECTOR is specified for the Selector parameter, the ItemIndex parameter is the index relative to the set of system items, and must be zero or greater. If an explicit selector value is specified, the ItemIndex parameter is the index relative to the set of items that have that selector value and can be MQIND_NONE, zero, or greater.

ItemType (MQLONG) – output

The datatype of the specified data item.

The following can be returned:

MQITEM_BAG

Bag handle item.

MQITEM_BYTE_STRING

Byte string.

MQITEM_INTEGER

Integer item.

MQITEM_INTEGER_FILTER

Integer filter.

MQITEM_INTEGER64

64-bit integer item.

MQITEM_STRING

Character-string item.

MQITEM_STRING_FILTER

String filter.

OutSelector (MQLONG) – output

Selector of the specified data item.

CompCode (MQLONG) – output

Completion code.

Reason (MQLONG) – output

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqInquireItemInfo call:

MQRC_HBAG_ERROR

Bag handle not valid.

MQRC_INDEX_ERROR

MQIND_NONE specified with one of the
MQSEL_ANY_XXX_SELECTOR values.

MQRC_INDEX_NOT_PRESENT

No item with the specified index is present within the bag for the selector given.

MQRC_ITEM_TYPE_ERROR

ItemType parameter not valid (invalid parameter address).

MQRC_OUT_SELECTOR_ERROR

OutSelector parameter not valid (invalid parameter address).

MQRC_SELECTOR_NOT_PRESENT

No item with the specified selector is present within the bag.

MQRC_SELECTOR_NOT_SUPPORTED

Specified system selector not supported by the MQAI.

MQRC_SELECTOR_NOT_UNIQUE

MQIND_NONE specified when more than one occurrence of the specified selector is present in the bag.

MQRC_SELECTOR_OUT_OF_RANGE

Selector not within valid range for call.

MQRC_STORAGE_NOT_AVAILABLE

Insufficient storage available.

C language invocation

```
mqInquireItemInfo (Bag, Selector, ItemIndex, &OutSelector, &ItemType,  
&CompCode, &Reason);
```

Declare the parameters as follows:

```

MQHBAG    Bag;          /* Bag handle */
MQLONG    Selector;     /* Selector identifying item */
MQLONG    ItemIndex;    /* Index of data item */
MQLONG    OutSelector;  /* Selector of specified data item */
MQLONG    ItemType;     /* Data type of data item */
MQLONG    CompCode;     /* Completion code */
MQLONG    Reason;       /* Reason code qualifying CompCode */

```

Visual Basic invocation

(Supported on Windows only.)

```
mqInquireItemInfo Bag, Selector, ItemIndex, OutSelector, ItemType,
CompCode, Reason
```

Declare the parameters as follows:

```

Dim Bag      As Long 'Bag handle'
Dim Selector As Long 'Selector identifying item'
Dim ItemIndex As Long 'Index of data item'
Dim OutSelector As Long 'Selector of specified data item'
Dim ItemType As Long 'Data type of data item'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'

```

mqInquireString

The mqInquireString call requests the value of a character data item that is present in the bag. The data item can be a user item or a system item.

Syntax

```
mqInquireString (Bag, Selector, ItemIndex, Bufferlength, Buffer, StringLength, CodedCharSetId, CompCode,
Reason)
```

Parameters

Bag (MQHBAG) – input

Handle of the bag to which the inquiry relates. The bag can be a user bag or a system bag.

Selector (MQLONG) – input

Selector of the item to which the inquiry relates.

If the selector is less than zero (that is, a system selector), the selector must be one that is supported by the MQAI; MQRC_SELECTOR_NOT_SUPPORTED results if it is not.

The specified selector must be present in the bag; MQRC_SELECTOR_NOT_PRESENT results if it is not.

The datatype of the item must be the same as the datatype implied by the call; MQRC_SELECTOR_WRONG_TYPE results if it is not.

The following special values can be specified for *Selector*:

MQSEL_ANY_SELECTOR

The item to be inquired about is a user or system item identified by *ItemIndex*.

MQSEL_ANY_USER_SELECTOR

The item to be inquired about is a user item identified by *ItemIndex*.

MQSEL_ANY_SYSTEM_SELECTOR

The item to be inquired about is a system item identified by *ItemIndex*.

***ItemIndex* (MQLONG) – input**

Index of the data item to which the inquiry relates. The value must be zero or greater, or the special value MQIND_NONE. If the value is less than zero and not MQIND_NONE, MQRC_INDEX_ERROR results. If the item is not already present in the bag, MQRC_INDEX_NOT_PRESENT results. The following special value can be specified:

MQIND_NONE

This specifies that there must be one occurrence only of the selector in the bag. If there is more than one occurrence, MQRC_SELECTOR_NOT_UNIQUE results.

If MQSEL_ANY_SELECTOR is specified for the *Selector* parameter, *ItemIndex* is the index relative to the set of items that contains both user items and system items, and must be zero or greater.

If MQSEL_ANY_USER_SELECTOR is specified for the *Selector* parameter, *ItemIndex* is the index relative to the set of user items, and must be zero or greater.

If MQSEL_ANY_SYSTEM_SELECTOR is specified for *Selector*, *ItemIndex* is the index relative to the set of system items, and must be zero or greater.

If an explicit selector value is specified, *ItemIndex* is the index relative to the set of items that have that selector value, and can be MQIND_NONE, zero, or greater.

***BufferLength* (MQLONG) – input**

Length in bytes of the buffer to receive the string. Zero is a valid value.

***Buffer* (MQCHAR × *BufferLength*) – output**

Buffer to receive the character string. The length is given by the *BufferLength* parameter. If zero is specified for *BufferLength*, the null pointer can be specified for the address of the *Buffer* parameter; in all other cases, a valid (nonnull) address must be specified for the *Buffer* parameter.

The string is padded with blanks to the length of the buffer; the string is not null-terminated. If the string is longer than the buffer, the string is truncated to fit; in this case *StringLength* indicates the size of the buffer needed to accommodate the string without truncation.

***StringLength* (MQLONG) – output**

The length in bytes of the string contained in the bag. If the *Buffer* parameter is too small, the length of the string returned is less than *StringLength*.

***CodedCharSetId* (MQLONG) – output**

The coded character set identifier for the character data in the string. This parameter can be set to a null pointer if not required.

***CompCode* (MQLONG) – output**

Completion code.

***Reason* (MQLONG) – output**

Reason code qualifying *CompCode*.

The following reason codes indicating error and warning conditions can be returned from the mqInquireString call:

MQRC_BUFFER_ERROR

Buffer parameter not valid (invalid parameter address or buffer not completely accessible).

MQRC_BUFFER_LENGTH_ERROR

Buffer length not valid.

MQRC_HBAG_ERROR

Bag handle not valid.

MQRC_INDEX_ERROR

Index not valid (index negative and not MQIND_NONE, or MQIND_NONE specified with one of the MQSEL_ANY_xxx_SELECTOR values).

MQRC_INDEX_NOT_PRESENT

No item with the specified index is present within the bag for the selector given.

MQRC_SELECTOR_NOT_PRESENT

No item with the specified selector is present within the bag.

MQRC_SELECTOR_NOT_SUPPORTED

Specified system selector not supported by the MQAI.

MQRC_SELECTOR_NOT_UNIQUE

MQIND_NONE specified when more than one occurrence of the specified selector is present in the bag.

MQRC_SELECTOR_OUT_OF_RANGE

Selector not within valid range for call.

MQRC_SELECTOR_WRONG_TYPE

Data item has wrong datatype for call.

MQRC_STORAGE_NOT_AVAILABLE

Insufficient storage available.

MQRC_STRING_LENGTH_ERROR

StringLength parameter not valid (invalid parameter address).

MQRC_STRING_TRUNCATED

Data too long for output buffer and has been truncated.

C language invocation

```
mqInquireString (Bag, Selector, ItemIndex,  
BufferLength, Buffer, &StringLength, &CodedCharSetId,  
&CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG Bag;           /* Bag handle */  
MQLONG Selector;     /* Selector */  
MQLONG ItemIndex;    /* Item index */  
MQLONG BufferLength; /* Buffer length */  
PMQCHAR Buffer;      /* Buffer to contain string */  
MQLONG StringLength; /* Length of string returned */  
MQLONG CodedCharSetId /* Coded Character Set ID */  
MQLONG CompCode;     /* Completion code */  
MQLONG Reason;       /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqInquireString Bag, Selector, ItemIndex,  
BufferLength, Buffer, StringLength, CodedCharSetId,  
CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag As Long 'Bag handle'  
Dim Selector As Long 'Selector'  
Dim ItemIndex As Long 'Item index'  
Dim BufferLength As Long 'Buffer length'  
Dim Buffer As String 'Buffer to contain string'  
Dim StringLength As Long 'Length of string returned'  
Dim CodedCharSetId As Long 'Coded Character Set ID'  
Dim CompCode As Long 'Completion code'  
Dim Reason As Long 'Reason code qualifying CompCode'
```

mqInquireStringFilter

The mqInquireStringFilter call requests the value and operator of a string filter item that is present in the bag. The data item can be a user item or a system item.

Syntax

```
mqInquireStringFilter (Bag, Selector, ItemIndex, Bufferlength, Buffer, StringLength, CodedCharSetId,  
Operator, CompCode, Reason)
```

Parameters

Bag (MQHBAG) – input

Handle of the bag to which the inquiry relates. The bag can be a user bag or a system bag.

Selector (MQLONG) – input

Selector of the item to which the inquiry relates.

If the selector is less than zero (that is, a system selector), the selector must be one that is supported by the MQAI; MQRC_SELECTOR_NOT_SUPPORTED results if it is not.

The specified selector must be present in the bag; MQRC_SELECTOR_NOT_PRESENT results if it is not.

The datatype of the item must be the same as the datatype implied by the call; MQRC_SELECTOR_WRONG_TYPE results if it is not.

The following special values can be specified for *Selector*:

MQSEL_ANY_SELECTOR

The item to be inquired about is a user or system item identified by *ItemIndex*.

MQSEL_ANY_USER_SELECTOR

The item to be inquired about is a user item identified by *ItemIndex*.

MQSEL_ANY_SYSTEM_SELECTOR

The item to be inquired about is a system item identified by *ItemIndex*.

ItemIndex (MQLONG) – input

Index of the data item to which the inquiry relates. The value must be zero or greater, or the special value MQIND_NONE. If the value is less than zero and not MQIND_NONE, MQRC_INDEX_ERROR results. If the item is not already present in the bag, MQRC_INDEX_NOT_PRESENT results. The following special value can be specified:

MQIND_NONE

This specifies that there must be one occurrence only of the selector in the bag. If there is more than one occurrence, MQRC_SELECTOR_NOT_UNIQUE results.

If MQSEL_ANY_SELECTOR is specified for the *Selector* parameter, *ItemIndex* is the index relative to the set of items that contains both user items and system items, and must be zero or greater.

If MQSEL_ANY_USER_SELECTOR is specified for the *Selector* parameter, *ItemIndex* is the index relative to the set of user items, and must be zero or greater.

If MQSEL_ANY_SYSTEM_SELECTOR is specified for *Selector*, *ItemIndex* is the index relative to the set of system items, and must be zero or greater.

If an explicit selector value is specified, *ItemIndex* is the index relative to the set of items that have that selector value, and can be MQIND_NONE, zero, or greater.

BufferLength (MQLONG) – input

Length in bytes of the buffer to receive the condition string. Zero is a valid value.

Buffer (MQCHAR × *BufferLength*) – output

Buffer to receive the character condition string. The length is given by the *BufferLength* parameter. If zero is specified for *BufferLength*, the null pointer can be specified for the address of the *Buffer* parameter; in all other cases, a valid (nonnull) address must be specified for the *Buffer* parameter.

The string is padded with blanks to the length of the buffer; the string is not null-terminated. If the string is longer than the buffer, the string is truncated to fit; in this case *StringLength* indicates the size of the buffer needed to accommodate the string without truncation.

StringLength (MQLONG) – output

The length in bytes of the condition string contained in the bag. If the *Buffer* parameter is too small, the length of the string returned is less than *StringLength*.

CodedCharSetId (MQLONG) – output

The coded character set identifier for the character data in the string. This parameter can be set to a null pointer if not required.

Operator (MQLONG) – output

String filter operator in the bag.

CompCode (MQLONG) – output

Completion code.

Reason (MQLONG) – output

Reason code qualifying *CompCode*.

The following reason codes indicating error and warning conditions can be returned from the mqInquireStringFilter call:

MQRC_BUFFER_ERROR

Buffer parameter not valid (invalid parameter address or buffer not completely accessible).

MQRC_BUFFER_LENGTH_ERROR

Buffer length not valid.

MQRC_FILTER_OPERATOR_ERROR

Filter operator not valid.

MQRC_HBAG_ERROR

Bag handle not valid.

MQRC_INDEX_ERROR

Index not valid (index negative and not MQIND_NONE, or MQIND_NONE specified with one of the MQSEL_ANY_xxx_SELECTOR values).

MQRC_INDEX_NOT_PRESENT

No item with the specified index is present within the bag for the selector given.

MQRC_SELECTOR_NOT_PRESENT

No item with the specified selector is present within the bag.

MQRC_SELECTOR_NOT_SUPPORTED

Specified system selector not supported by the MQAI.

MQRC_SELECTOR_NOT_UNIQUE

MQIND_NONE specified when more than one occurrence of the specified selector is present in the bag.

MQRC_SELECTOR_OUT_OF_RANGE

Selector not within valid range for call.

MQRC_SELECTOR_WRONG_TYPE

Data item has wrong datatype for call.

MQRC_STORAGE_NOT_AVAILABLE

Insufficient storage available.

MQRC_STRING_LENGTH_ERROR

StringLength parameter not valid (invalid parameter address).

MQRC_STRING_TRUNCATED

Data too long for output buffer and has been truncated.

C language invocation

```
mqInquireStringFilter (Bag, Selector, ItemIndex,  
BufferLength, Buffer, &StringLength, &CodedCharSetId,  
&Operator, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG    Bag;          /* Bag handle */  
MQLONG    Selector;     /* Selector */  
MQLONG    ItemIndex;    /* Item index */  
MQLONG    BufferLength; /* Buffer length */  
PMQCHAR   Buffer;       /* Buffer to contain string */  
MQLONG    StringLength; /* Length of string returned */  
MQLONG    CodedCharSetId /* Coded Character Set ID */  
MQLONG    Operator;      /* Item operator */  
MQLONG    CompCode;      /* Completion code */  
MQLONG    Reason;        /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqInquireStringFilter Bag, Selector, ItemIndex,  
BufferLength, Buffer, StringLength, CodedCharSetId,  
Operator, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag          As Long   'Bag handle'  
Dim Selector    As Long   'Selector'  
Dim ItemIndex   As Long   'Item index'  
Dim BufferLength As Long   'Buffer length'  
Dim Buffer      As String 'Buffer to contain string'  
Dim StringLength As Long   'Length of string returned'  
Dim CodedCharSetId As Long  'Coded Character Set ID'  
Dim Operator     As Long   'Item operator'  
Dim CompCode     As Long   'Completion code'  
Dim Reason       As Long   'Reason code qualifying CompCode'
```

mqPad

The mqPad call pads a null-terminated string with blanks.

Syntax

mqPad (*String*, *BufferLength*, *Buffer*, *CompCode*, *Reason*)

Parameters

***String* (PMQCHAR) – input**

Null-terminated string. The null pointer is valid for the address of the *String* parameter, and denotes a string of zero length.

***BufferLength* (MQLONG) – input**

Length in bytes of the buffer to receive the string padded with blanks. Must be zero or greater.

***Buffer* (MQCHAR × *BufferLength*) – output**

Buffer to receive the blank-padded string. The length is given by the *BufferLength* parameter. If zero is specified for *BufferLength*, the null pointer can be specified for the address of the *Buffer* parameter; in all other cases, a valid (nonnull) address must be specified for the *Buffer* parameter.

If the number of characters preceding the first null in the *String* parameter is greater than the *BufferLength* parameter, the excess characters are omitted and MQRC_DATA_TRUNCATED results.

***CompCode* (MQLONG) – output**

Completion code.

***Reason* (MQLONG) – output**

Reason code qualifying *CompCode*.

The following reason codes indicating error and warning conditions can be returned from the mqPad call:

MQRC_BUFFER_ERROR

Buffer parameter not valid (invalid parameter address or buffer not completely accessible).

MQRC_BUFFER_LENGTH_ERROR

Buffer length not valid.

MQRC_STRING_ERROR

String parameter not valid (invalid parameter address or buffer not completely accessible).

MQRC_STRING_TRUNCATED

Data too long for output buffer and has been truncated.

Usage notes

1. If the buffer pointers are the same, the padding is done in place. If not, at most *BufferLength* characters are copied into the second buffer; any space remaining, including the null-termination character, is overwritten with spaces.
2. If the *String* and *Buffer* parameters partially overlap, the result is undefined.

C language invocation

```
mqPad (String, BufferLength, Buffer, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQCHAR    String;          /* String to be padded */
MQLONG    BufferLength;    /* Buffer length */
PMQCHAR   Buffer;         /* Buffer to contain padded string */
MQLONG    CompCode;       /* Completion code */
MQLONG    Reason;         /* Reason code qualifying CompCode */
```

Note: This call is not supported in Visual Basic.

mqPutBag

The mqPutBag call converts the contents of the specified bag into a PCF message and sends the message to the specified queue. The contents of the bag are unchanged after the call.

Syntax

```
mqPutBag (Hconn, Hobj, MsgDesc, PutMsgOpts, Bag, CompCode, Reason)
```

Parameters

Hconn (MQHCONN) – input

MQI connection handle.

Hobj (MQHOBJ) – input

Object handle of the queue on which the message is to be placed. This handle was returned by a preceding MQOPEN call issued by the application. The queue must be open for output.

MsgDesc (MQMD) – input/output

Message descriptor. (For more information, see the *WebSphere MQ Application Programming Reference*.)

If the *Format* field has a value other than MQFMT_ADMIN, MQFMT_EVENT, or MQFMT_PCF, MQRC_FORMAT_NOT_SUPPORTED results.

If the *Encoding* field has a value other than MQENC_NATIVE, MQRC_ENCODING_NOT_SUPPORTED results.

PutMsgOpts (MQPMO) – input/output

Put-message options. (For more information, see the *WebSphere MQ Application Programming Reference*.)

Bag (MQHBAG) – input

Handle of the data bag to be converted to a message.

If the bag contains an administration message, and mqAddInquiry was used to insert values into the bag, the value of the MQIASY_COMMAND data item must be an INQUIRE command recognized by the MQAI; MQRC_INQUIRY_COMMAND_ERROR results if it is not.

If the bag contains nested system bags,
MQRC_NESTED_BAG_NOT_SUPPORTED results.

CompCode (MQLONG) – output

Completion code.

Reason (MQLONG) – output

Reason code qualifying *CompCode*. The following reason codes indicating error and warning conditions can be returned from the mqPutBag call:

MQRC_*

Anything from the MQPUT call or bag manipulation.

MQRC_BAG_WRONG_TYPE

Input data bag is a group bag.

MQRC_ENCODING_NOT_SUPPORTED

Encoding not supported (value in *Encoding* field in MQMD must be MQENC_NATIVE).

MQRC_FORMAT_NOT_SUPPORTED

Format not supported (name in *Format* field in MQMD must be MQFMT_ADMIN, MQFMT_EVENT, or MQFMT_PCF).

MQRC_HBAG_ERROR

Bag handle not valid.

MQRC_INQUIRY_COMMAND_ERROR

mqAddInquiry call used with a command code that is not a recognized INQUIRE command.

MQRC_NESTED_BAG_NOT_SUPPORTED

Input data bag contains one or more nested system bags.

MQRC_PARAMETER_MISSING

Administration message requires a parameter that is not present in the bag. This reason code occurs for bags created with the MQCBO_ADMIN_BAG or MQCBO_REORDER_AS_REQUIRED options only.

MQRC_SELECTOR_WRONG_TYPE

mqAddString or mqSetString was used to add the MQIACF_INQUIRY selector to the bag.

MQRC_STORAGE_NOT_AVAILABLE

Insufficient storage available.

C language invocation

```
mqPutBag (HConn, HObj, &MsgDesc, &PutMsgOpts, Bag,  
&CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHCONN HConn;           /* MQI connection handle */  
MQHOBJ HObj;            /* Object handle */  
MQMD MsgDesc;           /* Message descriptor */  
MQPMO PutMsgOpts;       /* Put-message options */  
MQHBAG Bag;              /* Bag handle */  
MQLONG CompCode;         /* Completion code */  
MQLONG Reason;           /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqPutBag (HConn, HObj, MsgDesc, PutMsgOpts, Bag,  
          CompCode, Reason);
```

Declare the parameters as follows:

```
Dim HConn      As Long   'MQI connection handle'  
Dim HObj       As Long   'Object handle'  
Dim MsgDesc    As MQMD   'Message descriptor'  
Dim PutMsgOpts As MQPMO  'Put-message options'  
Dim Bag        As Long   'Bag handle'  
Dim CompCode   As Long   'Completion code'  
Dim Reason     As Long   'Reason code qualifying CompCode'
```

mqSetByteString

The mqSetByteString call either modifies a byte string data item that is already present in the bag, or deletes all existing occurrences of the specified selector and adds a new occurrence at the end of the bag. The data item is usually a user item, but certain system-data items can also be modified.

Syntax

```
mqSetByteString (Bag, Selector, ItemIndex, Bufferlength, Buffer, CompCode, Reason)
```

Parameters

Bag (MQHBAG) – input

Handle of the bag to be set. This must be the handle of a bag created by the user, not the handle of a system bag;
MQRC_SYSTEM_BAG_NOT_ALTERABLE results if you specify the handle of a system bag.

Selector (MQLONG) – input

Selector of the item to be modified.

If the selector is less than zero (that is, a system selector), the selector must be one that is supported by the MQAI; MQRC_SELECTOR_NOT_SUPPORTED results if it is not.

If the selector is a supported system selector, but is one that is read only, MQRC_SYSTEM_ITEM_NOT_ALTERABLE results.

If the selector is an alterable system selector, but is always a single-instance selector and the application attempts to create a second instance in the bag, MQRC_MULTIPLE_INSTANCE_ERROR results.

If the selector is zero or greater (that is, a user selector), and the bag was created with the MQCBO_CHECK_SELECTORS option or as an administration bag (MQCBO_ADMIN_BAG), the selector must be in the range MQBA_FIRST through MQBA_LAST; MQRC_SELECTOR_OUT_OF_RANGE results if it is not. If MQCBO_CHECK_SELECTORS was not specified, the selector can be any value zero or greater.

If MQIND_ALL is *not* specified for the *ItemIndex* parameter, the specified selector must already be present in the bag; MQRC_SELECTOR_NOT_PRESENT results if it is not.

If MQIND_ALL is *not* specified for the *ItemIndex* parameter, the datatype of the item must be the same as the datatype implied by the call; MQRC_SELECTOR_WRONG_TYPE results if it is not.

ItemIndex (MQLONG) – input

This identifies which occurrence of the item with the specified selector is to be modified. The value must be zero or greater, or one of the special values described below; if it is none of these, MQRC_INDEX_ERROR results.

Zero or greater

The item with the specified index must already be present in the bag; MQRC_INDEX_NOT_PRESENT results if it is not. The index is counted relative to the items in the bag that have the specified selector. For example, if there are five items in the bag with the specified selector, the valid values for *ItemIndex* are 0 through 4.

MQIND_NONE

This specifies that there must be only one occurrence of the specified selector in the bag. If there is more than one occurrence, MQRC_SELECTOR_NOT_UNIQUE results.

MQIND_ALL

This specifies that all existing occurrences of the specified selector (if any) are to be deleted from the bag, and a new occurrence of the selector created at the end of the bag.

BufferLength (MQLONG) – input

The length in bytes of the byte string contained in the *Buffer* parameter. The value must be zero or greater.

Buffer (MQBYTE × *BufferLength*) – input

Buffer containing the byte string. The length is given by the *BufferLength* parameter. If zero is specified for *BufferLength*, the null pointer can be specified for the address of the *Buffer* parameter; in all other cases, a valid (nonnull) address must be specified for the *Buffer* parameter.

CompCode (MQLONG) – output

Completion code.

Reason (MQLONG) – output

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqSetByteString call:

MQRC_BUFFER_ERROR

Buffer parameter not valid (invalid parameter address or buffer not completely accessible).

MQRC_BUFFER_LENGTH_ERROR

Buffer length not valid.

MQRC_HBAG_ERROR

Bag handle not valid.

MQRC_INDEX_ERROR

Index not valid (index negative and not MQIND_NONE or MQIND_ALL).

MQRC_INDEX_NOT_PRESENT

No item with the specified index is present within the bag for the selector given.

MQRC_MULTIPLE_INSTANCE_ERROR

Multiple instances of system selector not valid.

MQRC_SELECTOR_NOT_PRESENT

No item with the specified selector is present within the bag.

MQRC_SELECTOR_NOT_SUPPORTED

Specified system selector not supported by the MQAI.

MQRC_SELECTOR_NOT_UNIQUE

MQIND_NONE specified when more than one occurrence of the specified selector is present in the bag.

MQRC_SELECTOR_OUT_OF_RANGE

Selector not within valid range for call.

MQRC_SELECTOR_WRONG_TYPE

Data item has wrong datatype for call.

MQRC_STORAGE_NOT_AVAILABLE

Insufficient storage available.

MQRC_SYSTEM_BAG_NOT_ALTERABLE

System bag cannot be altered or deleted.

MQRC_SYSTEM_ITEM_NOT_ALTERABLE

System item is read-only and cannot be altered.

C language invocation

```
mqSetByteString (Bag, Selector, ItemIndex, BufferLength, Buffer,  
&CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG Bag;           /* Bag handle */  
MQLONG Selector;     /* Selector */  
MQLONG ItemIndex;    /* Item index */  
MQLONG BufferLength; /* Buffer length */  
PMQBYTE Buffer;      /* Buffer containing string */  
MQLONG CompCode;     /* Completion code */  
MQLONG Reason;       /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqSetByteString Bag, Selector, ItemIndex, BufferLength, Buffer,  
CompCode, Reason
```

Declare the parameters as follows:

Dim Bag	As Long	'Bag handle'
Dim Selector	As Long	'Selector'
Dim ItemIndex	As Long	'Item index'
Dim BufferLength	As Long	'Buffer length'
Dim Buffer	As Byte	'Buffer containing string'
Dim CompCode	As Long	'Completion code'
Dim Reason	As Long	'Reason code qualifying CompCode'

mqSetByteStringFilter

The mqSetByteStringFilter call either modifies a byte string filter item that is already present in the bag, or deletes all existing occurrences of the specified selector and adds a new occurrence at the end of the bag. The data item is usually a user item, but certain system-data items can also be modified.

Syntax

mqSetByteStringFilter (Bag, Selector, ItemIndex, Bufferlength, Buffer, Operator, CompCode, Reason)

Parameters

Bag (MQHBAG) – input

Handle of the bag to be set. This must be the handle of a bag created by the user, not the handle of a system bag; MQRC_SYSTEM_BAG_NOT_ALTERABLE results if you specify the handle of a system bag.

Selector (MQLONG) – input

Selector of the item to be modified.

If the selector is less than zero (that is, a system selector), the selector must be one that is supported by the MQAI; MQRC_SELECTOR_NOT_SUPPORTED results if it is not.

If the selector is a supported system selector, but is one that is read only, MQRC_SYSTEM_ITEM_NOT_ALTERABLE results.

If the selector is an alterable system selector, but is always a single-instance selector and the application attempts to create a second instance in the bag, MQRC_MULTIPLE_INSTANCE_ERROR results.

If the selector is zero or greater (that is, a user selector), and the bag was created with the MQCBO_CHECK_SELECTORS option or as an administration bag (MQCBO_ADMIN_BAG), the selector must be in the range MQBA_FIRST through MQBA_LAST; MQRC_SELECTOR_OUT_OF_RANGE results if it is not. If MQCBO_CHECK_SELECTORS was not specified, the selector can be any value zero or greater.

If MQIND_ALL is *not* specified for the *ItemIndex* parameter, the specified selector must already be present in the bag; MQRC_SELECTOR_NOT_PRESENT results if it is not.

If MQIND_ALL is *not* specified for the *ItemIndex* parameter, the datatype of the item must be the same as the datatype implied by the call; MQRC_SELECTOR_WRONG_TYPE results if it is not.

ItemIndex (MQLONG) – input

This identifies which occurrence of the item with the specified selector is to be modified. The value must be zero or greater, or one of the special values described below; if it is none of these, MQRC_INDEX_ERROR results.

Zero or greater

The item with the specified index must already be present in the bag; MQRC_INDEX_NOT_PRESENT results if it is not. The index is counted relative to the items in the bag that have the specified selector. For example, if there are five items in the bag with the specified selector, the valid values for *ItemIndex* are 0 through 4.

MQIND_NONE

This specifies that there must be only one occurrence of the specified selector in the bag. If there is more than one occurrence, MQRC_SELECTOR_NOT_UNIQUE results.

MQIND_ALL

This specifies that all existing occurrences of the specified selector (if any) are to be deleted from the bag, and a new occurrence of the selector created at the end of the bag.

BufferLength (MQLONG) – input

The length in bytes of the condition byte string contained in the *Buffer* parameter. The value must be zero or greater.

Buffer (MQBYTE × *BufferLength*) – input

Buffer containing the condition byte string. The length is given by the *BufferLength* parameter. If zero is specified for *BufferLength*, the null pointer can be specified for the address of the *Buffer* parameter; in all other cases, a valid (nonnull) address must be specified for the *Buffer* parameter.

Operator (MQLONG × *Operator*) – input

Byte string filter operator to be placed in the bag. Valid operators are of the form MQCFOP_*

CompCode (MQLONG) – output

Completion code.

Reason (MQLONG) – output

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqSetByteStringFilter call:

MQRC_BUFFER_ERROR

Buffer parameter not valid (invalid parameter address or buffer not completely accessible).

MQRC_BUFFER_LENGTH_ERROR

Buffer length not valid.

MQRC_FILTER_OPERATOR_ERROR

Bag handle not valid.

MQRC_HBAG_ERROR

Bag handle not valid.

MQRC_INDEX_ERROR

Index not valid (index negative and not MQIND_NONE or MQIND_ALL).

MQRC_INDEX_NOT_PRESENT

No item with the specified index is present within the bag for the selector given.

MQRC_MULTIPLE_INSTANCE_ERROR

Multiple instances of system selector not valid.

MQRC_SELECTOR_NOT_PRESENT

No item with the specified selector is present within the bag.

MQRC_SELECTOR_NOT_SUPPORTED

Specified system selector not supported by the MQAI.

MQRC_SELECTOR_NOT_UNIQUE

MQIND_NONE specified when more than one occurrence of the specified selector is present in the bag.

MQRC_SELECTOR_OUT_OF_RANGE

Selector not within valid range for call.

MQRC_SELECTOR_WRONG_TYPE

Data item has wrong datatype for call.

MQRC_STORAGE_NOT_AVAILABLE

Insufficient storage available.

MQRC_SYSTEM_BAG_NOT_ALTERABLE

System bag cannot be altered or deleted.

MQRC_SYSTEM_ITEM_NOT_ALTERABLE

System item is read-only and cannot be altered.

C language invocation

```
mqSetByteStringFilter (Bag, Selector, ItemIndex, BufferLength, Buffer,  
Operator, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG Bag;           /* Bag handle */  
MQLONG Selector;     /* Selector */  
MQLONG ItemIndex;    /* Item index */  
MQLONG BufferLength; /* Buffer length */  
PMQBYTE Buffer;      /* Buffer containing string */  
MQLONG Operator;     /* Operator */  
PMQLONG CompCode;    /* Completion code */  
PMQLONG Reason;      /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqSetByteStringFilter Bag, Selector, ItemIndex, BufferLength, Buffer,  
Operator, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag          As Long   'Bag handle'  
Dim Selector     As Long   'Selector'  
Dim ItemIndex    As Long   'Item index'  
Dim BufferLength As Long   'Buffer length'  
Dim Buffer       As String 'Buffer containing string'  
Dim Operator     As Long   'Item operator'  
Dim CompCode     As Long   'Completion code'  
Dim Reason       As Long   'Reason code qualifying CompCode'
```

mqSetInteger

The mqSetInteger call either modifies an integer item that is already present in the bag, or deletes all existing occurrences of the specified selector and adds a new occurrence at the end of the bag. The data item is usually a user item, but specific system-data items can also be modified.

Syntax

`mqSetInteger (Bag, Selector, ItemIndex, ItemValue, CompCode, Reason)`

Parameters

Bag (MQHBAG) – input

Handle of the bag to be set. This must be the handle of a bag created by the user, and not the handle of a system bag; MQRC_SYSTEM_BAG_NOT_ALTERABLE results if the handle you specify refers to a system bag.

Selector (MLQLONG) – input

Selector of the item to be modified. If the selector is less than zero (that is, a system selector), the selector must be one that is supported by the MQAI; MQRC_SELECTOR_NOT_SUPPORTED results if it is not.

If the selector is a supported system selector, but is one that is read-only, MQRC_SYSTEM_ITEM_NOT_ALTERABLE results.

If the selector is an alterable system selector, but is always a single-instance selector and the application attempts to create a second instance in the bag, MQRC_MULTIPLE_INSTANCE_ERROR results.

If the selector is zero or greater (that is, a user selector), and the bag was created with the MQCBO_CHECK_SELECTORS option or as an administration bag (MQCBO_ADMIN_BAG), the selector must be in the range MQIA_FIRST through MQIA_LAST; MQRC_SELECTOR_OUT_OF_RANGE results if it is not. If MQCBO_CHECK_SELECTORS was not specified, the selector can be any value zero or greater.

If MQIND_ALL is *not* specified for the *ItemIndex* parameter, the specified selector must already be present in the bag; MQRC_SELECTOR_NOT_PRESENT results if it is not.

If MQIND_ALL is *not* specified for the *ItemIndex* parameter, the datatype of the item must agree with the datatype implied by the call; MQRC_SELECTOR_WRONG_TYPE results if it is not.

ItemIndex (MLQLONG) – input

This value identifies the occurrence of the item with the specified selector that is to be modified. The value must be zero or greater, or one of the special values described below; if it is none of these, MQRC_INDEX_ERROR results.

Zero or greater

The item with the specified index must already be present in the bag; MQRC_INDEX_NOT_PRESENT results if it is not. The index is counted relative to the items in the bag that have the specified selector. For example, if there are five items in the bag with the specified selector, the valid values for *ItemIndex* are 0 through 4.

MQIND_NONE

This specifies that there must be one occurrence only of the specified selector in the bag. If there is more than one occurrence, MQRC_SELECTOR_NOT_UNIQUE results.

MQIND_ALL

This specifies that all existing occurrences of the specified selector (if any) are to be deleted from the bag, and a new occurrence of the selector created at the end of the bag.

Note: For system selectors, the order is not changed.

***ItemValue* (MQLONG) – input**

The integer value to be placed in the bag.

***CompCode* (MQLONG) – output**

Completion code.

***Reason* (MQLONG) – output**

Reason code qualifying *CompCode*.

The following reason codes indicating error and warning conditions can be returned from the mqSetInteger call:

MQRC_HBAG_ERROR

Bag handle not valid.

MQRC_INDEX_ERROR

Index not valid (index negative and not MQIND_NONE or MQIND_ALL).

MQRC_INDEX_NOT_PRESENT

No item with the specified index is present within the bag for the selector given.

MQRC_MULTIPLE_INSTANCE_ERROR

Multiple instances of system selector not valid.

MQRC_SELECTOR_NOT_PRESENT

No item with the specified selector is present within the bag.

MQRC_SELECTOR_NOT_SUPPORTED

Specified system selector not supported by the MQAI.

MQRC_SELECTOR_NOT_UNIQUE

MQIND_NONE specified when more than one occurrence of the specified selector is present in the bag.

MQRC_SELECTOR_OUT_OF_RANGE

Selector not in valid range for call.

MQRC_SELECTOR_WRONG_TYPE

Data item has wrong datatype for call.

MQRC_STORAGE_NOT_AVAILABLE

Insufficient storage available.

MQRC_SYSTEM_BAG_NOT_ALTERABLE

System bag cannot be altered or deleted.

MQRC_SYSTEM_ITEM_NOT_ALTERABLE

System item is read only and cannot be altered.

C language invocation

```
mqSetInteger (Bag, Selector, ItemIndex, ItemValue, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG Bag;           /* Bag handle */
MQLONG Selector;     /* Selector */
MQLONG ItemIndex;    /* Item index */
MQLONG ItemValue;    /* Integer value */
MQLONG CompCode;     /* Completion code */
MQLONG Reason;       /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqSetInteger Bag, Selector, ItemIndex, ItemValue, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag      As Long 'Bag handle'
Dim Selector As Long 'Selector'
Dim ItemIndex As Long 'Item index'
Dim ItemValue As Long 'Integer value'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'
```

mqSetInteger64

The mqSetInteger64 call either modifies a 64-bit integer item that is already present in the bag, or deletes all existing occurrences of the specified selector and adds a new occurrence at the end of the bag. The data item is usually a user item, but specific system-data items can also be modified.

Syntax

```
mqSetInteger64 (Bag, Selector, ItemIndex, ItemValue, CompCode, Reason)
```

Parameters

Bag (MQHBAG) – input

Handle of the bag to be set. This must be the handle of a bag created by the user, and not the handle of a system bag;
MQRC_SYSTEM_BAG_NOT_ALTERABLE results if the handle you specify refers to a system bag.

Selector (MQLONG) – input

Selector of the item to be modified. If the selector is less than zero (that is, a system selector), the selector must be one that is supported by the MQAI; MQRC_SELECTOR_NOT_SUPPORTED results if it is not.

If the selector is a supported system selector, but is one that is read-only, MQRC_SYSTEM_ITEM_NOT_ALTERABLE results.

If the selector is an alterable system selector, but is always a single-instance selector and the application attempts to create a second instance in the bag, MQRC_MULTIPLE_INSTANCE_ERROR results.

If the selector is zero or greater (that is, a user selector), and the bag was created with the MQCBO_CHECK_SELECTORS option or as an administration

bag (MQCBO_ADMIN_BAG), the selector must be in the range MQIA_FIRST through MQIA_LAST; MQRC_SELECTOR_OUT_OF_RANGE results if it is not. If MQCBO_CHECK_SELECTORS was not specified, the selector can be any value zero or greater.

If MQIND_ALL is *not* specified for the *ItemIndex* parameter, the specified selector must already be present in the bag;
MQRC_SELECTOR_NOT_PRESENT results if it is not.

If MQIND_ALL is *not* specified for the *ItemIndex* parameter, the datatype of the item must agree with the datatype implied by the call;
MQRC_SELECTOR_WRONG_TYPE results if it is not.

ItemIndex (MQLONG) – input

This value identifies the occurrence of the item with the specified selector that is to be modified. The value must be zero or greater, or one of the special values described below; if it is none of these, MQRC_INDEX_ERROR results.

Zero or greater

The item with the specified index must already be present in the bag;
MQRC_INDEX_NOT_PRESENT results if it is not. The index is counted relative to the items in the bag that have the specified selector. For example, if there are five items in the bag with the specified selector, the valid values for *ItemIndex* are 0 through 4.

MQIND_NONE

This specifies that there must be one occurrence only of the specified selector in the bag. If there is more than one occurrence, MQRC_SELECTOR_NOT_UNIQUE results.

MQIND_ALL

This specifies that all existing occurrences of the specified selector (if any) are to be deleted from the bag, and a new occurrence of the selector created at the end of the bag.

Note: For system selectors, the order is not changed.

ItemValue (MQINT64) – input

The integer value to be placed in the bag.

CompCode (MQLONG) – output

Completion code.

Reason (MQLONG) – output

Reason code qualifying *CompCode*.

The following reason codes indicating error and warning conditions can be returned from the mqSetInteger64 call:

MQRC_HBAG_ERROR

Bag handle not valid.

MQRC_INDEX_ERROR

Index not valid (index negative and not MQIND_NONE or MQIND_ALL).

MQRC_INDEX_NOT_PRESENT

No item with the specified index is present within the bag for the selector given.

MQRC_MULTIPLE_INSTANCE_ERROR

Multiple instances of system selector not valid.

MQRC_SELECTOR_NOT_PRESENT

No item with the specified selector is present within the bag.

MQRC_SELECTOR_NOT_SUPPORTED

Specified system selector not supported by the MQAI.

MQRC_SELECTOR_NOT_UNIQUE

MQIND_NONE specified when more than one occurrence of the specified selector is present in the bag.

MQRC_SELECTOR_OUT_OF_RANGE

Selector not in valid range for call.

MQRC_SELECTOR_WRONG_TYPE

Data item has wrong datatype for call.

MQRC_STORAGE_NOT_AVAILABLE

Insufficient storage available.

MQRC_SYSTEM_BAG_NOT_ALTERABLE

System bag cannot be altered or deleted.

MQRC_SYSTEM_ITEM_NOT_ALTERABLE

System item is read only and cannot be altered.

C language invocation

```
mqSetInteger64 (Bag, Selector, ItemIndex, ItemValue, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG    Bag;          /* Bag handle */
MQLONG    Selector;     /* Selector */
MQLONG    ItemIndex;    /* Item index */
MQINT64   ItemValue;    /* Integer value */
MQLONG    CompCode;     /* Completion code */
MQLONG    Reason;       /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqSetInteger64 Bag, Selector, ItemIndex, ItemValue, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag      As Long 'Bag handle'
Dim Selector As Long 'Selector'
Dim ItemIndex As Long 'Item index'
Dim ItemValue As Long 'Integer value'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'
```

mqSetIntegerFilter

The mqSetIntegerFilter call either modifies an integer filter item that is already present in the bag, or deletes all existing occurrences of the specified selector and adds a new occurrence at the end of the bag. The data item is usually a user item, but specific system-data items can also be modified.

Syntax

Parameters

Bag (MQHBAG) – input

Handle of the bag to be set. This must be the handle of a bag created by the user, and not the handle of a system bag;

MQRC_SYSTEM_BAG_NOT_ALTERABLE results if the handle you specify refers to a system bag.

Selector (MQLONG) – input

Selector of the item to be modified. If the selector is less than zero (that is, a system selector), the selector must be one that is supported by the MQAI; MQRC_SELECTOR_NOT_SUPPORTED results if it is not.

If the selector is a supported system selector, but is one that is read-only, MQRC_SYSTEM_ITEM_NOT_ALTERABLE results.

If the selector is an alterable system selector, but is always a single-instance selector and the application attempts to create a second instance in the bag, MQRC_MULTIPLE_INSTANCE_ERROR results.

If the selector is zero or greater (that is, a user selector), and the bag was created with the MQCBO_CHECK_SELECTORS option or as an administration bag (MQCBO_ADMIN_BAG), the selector must be in the range MQIA_FIRST through MQIA_LAST; MQRC_SELECTOR_OUT_OF_RANGE results if it is not. If MQCBO_CHECK_SELECTORS was not specified, the selector can be any value zero or greater.

If MQIND_ALL is *not* specified for the *ItemIndex* parameter, the specified selector must already be present in the bag; MQRC_SELECTOR_NOT_PRESENT results if it is not.

If MQIND_ALL is *not* specified for the *ItemIndex* parameter, the datatype of the item must agree with the datatype implied by the call; MQRC_SELECTOR_WRONG_TYPE results if it is not.

ItemIndex (MQLONG) – input

This value identifies the occurrence of the item with the specified selector that is to be modified. The value must be zero or greater, or one of the special values described below; if it is none of these, MQRC_INDEX_ERROR results.

Zero or greater

The item with the specified index must already be present in the bag; MQRC_INDEX_NOT_PRESENT results if it is not. The index is counted relative to the items in the bag that have the specified selector. For example, if there are five items in the bag with the specified selector, the valid values for *ItemIndex* are 0 through 4.

MQIND_NONE

This specifies that there must be one occurrence only of the specified selector in the bag. If there is more than one occurrence, MQRC_SELECTOR_NOT_UNIQUE results.

MQIND_ALL

This specifies that all existing occurrences of the specified selector (if any) are to be deleted from the bag, and a new occurrence of the selector created at the end of the bag.

Note: For system selectors, the order is not changed.

***ItemValue* (MQLONG) – input**

The integer condition value to be placed in the bag.

***Operator* (MQLONG) – input**

The integer filter operator to be placed in the bag. Valid operators are of the form MQCFOP_*.

***CompCode* (MQLONG) – output**

Completion code.

***Reason* (MQLONG) – output**

Reason code qualifying *CompCode*.

The following reason codes indicating error and warning conditions can be returned from the mqSetIntegerFilter call:

MQRC_FILTER_OPERATOR_ERROR

Filter operator not valid.

MQRC_HBAG_ERROR

Bag handle not valid.

MQRC_INDEX_ERROR

Index not valid (index negative and not MQIND_NONE or MQIND_ALL).

MQRC_INDEX_NOT_PRESENT

No item with the specified index is present within the bag for the selector given.

MQRC_MULTIPLE_INSTANCE_ERROR

Multiple instances of system selector not valid.

MQRC_SELECTOR_NOT_PRESENT

No item with the specified selector is present within the bag.

MQRC_SELECTOR_NOT_SUPPORTED

Specified system selector not supported by the MQAI.

MQRC_SELECTOR_NOT_UNIQUE

MQIND_NONE specified when more than one occurrence of the specified selector is present in the bag.

MQRC_SELECTOR_OUT_OF_RANGE

Selector not in valid range for call.

MQRC_SELECTOR_WRONG_TYPE

Data item has wrong datatype for call.

MQRC_STORAGE_NOT_AVAILABLE

Insufficient storage available.

MQRC_SYSTEM_BAG_NOT_ALTERABLE

System bag cannot be altered or deleted.

MQRC_SYSTEM_ITEM_NOT_ALTERABLE

System item is read only and cannot be altered.

C language invocation

```
mqSetIntegerFilter (Bag, Selector, ItemIndex, ItemValue, Operator,  
&CompCode, &Reason);
```

Declare the parameters as follows:

```

MQHBAG Bag;           /* Bag handle */
MQLONG Selector;     /* Selector */
MQLONG ItemIndex;    /* Item index */
MQLONG ItemValue;    /* Integer value */
MQLONG Operator;     /* Item operator */
MQLONG CompCode;     /* Completion code */
MQLONG Reason;       /* Reason code qualifying CompCode */

```

Visual Basic invocation

(Supported on Windows only.)

```
mqSetIntegerFilter Bag, Selector, ItemIndex, ItemValue, Operator,
CompCode, Reason
```

Declare the parameters as follows:

```

Dim Bag      As Long 'Bag handle'
Dim Selector As Long 'Selector'
Dim ItemIndex As Long 'Item index'
Dim ItemValue As Long 'Integer value'
Dim Operator  As Long 'Item operator'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'

```

mqSetString

The mqSetString call either modifies a character data item that is already present in the bag, or deletes all existing occurrences of the specified selector and adds a new occurrence at the end of the bag. The data item is usually a user item, but certain system-data items can also be modified.

Syntax

mqSetString (*Bag*, *Selector*, *ItemIndex*, *Bufferlength*, *Buffer*, *CompCode*, *Reason*)

Parameters

Bag (MQHBAG) – input

Handle of the bag to be set. This must be the handle of a bag created by the user, not the handle of a system bag;

MQRC_SYSTEM_BAG_NOT_ALTERABLE results if you specify the handle of a system bag.

Selector (MQLONG) – input

Selector of the item to be modified.

If the selector is less than zero (that is, a system selector), the selector must be one that is supported by the MQAI; MQRC_SELECTOR_NOT_SUPPORTED results if it is not.

If the selector is a supported system selector, but is one that is read only, MQRC_SYSTEM_ITEM_NOT_ALTERABLE results.

If the selector is an alterable system selector, but is always a single-instance selector and the application attempts to create a second instance in the bag, MQRC_MULTIPLE_INSTANCE_ERROR results.

If the selector is zero or greater (that is, a user selector), and the bag was created with the MQCBO_CHECK_SELECTORS option or as an administration

bag (MQCBO_ADMIN_BAG), the selector must be in the range MQCA_FIRST through MQCA_LAST; MQRC_SELECTOR_OUT_OF_RANGE results if it is not. If MQCBO_CHECK_SELECTORS was not specified, the selector can be any value zero or greater.

If MQIND_ALL is *not* specified for the *ItemIndex* parameter, the specified selector must already be present in the bag;
MQRC_SELECTOR_NOT_PRESENT results if it is not.

If MQIND_ALL is *not* specified for the *ItemIndex* parameter, the datatype of the item must be the same as the datatype implied by the call;
MQRC_SELECTOR_WRONG_TYPE results if it is not.

ItemIndex (MQLONG) – input

This identifies which occurrence of the item with the specified selector is to be modified. The value must be zero or greater, or one of the special values described below; if it is none of these, MQRC_INDEX_ERROR results.

Zero or greater

The item with the specified index must already be present in the bag;
MQRC_INDEX_NOT_PRESENT results if it is not. The index is counted relative to the items in the bag that have the specified selector. For example, if there are five items in the bag with the specified selector, the valid values for *ItemIndex* are 0 through 4.

MQIND_NONE

This specifies that there must be only one occurrence of the specified selector in the bag. If there is more than one occurrence, MQRC_SELECTOR_NOT_UNIQUE results.

MQIND_ALL

This specifies that all existing occurrences of the specified selector (if any) are to be deleted from the bag, and a new occurrence of the selector created at the end of the bag.

BufferLength (MQLONG) – input

The length in bytes of the string contained in the *Buffer* parameter. The value must be zero or greater, or the special value MQBL_NULL_TERMINATED.

If MQBL_NULL_TERMINATED is specified, the string is delimited by the first null encountered in the string.

If MQBL_NULL_TERMINATED is not specified, *BufferLength* characters are inserted into the bag, even if null characters are present; the nulls do not delimit the string.

Buffer (MQCHAR × *BufferLength*) – input

Buffer containing the character string. The length is given by the *BufferLength* parameter. If zero is specified for *BufferLength*, the null pointer can be specified for the address of the *Buffer* parameter; in all other cases, a valid (nonnull) address must be specified for the *Buffer* parameter.

CompCode (MQLONG) – output

Completion code.

Reason (MQLONG) – output

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqSetString call:

MQRC_BUFFER_ERROR

Buffer parameter not valid (invalid parameter address or buffer not completely accessible).

MQRC_BUFFER_LENGTH_ERROR

Buffer length not valid.

MQRC_HBAG_ERROR

Bag handle not valid.

MQRC_INDEX_ERROR

Index not valid (index negative and not MQIND_NONE or MQIND_ALL).

MQRC_INDEX_NOT_PRESENT

No item with the specified index is present within the bag for the selector given.

MQRC_MULTIPLE_INSTANCE_ERROR

Multiple instances of system selector not valid.

MQRC_SELECTOR_NOT_PRESENT

No item with the specified selector is present within the bag.

MQRC_SELECTOR_NOT_SUPPORTED

Specified system selector not supported by the MQAI.

MQRC_SELECTOR_NOT_UNIQUE

MQIND_NONE specified when more than one occurrence of the specified selector is present in the bag.

MQRC_SELECTOR_OUT_OF_RANGE

Selector not within valid range for call.

MQRC_SELECTOR_WRONG_TYPE

Data item has wrong datatype for call.

MQRC_STORAGE_NOT_AVAILABLE

Insufficient storage available.

MQRC_SYSTEM_BAG_NOT_ALTERABLE

System bag cannot be altered or deleted.

MQRC_SYSTEM_ITEM_NOT_ALTERABLE

System item is read-only and cannot be altered.

Usage notes

The Coded Character Set ID (CCSID) associated with this string is copied from the current CCSID of the bag.

C language invocation

```
mqSetString (Bag, Selector, ItemIndex, BufferLength, Buffer,  
&CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG Bag;           /* Bag handle */  
MQLONG Selector;     /* Selector */  
MQLONG ItemIndex;    /* Item index */  
MQLONG BufferLength; /* Buffer length */  
PMQCHAR Buffer;      /* Buffer containing string */  
MQLONG CompCode;     /* Completion code */  
MQLONG Reason;       /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqSetString Bag, Selector, ItemIndex, BufferLength, Buffer,  
CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag As Long 'Bag handle'  
Dim Selector As Long 'Selector'  
Dim ItemIndex As Long 'Item index'  
Dim BufferLength As Long 'Buffer length'  
Dim Buffer As String 'Buffer containing string'  
Dim CompCode As Long 'Completion code'  
Dim Reason As Long 'Reason code qualifying CompCode'
```

mqSetStringFilter

The mqSetStringFilter call either modifies a string filter item that is already present in the bag, or deletes all existing occurrences of the specified selector and adds a new occurrence at the end of the bag. The data item is usually a user item, but certain system-data items can also be modified.

Syntax

`mqSetStringFilter (Bag, Selector, ItemIndex, Bufferlength, Buffer, Operator, CompCode, Reason)`

Parameters

Bag (MQHBAG) – input

Handle of the bag to be set. This must be the handle of a bag created by the user, not the handle of a system bag;

MQRC_SYSTEM_BAG_NOT_ALTERABLE results if you specify the handle of a system bag.

Selector (MQLONG) – input

Selector of the item to be modified.

If the selector is less than zero (that is, a system selector), the selector must be one that is supported by the MQAI; MQRC_SELECTOR_NOT_SUPPORTED results if it is not.

If the selector is a supported system selector, but is one that is read only, MQRC_SYSTEM_ITEM_NOT_ALTERABLE results.

If the selector is an alterable system selector, but is always a single-instance selector and the application attempts to create a second instance in the bag, MQRC_MULTIPLE_INSTANCE_ERROR results.

If the selector is zero or greater (that is, a user selector), and the bag was created with the MQCBO_CHECK_SELECTORS option or as an administration bag (MQCBO_ADMIN_BAG), the selector must be in the range MQCA_FIRST through MQCA_LAST; MQRC_SELECTOR_OUT_OF_RANGE results if it is not. If MQCBO_CHECK_SELECTORS was not specified, the selector can be any value zero or greater.

If MQIND_ALL is *not* specified for the *ItemIndex* parameter, the specified selector must already be present in the bag;
MQRC_SELECTOR_NOT_PRESENT results if it is not.

If MQIND_ALL is *not* specified for the *ItemIndex* parameter, the datatype of the item must be the same as the datatype implied by the call; MQRC_SELECTOR_WRONG_TYPE results if it is not.

ItemIndex (MQLONG) – input

This identifies which occurrence of the item with the specified selector is to be modified. The value must be zero or greater, or one of the special values described below; if it is none of these, MQRC_INDEX_ERROR results.

Zero or greater

The item with the specified index must already be present in the bag; MQRC_INDEX_NOT_PRESENT results if it is not. The index is counted relative to the items in the bag that have the specified selector. For example, if there are five items in the bag with the specified selector, the valid values for *ItemIndex* are 0 through 4.

MQIND_NONE

This specifies that there must be only one occurrence of the specified selector in the bag. If there is more than one occurrence, MQRC_SELECTOR_NOT_UNIQUE results.

MQIND_ALL

This specifies that all existing occurrences of the specified selector (if any) are to be deleted from the bag, and a new occurrence of the selector created at the end of the bag.

BufferLength (MQLONG) – input

The length in bytes of the condition string contained in the *Buffer* parameter. The value must be zero or greater, or the special value MQBL_NULL_TERMINATED.

If MQBL_NULL_TERMINATED is specified, the string is delimited by the first null encountered in the string.

If MQBL_NULL_TERMINATED is not specified, *BufferLength* characters are inserted into the bag, even if null characters are present; the nulls do not delimit the string.

Buffer (MQCHAR × *BufferLength*) – input

Buffer containing the character condition string. The length is given by the *BufferLength* parameter. If zero is specified for *BufferLength*, the null pointer can be specified for the address of the *Buffer* parameter; in all other cases, a valid (nonnull) address must be specified for the *Buffer* parameter.

Operator (MQLONG × *Operator*) – input

String filter operator to be placed in the bag. Valid operators are of the form MQCFOP_*

CompCode (MQLONG) – output

Completion code.

Reason (MQLONG) – output

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqSetStringFilter call:

MQRC_BUFFER_ERROR

Buffer parameter not valid (invalid parameter address or buffer not completely accessible).

MQRC_BUFFER_LENGTH_ERROR

Buffer length not valid.

MQRC_FILTER_OPERATOR_ERROR	Bag handle not valid.
MQRC_HBAG_ERROR	Bag handle not valid.
MQRC_INDEX_ERROR	Index not valid (index negative and not MQIND_NONE or MQIND_ALL).
MQRC_INDEX_NOT_PRESENT	No item with the specified index is present within the bag for the selector given.
MQRC_MULTIPLE_INSTANCE_ERROR	Multiple instances of system selector not valid.
MQRC_SELECTOR_NOT_PRESENT	No item with the specified selector is present within the bag.
MQRC_SELECTOR_NOT_SUPPORTED	Specified system selector not supported by the MQAI.
MQRC_SELECTOR_NOT_UNIQUE	MQIND_NONE specified when more than one occurrence of the specified selector is present in the bag.
MQRC_SELECTOR_OUT_OF_RANGE	Selector not within valid range for call.
MQRC_SELECTOR_WRONG_TYPE	Data item has wrong datatype for call.
MQRC_STORAGE_NOT_AVAILABLE	Insufficient storage available.
MQRC_SYSTEM_BAG_NOT_ALTERABLE	System bag cannot be altered or deleted.
MQRC_SYSTEM_ITEM_NOT_ALTERABLE	System item is read-only and cannot be altered.

Usage notes

The Coded Character Set ID (CCSID) associated with this string is copied from the current CCSID of the bag.

C language invocation

```
mqSetStringFilter (Bag, Selector, ItemIndex, BufferLength, Buffer,
Operator, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG Bag;           /* Bag handle */
MQLONG Selector;     /* Selector */
MQLONG ItemIndex;    /* Item index */
MQLONG BufferLength; /* Buffer length */
PMQCHAR Buffer;      /* Buffer containing string */
MQLONG Operator;     /* Item operator */
MQLONG CompCode;     /* Completion code */
MQLONG Reason;       /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqSetStringFilter Bag, Selector, ItemIndex, BufferLength, Buffer,  
Operator, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag As Long 'Bag handle'  
Dim Selector As Long 'Selector'  
Dim ItemIndex As Long 'Item index'  
Dim BufferLength As Long 'Buffer length'  
Dim Buffer As String 'Buffer containing string'  
Dim Operator As Long 'Item operator'  
Dim CompCode As Long 'Completion code'  
Dim Reason As Long 'Reason code qualifying CompCode'
```

mqTrim

The mqTrim call trims the blanks from a blank-padded string, then terminates it with a null.

Syntax

mqTrim (*BufferLength*, *Buffer*, *String*, *CompCode*, *Reason*)

Parameters

BufferLength (MQLONG) – input

Length in bytes of the buffer containing the string padded with blanks. Must be zero or greater.

Buffer (MQCHAR × *BufferLength*) – input

Buffer containing the blank-padded string. The length is given by the *BufferLength* parameter. If zero is specified for *BufferLength*, the null pointer can be specified for the address of the *Buffer* parameter; in all other cases, a valid (nonnull) address must be specified for the *Buffer* parameter.

String (MQCHAR × (*BufferLength*+1)) – output

Buffer to receive the null-terminated string. The length of this buffer must be at least one byte greater than the value of the *BufferLength* parameter.

CompCode (MQLONG) – output

Completion code.

Reason (MQLONG) – output

Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqTrim call:

MQRC_BUFFER_ERROR

Buffer parameter not valid (invalid parameter address or buffer not completely accessible).

MQRC_BUFFER_LENGTH_ERROR

Buffer length not valid.

MQRC_STRING_ERROR

String parameter not valid (invalid parameter address or buffer not completely accessible).

Usage notes

1. If the two buffer pointers are the same, the trimming is done in place. If they are not the same, the blank-padded string is copied into the null-terminated string buffer. After copying, the buffer is scanned backwards from the end until a nonspace character is found. The byte following the nonspace character is then overwritten with a null character.
2. If *String* and *Buffer* partially overlap, the result is undefined.

C language invocation

```
mqTrim (BufferLength, Buffer, String, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQLONG    BufferLength;      /* Buffer length */  
PMQCHAR   Buffer;          /* Buffer containing blank-padded string */  
MQCHAR    String[n+1];     /* String with blanks discarded */  
MQLONG    CompCode;        /* Completion code */  
MQLONG    Reason;          /* Reason code qualifying CompCode */
```

Note: This call is not supported in Visual Basic.

mqTruncateBag

The mqTruncateBag call reduces the number of user items in a user bag to the specified value, by deleting user items from the end of the bag.

Syntax

```
mqTruncateBag (Bag, ItemCount, CompCode, Reason)
```

Parameters

Bag (**MQHBAG**) – input

Handle of the bag to be truncated. This must be the handle of a bag created by the user, not the handle of a system bag; MQRC_SYSTEM_BAG_NOT_ALTERABLE results if you specify the handle of a system bag.

ItemCount (**MQLONG**) – input

The number of user items to remain in the bag after truncation. Zero is a valid value.

Note: The *ItemCount* parameter is the number of data items, not the number of unique selectors. (If there are one or more selectors that occur multiple times in the bag, there will be fewer selectors than data items before truncation.) Data items are deleted from the end of the bag, in the opposite order to which they were added to the bag.

If the number specified exceeds the number of user items currently in the bag, MQRC_ITEM_COUNT_ERROR results.

CompCode (MQLONG) – output
Completion code.

Reason (MQLONG) – output
Reason code qualifying *CompCode*.

The following reason codes indicating error conditions can be returned from the mqTruncateBag call:

MQRC_HBAG_ERROR
Bag handle not valid.

MQRC_ITEM_COUNT_ERROR
ItemCount parameter not valid (value exceeds the number of user data items in the bag).

MQRC_SYSTEM_BAG_NOT_ALTERABLE
System bag cannot be altered or deleted.

Usage notes

1. System items in a bag are not affected by mqTruncateBag; the call cannot be used to truncate system bags.
2. mqTruncateBag with an *ItemCount* of zero is not the same as the mqClearBag call. The former deletes all of the user items but leaves the system items intact, and the latter deletes all of the user items and resets the system items to their initial values.

C language invocation

```
mqTruncateBag (Bag, ItemCount, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHBAG hBag;          /* Bag handle */  
MQLONG ItemCount;     /* Number of items to remain in bag */  
MQLONG CompCode;      /* Completion code */  
MQLONG Reason;        /* Reason code qualifying CompCode */
```

Visual Basic invocation

(Supported on Windows only.)

```
mqTruncateBag Bag, ItemCount, CompCode, Reason
```

Declare the parameters as follows:

```
Dim Bag      As Long 'Bag handle'  
Dim ItemCount As Long 'Number of items to remain in bag'  
Dim CompCode As Long 'Completion code'  
Dim Reason   As Long 'Reason code qualifying CompCode'
```

MQAI Selectors

Items in bags are identified by a *selector* that acts as an identifier for the item. There are two types of selector, *user selector* and *system selector*.

User selectors

User selectors have values that are zero or positive. For the administration of MQSeries objects, valid user selectors are already defined by the following constants:

- MQCA_* and MQIA_*
- object attributes
- MQCACF_* and MQIACF_* (items relating specifically to PCF)
- MQCACH_* and MQIACH_* (channel attributes)

For user messages, the meaning of a user selector is defined by the application.

The following additional user selectors are introduced by the MQAI:

MQIACF_INQUIRY

Identifies an WebSphere MQ object attribute to be returned by an Inquire command.

MQHA_BAG_HANDLE

Identifies a bag handle residing within another bag.

MQHA_FIRST

Lower limit for handle selectors.

MQHA_LAST

Upper limit for handle selectors.

MQHA_LAST_USED

Upper limit for last handle selector allocated.

MQCA_USER_LIST

Default user selector. Supported on Visual Basic only. This selector supports character type and represents the default value used if the *Selector* parameter is omitted on the mqAdd*, mqSet*, or mqInquire* calls.

MQIA_USER_LIST

Default user selector. Supported on Visual Basic only. This selector supports integer type and represents the default value used if the *Selector* parameter is omitted on the mqAdd*, mqSet*, or mqInquire* calls.

System selectors

System selectors have negative values. The following system selectors are included in the bag when it is created:

MQIASY_BAG_OPTIONS

Bag-creation options. A summation of the options used to create the bag. This selector cannot be changed by the user.

MQIASY_CODED_CHAR_SET_ID

Character-set identifier for the character data items in the bag. The initial value is the queue-manager's character set.

The value in the bag is used on entry to the mqExecute call and set on exit from the mqExecute call. This also applies when character strings are added to or modified in the bag.

MQIASY_COMMAND

PCF command identifier. Valid values are the MQCMD_* constants. For user messages, the value MQCMD_NONE should be used. The initial value is MQCMD_NONE.

The value in the bag is used on entry to the mqPutBag and mqBagToBuffer calls, and set on exit from the mqExecute, mqGetBag and mqBufferToBag calls.

MQIASY_COMP_CODE

Completion code. Valid values are the MQCC_* constants. The initial value is MQCC_OK.

The value in the bag is used on entry to the mqExecute, mqPutBag, and mqBagToBuffer calls, and set on exit from the mqExecute, mqGetBag, and mqBufferToBag calls.

MQIASY_CONTROL

PCF control options. Valid values are the MQCFC_* constants. The initial value is MQCFC_LAST.

The value in the bag is used on entry to the mqExecute, mqPutBag, and mqBagToBuffer calls, and set on exit from the mqExecute, mqGetBag, and mqBufferToBag calls.

MQIASY_MSG_SEQ_NUMBER

PCF message sequence number. Valid values are 1 or greater. The initial value is 1.

The value in the bag is used on entry to the mqExecute, mqPutBag, and mqBagToBuffer calls, and set on exit from the mqExecute, mqGetBag, and mqBufferToBag calls.

MQIASY_REASON

Reason code. Valid values are the MQRC_* constants. The initial value is MQRC_NONE.

The value in the bag is used on entry to the mqExecute, mqPutBag, and mqBagToBuffer calls, and set on exit from the mqExecute, mqGetBag, and mqBufferToBag calls.

MQIASY_TYPE

PCF command type. Valid values are the MQCFT_* constants. For user messages, the value MQCFT_USER should be used. The initial value is MQCFT_USER for bags created as user bags and MQCFT_COMMAND for bags created as administration or command bags.

The value in the bag is used on entry to the mqExecute, mqPutBag, and mqBagToBuffer calls, and set on exit from the mqExecute, mqGetBag, and mqBufferToBag calls.

MQIASY_VERSION

PCF version. Valid values are the MQCFH_VERSION_* constants. The initial value is MQCFH_VERSION_1.

If the value in the bag is set to a value other than MQCFH_VERSION_1, the value is used on entry to the mqExecute, mqPutBag, and mqBagToBuffer calls. If the value in the bag is MQCFH_VERSION_1, the PCF version is the lowest value required for the parameter structures that are present in the message.

The value in the bag is set on exit from the mqExecute, mqGetBag, and mqBufferToBag calls.

Chapter 11. Examples of using the MQAI

This topic includes some example programs that demonstrate use of the MQAI.
The samples perform the following tasks:

1. Create a local queue.
2. Print a list of all local queues and their current depths.
3. Display events on the screen using a simple event monitor.

Creating a local queue (amqsaicq.c)

```
/***********************************************/
/*
/* Program name: AMQSAICQ.C
/*
/* Description: Sample C program to create a local queue using the
/*               WebSphere MQ Administration Interface (MQAI).
/*
/* Statement: Licensed Materials - Property of IBM
/*
/*             84H2000, 5765-B73
/*
/*             84H2001, 5639-B42
/*
/*             84H2002, 5765-B74
/*
/*             84H2003, 5765-B75
/*
/*             84H2004, 5639-B43
/*
/*             (C) Copyright IBM Corp. 1999, 2005
/*
/***********************************************/
/*
/* Function:
/* AMQSAICQ is a sample C program that creates a local queue and is an
/* example of the use of the mqExecute call.
/*
/* - The name of the queue to be created is a parameter to the program.
/*
/* - A PCF command is built by placing items into an MQAI bag.
/* These are:-
/*
/*     - The name of the queue
/*     - The type of queue required, which, in this case, is local.
/*
/* - The mqExecute call is executed with the command MQCMD_CREATE_Q.
/* The call generates the correct PCF structure.
/* The call receives the reply from the command server and formats into
/* the response bag.
/*
/* - The completion code from the mqExecute call is checked and if there
/* is a failure from the command server then the code returned by the
/* command server is retrieved from the system bag that is
/* embedded in the response bag to the mqExecute call.
/*
/* Note: The command server must be running.
/*
/***********************************************/
/*
/* AMQSAICQ has 2 parameters - the name of the local queue to be created
/*                         - the queue manager name (optional)
/*
/***********************************************/
/*********************************************
```

```

/* Includes
 ****
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>

#include <cmqc.h>           /* MQI */          */
#include <cmqfc.h>           /* PCF */          */
#include <cmqbc.h>           /* MQAI */         */

void CheckCallResult(MQCHAR *, MQLONG , MQLONG );
void CreateLocalQueue(MQHCONN, MQCHAR *);

int main(int argc, char *argv[])
{
    MQHCONN hConn;           /* handle to WebSphere MQ connection */
    MQCHAR QMName[MQ_Q_MGR_NAME_LENGTH+1]="" /* default QMgr name */
    MQLONG connReason;       /* MQCONN reason code */
    MQLONG compCode;         /* completion code */
    MQLONG reason;           /* reason code */

    /*
    /* First check the required parameters
    */
    printf("Sample Program to Create a Local Queue\n");
    if (argc < 2)
    {
        printf("Required parameter missing - local queue name\n");
        exit(99);
    }

    /*
    /* Connect to the queue manager
    */
    if (argc > 2)
        strncpy(QMName, argv[2], (size_t)MQ_Q_MGR_NAME_LENGTH);
        MQCONN(QMName, &hConn, &compCode, &connReason);

    /*
    /* Report reason and stop if connection failed
    */
    if (compCode == MQCC_FAILED)
    {
        CheckCallResult("MQCONN", compCode, connReason);
        exit( (int)connReason);
    }

    /*
    /* Call the routine to create a local queue, passing the handle to the
    /* queue manager and also passing the name of the queue to be created.
    */
    CreateLocalQueue(hConn, argv[1]);

    /*
    /* Disconnect from the queue manager if not already connected
    */
    if (connReason != MQRC_ALREADY_CONNECTED)
    {
        MQDISC(&hConn, &compCode, &reason);
        CheckCallResult("MQDISC", compCode, reason);
    }
    return 0;
}
/*

```

```

/*
 * Function: CreateLocalQueue
 * Description: Create a local queue by sending a PCF command to the command
 * server.
 */
/*****************/
/*
 * Input Parameters: Handle to the queue manager
 *                   Name of the queue to be created
 */
/*
 * Output Parameters: None
 */
/*
 * Logic: The mqExecute call is executed with the command MQCMD_CREATE_Q.
 *        The call generates the correct PCF structure.
 *        The default options to the call are used so that the command is sent
 *        to the SYSTEM.ADMIN.COMMAND.QUEUE.
 *        The reply from the command server is placed on a temporary dynamic
 *        queue.
 *        The reply is read from the temporary queue and formatted into the
 *        response bag.
 */
/*
 * The completion code from the mqExecute call is checked and if there
 * is a failure from the command server then the code returned by the
 * command server is retrieved from the system bag that is
 * embedded in the response bag to the mqExecute call.
 */
/*****************/
void CreateLocalQueue(MQHCONN hConn, MQCHAR *qName)
{
    MQLONG reason;                      /* reason code */          */
    MQLONG compCode;                    /* completion code */      */
    MQHBAG commandBag = MQHB_UNUSABLE_HBAG; /* command bag for mqExecute */
    MQHBAG responseBag = MQHB_UNUSABLE_HBAG; /* response bag for mqExecute */
    MQHBAG resultBag;                  /* result bag from mqExecute */ */
    MQLONG mqExecuteCC;                /* mqExecute completion code */ */
    MQLONG mqExecuteRC;                /* mqExecute reason code */  */

    printf("\nCreating Local Queue %s\n\n", qName);

    /* Create a command Bag for the mqExecute call. Exit the function if the
     * create fails.
    */
    mqCreateBag(MQCBO_ADMIN_BAG, &commandBag, &compCode, &reason);
    CheckCallResult("Create the command bag", compCode, reason);
    if (compCode !=MQCC_OK)
        return;

    /* Create a response Bag for the mqExecute call, exit the function if the
     * create fails.
    */
    mqCreateBag(MQCBO_ADMIN_BAG, &responseBag, &compCode, &reason);
    CheckCallResult("Create the response bag", compCode, reason);
    if (compCode !=MQCC_OK)
        return;

    /* Put the name of the queue to be created into the command bag. This will */
    /* be used by the mqExecute call. */
    mqAddString(commandBag, MQCA_Q_NAME, MQBL_NULL_TERMINATED, qName, &compCode,
                &reason);
    CheckCallResult("Add q name to command bag", compCode, reason);

    /* **** */
}

```

```

/* Put queue type of local into the command bag. This will be used by the */
/* mqExecute call. */
/*********************************************
mqAddInteger(commandBag, MQIA_Q_TYPE, MQQT_LOCAL, &compCode, &reason);
CheckCallResult("Add q type to command bag", compCode, reason);

/*********************************************
/* Send the command to create the required local queue. */
/* The mqExecute call will create the PCF structure required, send it to */
/* the command server and receive the reply from the command server into */
/* the response bag. */
/*********************************************
mqExecute(hConn, /* WebSphere MQ connection handle */ */
          MQCMD_CREATE_Q, /* Command to be executed */ */
          MQHB_NONE, /* No options bag */ */
          commandBag, /* Handle to bag containing commands */ */
          responseBag, /* Handle to bag to receive the response*/ */
          MQHO_NONE, /* Put msg on SYSTEM.ADMIN.COMMAND.QUEUE*/ */
          MQHO_NONE, /* Create a dynamic q for the response */ */
          &compCode, /* Completion code from the mqExecute */ */
          &reason); /* Reason code from mqExecute call */ */

if (reason == MQRC_CMD_SERVER_NOT_AVAILABLE)
{
    printf("Please start the command server: <strmqcsv QMgrName>\n")
    MQDISC(&hConn, &compCode, &reason);
    CheckCallResult("MQDISC", compCode, reason);
    exit(98);
}

/*********************************************
/* Check the result from mqExecute call and find the error if it failed. */
/*********************************************
if ( compCode == MQCC_OK )
    printf("Local queue %s successfully created\n", qName);
else
{
    printf("Creation of local queue %s failed: Completion Code = %d
           qName, compCode, reason);
    if (reason == MQRCCF_COMMAND_FAILED)
    {
       /*********************************************
        /* Get the system bag handle out of the mqExecute response bag. */
        /* This bag contains the reason from the command server why the */
        /* command failed. */
       /*********************************************
        mqInquireBag(responseBag, MQHA_BAG_HANDLE, 0, &resultBag, &compCode,
                      &reason);
        CheckCallResult("Get the result bag handle", compCode, reason);

       /*********************************************
        /* Get the completion code and reason code, returned by the command */
        /* server, from the embedded error bag. */
       /*********************************************
        mqInquireInteger(resultBag, MQIASY_COMP_CODE, MQIND_NONE, &mqExecuteCC,
                         &compCode, &reason);
        CheckCallResult("Get the completion code from the result bag",
                       compCode, reason);
        mqInquireInteger(resultBag, MQIASY_REASON, MQIND_NONE, &mqExecuteRC,
                         &compCode, &reason);
        CheckCallResult("Get the reason code from the result bag", compCode,
                       reason);
        printf("Error returned by the command server: Completion code = %d :
               Reason = %d\n", mqExecuteCC, mqExecuteRC);
    }
}

```

```

/*
***** Delete the command bag if successfully created. *****/
if (commandBag != MQHB_UNUSABLE_HBAG)
{
    mqDeleteBag(&commandBag, &compCode, &reason);
    CheckCallResult("Delete the command bag", compCode, reason);
}

/*
***** Delete the response bag if successfully created. *****/
if (responseBag != MQHB_UNUSABLE_HBAG)
{
    mqDeleteBag(&responseBag, &compCode, &reason);
    CheckCallResult("Delete the response bag", compCode, reason);
}
/* end of CreateLocalQueue */

/*
/* Function: CheckCallResult
*/
/*
***** Input Parameters: Description of call
/* Completion code
/* Reason code
*/
/*
/* Output Parameters: None
*/
/*
/* Logic: Display the description of the call, the completion code and the
/* reason code if the completion code is not successful
*/
void CheckCallResult(char *callText, MQLONG cc, MQLONG rc)
{
    if (cc != MQCC_OK)
        printf("%s failed: Completion Code = %d :
               Reason = %d\n", callText, cc, rc);
}

```

Inquiring about queues and printing information (amqsailq.c)

```

/*
/* Program name: AMQSAILQ.C
*/
/*
/* Description: Sample C program to inquire the current depth of the local
/* queues using the WebSphere MQ Administration Interface (MQAI)
*/
/*
/* Statement: Licensed Materials - Property of IBM
*/
/*
/* 84H2000, 5765-B73
/*
/* 84H2001, 5639-B42
/*
/* 84H2002, 5765-B74
/*
/* 84H2003, 5765-B75
/*
/* 84H2004, 5639-B43
/*
/*
/* (C) Copyright IBM Corp. 1999, 2005
*/
/*
/*
/* Function:
/* AMQSAILQ is a sample C program that demonstrates how to inquire
*/

```

```

/*
 *      attributes of the local queue manager using the MQAI interface. In      */
/* particular, it inquires the current depths of all the local queues.      */
/*
 *      - A PCF command is built by placing items into an MQAI administration      */
/* bag.      */
/*      These are:-      */
/*          - The generic queue name "*"      */
/*          - The type of queue required. In this sample we want to      */
/*              inquire local queues.      */
/*          - The attribute to be inquired. In this sample we want the      */
/*              current depths.      */
/*
 *      - The mqExecute call is executed with the command MQCMD_INQUIRE_Q.      */
/*      The call generates the correct PCF structure.      */
/*      The default options to the call are used so that the command is sent      */
/*      to the SYSTEM.ADMIN.COMMAND.QUEUE.      */
/*      The reply from the command server is placed on a temporary dynamic      */
/*      queue.      */
/*      The reply from the MQCMD_INQUIRE_Q command is read from the      */
/*      temporary queue and formatted into the response bag.      */
/*
 *      - The completion code from the mqExecute call is checked and if there      */
/*      is a failure from the command server, then the code returned by      */
/*      command server is retrieved from the system bag that has been      */
/*      embedded in the response bag to the mqExecute call.      */
/*
 *      - If the call is successful, the depth of each local queue is placed      */
/*      in system bags embedded in the response bag of the mqExecute call.      */
/*      The name and depth of each queue is obtained from each of the bags      */
/*      and the result displayed on the screen.      */
/*
 * Note: The command server must be running.      */
/*
***** AMQSAILQ has 1 parameter - the queue manager name (optional) *****/
/*
***** Includes *****/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>

#include <cmqc.h>                      /* MQI */
#include <cmqcfc.h>                     /* PCF */
#include <cmqbc.h>                      /* MQAI */

/*
***** Function prototypes *****/
void CheckCallResult(MQCHAR *, MQLONG , MQLONG);

/*
***** Function: main *****/
int main(int argc, char *argv[])
{
    /*
     * MQAI variables
     */
    MQHCONN hConn;                         /* handle to WebSphere MQ connection */
    MQCHAR qmName[MQ_Q_MGR_NAME_LENGTH+1]=""; /* default QMgr name */
    MQLONG reason;                          /* reason code */

```

```

    MQLONG connReason;           /* MQCONN reason code          */
    MQLONG compCode;            /* completion code             */
    MQHBAG adminBag = MQHB_UNUSABLE_HBAG; /* admin bag for mqExecute   */
    MQHBAG responseBag = MQHB_UNUSABLE_HBAG; /* response bag for mqExecute */
    MQHBAG qAttrsBag;          /* bag containing q attributes */
    MQHBAG errorBag;           /* bag containing cmd server error */
    MQLONG mqExecuteCC;         /* mqExecute completion code  */
    MQLONG mqExecuteRC;         /* mqExecute reason code      */
    MQLONG qNameLength;         /* Actual length of q name    */
    MQLONG qDepth;              /* depth of queue             */
    MQLONG i;                  /* loop counter               */
    MQLONG numberofBags;        /* number of bags in response bag */
    MQCHAR qName[MQ_Q_NAME_LENGTH+1]; /* name of queue extracted from bag*/
}

printf("Display current depths of local queues\n\n");

/*********************************************
/* Connect to the queue manager
/*********************************************
if (argc > 1)
    strncpy(qmName, argv[1], (size_t)MQ_Q_MGR_NAME_LENGTH);
MQCONN(qmName, &hConn, &compCode, &connReason);

/*********************************************
/* Report the reason and stop if the connection failed.
/*********************************************
if (compCode == MQCC_FAILED)
{
    CheckCallResult("Queue Manager connection", compCode, connReason);
    exit( (int)connReason);
}

/*********************************************
/* Create an admin bag for the mqExecute call
/*********************************************
mqCreateBag(MQCBO_ADMIN_BAG, &adminBag, &compCode, &reason);
CheckCallResult("Create admin bag", compCode, reason);

/*********************************************
/* Create a response bag for the mqExecute call
/*********************************************
mqCreateBag(MQCBO_ADMIN_BAG, &responseBag, &compCode, &reason);
CheckCallResult("Create response bag", compCode, reason);

/*********************************************
/* Put the generic queue name into the admin bag
/*********************************************
mqAddString(adminBag, MQCA_Q_NAME, MQBL_NULL_TERMINATED, "*",
            &compCode, &reason);
CheckCallResult("Add q name", compCode, reason);

/*********************************************
/* Put the local queue type into the admin bag
/*********************************************
mqAddInteger(adminBag, MQIA_Q_TYPE, MQQT_LOCAL, &compCode, &reason);
CheckCallResult("Add q type", compCode, reason);

/*********************************************
/* Add an inquiry for current queue depths
/*********************************************
mqAddInquiry(adminBag, MQIA_CURRENT_Q_DEPTH, &compCode, &reason);
CheckCallResult("Add inquiry", compCode, reason);

/*********************************************
/* Send the command to find all the local queue names and queue depths. */
/* The mqExecute call creates the PCF structure required, sends it to      */
/* the command server, and receives the reply from the command server into */

```

```

/* the response bag. The attributes are contained in system bags that are */
/* embedded in the response bag, one set of attributes per bag.           */
/***********************************************************************/
mqExecute(hConn,                                /* WebSphere MQ connection handle      */
          MQCMD_INQUIRE_Q,                /* Command to be executed            */
          MQHB_NONE,                     /* No options bag                  */
          adminBag,                      /* Handle to bag containing commands */
          responseBag,                  /* Handle to bag to receive the response*/
          MQHO_NONE,                     /* Put msg on SYSTEM.ADMIN.COMMAND.QUEUE*/
          MQHO_NONE,                     /* Create a dynamic q for the response */
          &compCode,                      /* Completion code from the mqExecute */
          &reason);                     /* Reason code from mqExecute call   */

/***********************************************************************/
/* Check the command server is started. If not exit.                      */
/***********************************************************************/
if (reason == MQRC_CMD_SERVER_NOT_AVAILABLE)
{
    printf("Please start the command server: <strmqcsv QMgrName>\n");
    MQDISC(&hConn, &compCode, &reason);
    CheckCallResult("Disconnect from Queue Manager", compCode, reason);
    exit(98);
}

/***********************************************************************/
/* Check the result from mqExecute call. If successful find the current   */
/* depths of all the local queues. If failed find the error.             */
/***********************************************************************/
if ( compCode == MQCC_OK )                         /* Successful mqExecute */
{
    /***********************************************************************/
    /* Count the number of system bags embedded in the response bag from the */
    /* mqExecute call. The attributes for each queue are in a separate bag. */
    /***********************************************************************/
    mqCountItems(responseBag, MQHA_BAG_HANDLE, &numberOfBags, &compCode,
                 &reason);
    CheckCallResult("Count number of bag handles", compCode, reason);

    for ( i=0; i<numberOfBags; i++)
    {
        /***********************************************************************/
        /* Get the next system bag handle out of the mqExecute response bag. */
        /* This bag contains the queue attributes                         */
        /***********************************************************************/
        mqInquireBag(responseBag, MQHA_BAG_HANDLE, i, &qAttrsBag, &compCode,
                     &reason);
        CheckCallResult("Get the result bag handle", compCode, reason);

        /***********************************************************************/
        /* Get the queue name out of the queue attributes bag               */
        /***********************************************************************/
        mqInquireString(qAttrsBag, MQCA_Q_NAME, 0, MQ_Q_NAME_LENGTH, qName,
                        &qNameLength, NULL, &compCode, &reason);
        CheckCallResult("Get queue name", compCode, reason);

        /***********************************************************************/
        /* Get the depth out of the queue attributes bag                   */
        /***********************************************************************/
        mqInquireInteger(qAttrsBag, MQIA_CURRENT_Q_DEPTH, MQIND_NONE, &qDepth,
                         &compCode, &reason);
        CheckCallResult("Get depth", compCode, reason);

        /***********************************************************************/
        /* Use mqTrim to prepare the queue name for printing.           */
        /* Print the result.                                         */
        /***********************************************************************/

```

```

        mqTrim(MQ_Q_NAME_LENGTH, qName, qName, &compCode, &reason)
        printf("%4d %-48s\n", qDepth, qName);
    }

else                                /* Failed mqExecute */
{
    printf("Call to get queue attributes failed: Completion Code = %d :
           Reason = %d\n", compCode, reason);

/* **** */
/* If the command fails get the system bag handle out of the mqExecute */
/* response bag. This bag contains the reason from the command server */
/* why the command failed. */
/* **** */
if (reason == MQRCCF_COMMAND_FAILED)
{
    mqInquireBag(responseBag, MQHA_BAG_HANDLE, 0, &errorBag, &compCode,
                  &reason);
    CheckCallResult("Get the result bag handle", compCode, reason);

/* **** */
/* Get the completion code and reason code, returned by the command */
/* server, from the embedded error bag. */
/* **** */
mqInquireInteger(errorBag, MQIASY_COMP_CODE, MQIND_NONE, &mqExecuteCC,
                  &compCode, &reason );
CheckCallResult("Get the completion code from the result bag",
                compCode, reason);
mqInquireInteger(errorBag, MQIASY_REASON, MQIND_NONE, &mqExecuteRC,
                  &compCode, &reason );
CheckCallResult("Get the reason code from the result bag",
                compCode, reason);
printf("Error returned by the command server: Completion Code = %d :
       Reason = %d\n", mqExecuteCC, mqExecuteRC);
}

}

/* **** */
/* Delete the admin bag if successfully created. */
/* **** */
if (adminBag != MQHB_UNUSABLE_HBAG)
{
    mqDeleteBag(&adminBag, &compCode, &reason);
    CheckCallResult("Delete the admin bag", compCode, reason);
}

/* **** */
/* Delete the response bag if successfully created. */
/* **** */
if (responseBag != MQHB_UNUSABLE_HBAG)
{
    mqDeleteBag(&responseBag, &compCode, &reason);
    CheckCallResult("Delete the response bag", compCode, reason);
}

/* **** */
/* Disconnect from the queue manager if not already connected */
/* **** */
if (connReason != MQRC_ALREADY_CONNECTED)
{
    MQDISC(&hConn, &compCode, &reason);
    CheckCallResult("Disconnect from queue manager", compCode, reason);
}
return 0;
}

```

```
*****
/*
* Function: CheckCallResult
*
*****
*/
*
* Input Parameters: Description of call
*                   Completion code
*                   Reason code
*
* Output Parameters: None
*
* Logic: Display the description of the call, the completion code and the
*        reason code if the completion code is not successful
*
*****
void CheckCallResult(char *callText, MQLONG cc, MQLONG rc)
{
    if (cc != MQCC_OK)
        printf("%s failed: Completion Code = %d : Reason = %d\n",
               callText, cc, rc);
}
```

Displaying events using an event monitor (amqsaiem.c)

```
*****
/*
/* Program name: AMQSAIEM.C
/*
/* Description: Sample C program to demonstrate a basic event monitor
/*           using the WebSphere MQ Admin Interface (MQAI).
/* Licensed Materials - Property of IBM
/*
/* 63H9336
/* (c) Copyright IBM Corp. 1999, 2005 All Rights Reserved.
/*
/* US Government Users Restricted Rights - Use, duplication or
/* disclosure restricted by GSA ADP Schedule Contract with
/* IBM Corp.
*****
/*
/* Function:
/* AMQSAIEM is a sample C program that demonstrates how to write a simple
/* event monitor using the mqGetBag call and other MQAI calls.
/*
/* The name of the event queue to be monitored is passed as a parameter
/* to the program. This would usually be one of the system event queues:- */
/*     SYSTEM.ADMIN.QMGR.EVENT      Queue Manager events
/*     SYSTEM.ADMIN.PERFM.EVENT    Performance events
/*     SYSTEM.ADMIN.CHANNEL.EVENT Channel events
/*     SYSTEM.ADMIN.LOGGER.EVENT  Logger events
/*
/* To monitor the queue manager event queue or the performance event queue, */
/* the attributes of the queue manager needs to be changed to enable
/* these events. For more information about this, see Part 1 of the
/* Programmable System Management book. The queue manager attributes can
/* be changed using either MQSC commands or the MQAI interface.
/* Channel events are enabled by default.
/*
/* Program logic
/* Connect to the Queue Manager.
/* Open the requested event queue with a wait interval of 30 seconds.
/* Wait for a message, and when it arrives get the message from the queue
/* and format it into an MQAI bag using the mqGetBag call.
/* There are many types of event messages and it is beyond the scope of
/* this sample to program for all event messages. Instead the program
/* prints out the contents of the formatted bag.
```

```

/*      Loop around to wait for another message until either there is an error */
/*      or the wait interval of 30 seconds is reached.                                */
/*
*****AMQSAIEM has 2 parameters - the name of the event queue to be monitored   */
/*                                         - the queue manager name (optional)          */
/*
*****                                                               */

/*
*****Includes
*****#include <stdio.h>
*****#include <string.h>
*****#include <stdlib.h>
*****#include <ctype.h>

*****#include <cmqc.h>                      /* MQI
*****#include <cmqcfc.h>                     /* PCF
*****#include <cmqbc.h>                      /* MQAI

/*
*****Macros
*****#if MQAT_DEFAULT == MQAT_WINDOWS_NT
*****    #define Int64 "I64"
*****#elif defined(MQ_64_BIT)
*****    #define Int64 "l"
*****#else
*****    #define Int64 "ll"
*****#endif

/*
*****Function prototypes
*****void CheckCallResult(MQCHAR *, MQLONG , MQLONG);
*****void GetQEEvents(MQHCONN, MQCHAR *);
*****int PrintBag(MQHBAG);
*****int PrintBagContents(MQHBAG, int);

/*
*****Function: main
*****int main(int argc, char *argv[])
{
    MQHCONN hConn;                           /* handle to connection           */
    MQCHAR QMName[MQ_Q_MGR_NAME_LENGTH+1] = ""; /* default QM name             */
    MQLONG reason;                          /* reason code                  */
    MQLONG connReason;                     /* MQCONN reason code          */
    MQLONG compCode;                        /* completion code              */

/*
*****First check the required parameters
    printf("Sample Event Monitor (times out after 30 secs)\n");
    if (argc < 2)
    {
        printf("Required parameter missing - event queue to be monitored\n");
        exit(99);
    }

/*
*****Connect to the queue manager
    if (argc > 2)
        strncpy(QMName, argv[2], (size_t)MQ_Q_MGR_NAME_LENGTH);

```

```

MQCONN(QMName, &hConn, &compCode, &connReason);
/*********************************************
/* Report the reason and stop if the connection failed */
/*********************************************
if (compCode == MQCC_FAILED)
{
    CheckCallResult("MQCONN", compCode, connReason);
    exit( (int)connReason);
}

/*********************************************
/* Call the routine to open the event queue and format any event messages */
/* read from the queue. */
/*********************************************
GetQEvents(hConn, argv[1]);

/*********************************************
/* Disconnect from the queue manager if not already connected */
/*********************************************
if (connReason != MQRC_ALREADY_CONNECTED)
{
    MQDISC(&hConn, &compCode, &reason);
    CheckCallResult("MQDISC", compCode, reason);
}

return 0;
}

/*********************************************
/*
/* Function: CheckCallResult
/*
/*********************************************
/*
/* Input Parameters: Description of call
/* Completion code
/* Reason code
/*
/* Output Parameters: None
/*
/* Logic: Display the description of the call, the completion code and the
/* reason code if the completion code is not successful
/*
/*********************************************
void CheckCallResult(char *callText, MQLONG cc, MQLONG rc)
{
    if (cc != MQCC_OK)
        printf("%s failed: Completion Code = %d : Reason = %d\n",
               callText, cc, rc);
}

/*********************************************
/*
/* Function: GetQEvents
/*
/*********************************************
/*
/* Input Parameters: Handle to the queue manager
/* Name of the event queue to be monitored
/*
/* Output Parameters: None
/*
/* Logic: Open the event queue.
/* Get a message off the event queue and format the message into
/* a bag.
*/

```

```

/*
 *      A real event monitor would need to be programmed to deal with      */
/*      each type of event that it receives from the queue. This is      */
/*      outside the scope of this sample, so instead, the contents of      */
/*      the bag are printed.                                              */
/*      The program waits for 30 seconds for an event message and then      */
/*      terminates if no more messages are available.                      */
/*
 ****
void GetQEvents(MQHCONN hConn, MQCHAR *qName)
{
    MQLONG openReason;                                /* MQOPEN reason code      */
    MQLONG reason;                                   /* reason code             */
    MQLONG compCode;                                 /* completion code         */
    MQHOBJ eventQueue;                             /* handle to event queue   */

    MQHBAG eventBag = MQHB_UNUSABLE_HBAG;           /* event bag to receive event msg */
    MQOD od = {MQOD_DEFAULT};                        /* Object Descriptor        */
    MQMD md = {MQMD_DEFAULT};                        /* Message Descriptor       */
    MQGMO gmo = {MQGMO_DEFAULT};                     /* get message options     */
    MQLONG bQueueOK = 1;                            /* keep reading msgs while true */

    /*
    ****
    /* Create an Event Bag in which to receive the event.                  */
    /* Exit the function if the create fails.                               */
    /*
    ****
    mqCreateBag(MQCBO_USER_BAG, &eventBag, &compCode, &reason);
    CheckCallResult("Create event bag", compCode, reason);
    if (compCode != MQCC_OK)
        return;

    /*
    ****
    /* Open the event queue chosen by the user                          */
    /*
    strncpy(od.ObjectName, qName, (size_t)MQ_Q_NAME_LENGTH);
    MQOPEN(hConn, &od, MQOO_INPUT_AS_Q_DEF+MQOO_FAIL_IF QUIESCING, &eventQueue,
           &compCode, &openReason);
    CheckCallResult("Open event queue", compCode, openReason);

    /*
    ****
    /* Set the GMO options to control the action of the get message from the */
    /* queue.                                                               */
    /*
    gmo.WaitInterval = 30000;                         /* 30 second wait for message */
    gmo.Options = MQGMO_WAIT + MQGMO_FAIL_IF QUIESCING + MQGMO_CONVERT;
    gmo.Version = MQGMO_VERSION_2;                    /* Avoid need to reset Message ID */
    gmo.MatchOptions = MQMO_NONE;                   /* and Correlation ID after every */
    /* mqGetBag
    /*
    /* If open fails, we cannot access the queue and must stop the monitor. */
    /*
    if (compCode != MQCC_OK)
        bQueueOK = 0;

    /*
    ****
    /* Main loop to get an event message when it arrives               */
    /*
    while (bQueueOK)
    {
        printf("\nWaiting for an event\n");

        /*
        /* Get the message from the event queue and convert it into the event */
        /* bag.                                                               */
        /*
        mqGetBag(hConn, eventQueue, &md, &gmo, eventBag, &compCode, &reason);

    /*

```

```

/* If get fails, we cannot access the queue and must stop the monitor. */
/*********************************************
if (compCode != MQCC_OK)
{
    bQueueOK = 0;

   /*********************************************
    /* If get fails because no message available then we have timed out, */
    /* so report this, otherwise report an error. */
   /*********************************************
    if (reason == MQRC_NO_MSG_AVAILABLE)
    {
        printf("No more messages\n");
    }
    else
    {
        CheckCallResult("Get bag", compCode, reason);
    }
}

/*********************************************
/* Event message read - Print the contents of the event bag */
/*********************************************
else
{
    if (PrintBag(eventBag) )
        printf("\nError found while printing bag contents\n");

    } /* end of msg found */
} /* end of main loop */
/*********************************************
/* Close the event queue if successfully opened */
/*********************************************
if (openReason == MQRC_NONE)
{
    MQCLOSE(hConn, &eventQueue, MQCO_NONE, &compCode, &reason);
    CheckCallResult("Close event queue", compCode, reason);
}

/*********************************************
/* Delete the event bag if successfully created.
/*********************************************
if (eventBag != MQHB_UNUSABLE_HBAG)
{
    mqDeleteBag(&eventBag, &compCode, &reason);
    CheckCallResult("Delete the event bag", compCode, reason);
}

} /* end of GetQEvents */

/*********************************************
/*
/* Function: PrintBag
/*
/*********************************************
/*
/* Input Parameters: Bag Handle
/*
/* Output Parameters: None
/*
/* Returns:      Number of errors found
/*
/* Logic: Calls PrintBagContents to display the contents of the bag.
/*
/*********************************************
```

int PrintBag(MQHBAG dataBag)

```

{
    int errors;

    printf("\n");
    errors = PrintBagContents(dataBag, 0);
    printf("\n");

    return errors;
}

/*********************************************
/*
/* Function: PrintBagContents
/*
/*********************************************
/*
/* Input Parameters: Bag Handle
/*           Indentation level of bag
/*
/* Output Parameters: None
/*
/* Returns:      Number of errors found
/*
/* Logic: Count the number of items in the bag
/*           Obtain selector and item type for each item in the bag.
/*           Obtain the value of the item depending on item type and display the
/*           index of the item, the selector and the value.
/*           If the item is an embedded bag handle then call this function again
/*           to print the contents of the embedded bag increasing the
/*           indentation level.
/*
/*********************************************
int PrintBagContents(MQHBAG dataBag, int indent)
{
   /*********************************************
/* Definitions
/*********************************************
#define LENGTH 500                      /* Max length of string to be read*/
#define INDENT 4                         /* Number of spaces to indent */
                                         /* embedded bag display */

/*********************************************
/* Variables
/*********************************************
MQLONG itemCount;                     /* Number of items in the bag */
MQLONG itemType;                     /* Type of the item */
int i;                                /* Index of item in the bag */
MQCHAR stringVal[LENGTH+1];           /* Value if item is a string */
MQBYTE byteStringVal[LENGTH];         /* Value if item is a byte string */
MQLONG stringLength;                  /* Length of string value */
MQLONG ccsid;                         /* CCSID of string value */
MQINT32 iValue;                       /* Value if item is an integer */
MQINT64 i64Value;                     /* Value if item is a 64-bit
                                         /* integer */

MQLONG selector;                      /* Selector of item */
MQHBAG bagHandle;                    /* Value if item is a bag handle */
MQLONG reason;                        /* reason code */
MQLONG compCode;                      /* completion code */
MQLONG trimLength;                   /* Length of string to be trimmed */
int errors = 0;                       /* Count of errors found */
char blanks[] = " ";                 /* Blank string used to
                                         /* indent display */

/*********************************************
/* Count the number of items in the bag
/*********************************************

```

```

mqCountItems(dataBag, MQSEL_ALL_SELECTORS, &itemCount, &compCode, &reason);

if (compCode != MQCC_OK)
    errors++;
else
{
    printf("
    printf("
    printf("
}

/************************************************
/* If no errors found, display each item in the bag */
/************************************************
if (!errors)
{
    for (i = 0; i < itemCount; i++)
    {

        /* First inquire the type of the item for each item in the bag */
        mqInquireItemInfo(dataBag,          /* Bag handle           */
                           MQSEL_ANY_SELECTOR, /* Item can have any selector*/
                           i,                  /* Index position in the bag */
                           &selector,          /* Actual value of selector */
                           &itemType,          /* returned by call   */
                           &compCode,          /* Actual type of item */
                           &compCode,          /* returned by call   */
                           &reason);          /* Completion code     */
                           /* Reason Code         */

        if (compCode != MQCC_OK)
            errors++;

        switch(itemType)
        {
        case MQITEM_INTEGER:
            /* Item is an integer. Find its value and display its index,   */
            /* selector and value.                                            */
            mqInquireInteger(dataBag,          /* Bag handle           */
                             MQSEL_ANY_SELECTOR, /* Allow any selector */
                             i,                  /* Index position in the bag */
                             &iValue,            /* Returned integer value */
                             &compCode,          /* Completion code     */
                             &reason);          /* Reason Code         */

            if (compCode != MQCC_OK)
                errors++;
            else
                printf("%.s %2d %4d (%d)\n",
                       indent, blanks, i, selector, iValue);
            break

        case MQITEM_INTEGER64:
            /* Item is a 64-bit integer. Find its value and display its   */
            /* index, selector and value.                                     */
            mqInquireInteger64(dataBag,          /* Bag handle           */
                               MQSEL_ANY_SELECTOR, /* Allow any selector */
                               i,                  /* Index position in the bag */
                               &i64Value,          /* Returned integer value */
                               &compCode,          /* Completion code     */
                               &reason);          /* Reason Code         */
        }
    }
}

```

```

        if (compCode != MQCC_OK)
            errors++;
        else
            printf("%.s %d %d (%"Int64"d)\n",
                   indent, blanks, i, selector, i64Value);
        break;

case MQITEM_STRING:
    /*****
     * Item is a string. Obtain the string in a buffer, prepare
     * the string for displaying and display the index, selector,
     * string and Character Set ID.
     ****/
    mqInquireString(dataBag,          /* Bag handle */
                     MQSEL_ANY_SELECTOR, /* Allow any selector */
                     i,                  /* Index position in the bag */
                     LENGTH,             /* Maximum length of buffer */
                     stringVal,          /* Buffer to receive string */
                     &stringLength,       /* Actual length of string */
                     &ccsid,              /* Coded character set id */
                     &compCode,            /* Completion code */
                     &reason);           /* Reason Code */

    /*****
     * The call can return a warning if the string is too long for
     * the output buffer and has been truncated, so only check
     * explicitly for call failure.
     ****/
    if (compCode == MQCC_FAILED)
        errors++;
    else
    {
        /*****
         * Remove trailing blanks from the string and terminate with
         * a null. First check that the string should not have been
         * longer than the maximum buffer size allowed.
         ****/
        if (stringLength > LENGTH)
            trimLength = LENGTH;
        else
            trimLength = stringLength;
        mqTrim(trimLength, stringVal, stringVal, &compCode, &reason);
        printf("%.s %d %d '%s' %d\n",
               indent, blanks, i, selector, stringVal, ccsid);
    }
    break;

case MQITEM_BYTE_STRING:
    /*****
     * Item is a byte string. Obtain the byte string in a buffer,
     * prepare the byte string for displaying and display the
     * index, selector and string.
     ****/
    mqInquireByteString(dataBag,          /* Bag handle */
                         MQSEL_ANY_SELECTOR, /* Allow any selector */
                         i,                  /* Index position in the bag */
                         LENGTH,             /* Maximum length of buffer */
                         byteStringVal,       /* Buffer to receive string */
                         &stringLength,       /* Actual length of string */
                         &compCode,            /* Completion code */
                         &reason);           /* Reason Code */

    /*****
     * The call can return a warning if the string is too long for
     * the output buffer and has been truncated, so only check
     */

```

```

/* explicitly for call failure. */
/*****************************************/
if (compCode == MQCC_FAILED)
    errors++;
else
{
    printf("%.*s %-2d    %-4d    X'",
           indent, blanks, i, selector);

    for (i = 0 ; i < stringLength ; i++)
        printf(""

    printf("\n");
}
break;

case MQITEM_BAG:
/*****************************************/
/* Item is an embedded bag handle, so call the PrintBagContents*/
/* function again to display the contents. */
/*****************************************/
mqInquireBag(dataBag,          /* Bag handle          */
              MQSEL_ANY_SELECTOR, /* Allow any selector */
              i,                  /* Index position in the bag */
              &bagHandle,         /* Returned embedded bag hdle*/
              &compCode,          /* Completion code      */
              &reason);          /* Reason Code          */

if (compCode != MQCC_OK)
    errors++;
else
{
    printf("%.*s %-2d    %-4d    (%d)\n", indent, blanks, i,
           selector, bagHandle);
    if (selector == MQHA_BAG_HANDLE)
        printf(""
    else
        printf(
            PrintBagContents(bagHandle, indent+INDENT));
    }
    break;
}

default:
    printf(""
}
}
}
return errors;
}

```

Chapter 12. Advanced topics

This topic discusses the following:

- Indexing
- Data conversion
- Use of the message descriptor

Indexing

Each selector and value within a data item in a bag have three associated index numbers:

- The index relative to other items that have the same selector.
- The index relative to the category of selector (user or system) to which the item belongs.
- The index relative to all the data items in the bag (user and system).

This allows indexing by user selectors, system selectors, or both as shown in Figure 14.

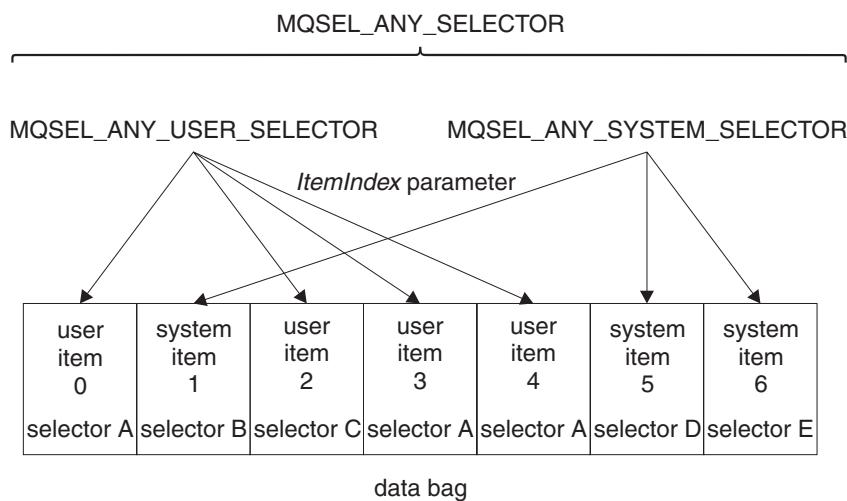


Figure 14. Indexing

In Figure Figure 14, user item 3 (selector A) can be referred to by the following index pairs:

Selector	ItemIndex
selector A	1
MQSEL_ANY_USER_SELECTOR	2
MQSEL_ANY_SELECTOR	3

The index is zero-based like an array in C; if there are 'n' occurrences, the index ranges from zero through 'n-1', with no gaps.

Indexes are used when replacing or removing existing data items from a bag. When used in this way, the insertion order is preserved, but indexes of other data items can be affected. For examples of this, see “Changing information within a bag” on page 470 and “Deleting data items” on page 472.

The three types of indexing allow easy retrieval of data items. For example, if there are three instances of a particular selector in a bag, the mqCountItems call can count the number of instances of that selector, and the mqInquire* calls can specify both the selector and the index to inquire those values only. This is useful for attributes that can have a list of values such as some of the exits on channels.

Data conversion

Like PCF messages, the strings contained in an MQAI data bag can be in a variety of coded character sets. Usually, all of the strings in a PCF message are in the same coded character set; that is, the same set as the queue manager.

Each string item in a data bag contains two values; the string itself and the CCSID. The string that is added to the bag is obtained from the *Buffer* parameter of the mqAddString or mqSetString call. The CCSID is obtained from the system item containing a selector of MQIASY_CODED_CHAR_SET_ID. This is known as the *bag CCSID* and can be changed using the mqSetInteger call.

When you inquire the value of a string contained in a data bag, the CCSID is an output parameter from the call.

Table 15 shows the rules applied when converting data bags into messages and vice versa:

Table 15. CCSID processing

MQAI call	CCSID	Input to call	Output to call
mqBagToBuffer	Bag CCSID (1 on page 591)	Ignored	Unchanged
mqBagToBuffer	String CCSIDs in bag	Used	Unchanged
mqBagToBuffer	String CCSIDs in buffer	Not applicable	Copied from string CCSIDs in bag
mqBufferToBag	Bag CCSID (1 on page 591)	Ignored	Unchanged
mqBufferToBag	String CCSIDs in buffer	Used	Unchanged
mqBufferToBag	String CCSIDs in bag	Not applicable	Copied from string CCSIDs in buffer
mqPutBag	MQMD CCSID	Used	Unchanged (2 on page 591)
mqPutBag	Bag CCSID (1 on page 591)	Ignored	Unchanged
mqPutBag	String CCSIDs in bag	Used	Unchanged
mqPutBag	String CCSIDs in message sent	Not applicable	Copied from string CCSIDs in bag
mqGetBag	MQMD CCSID	Used for data conversion of message	Set to CCSID of data returned (3 on page 591)

Table 15. CCSID processing (continued)

MQAI call	CCSID	Input to call	Output to call
mqGetBag	Bag CCSID (1)	Ignored	Unchanged
mqGetBag	String CCSIDs in message	Used	Unchanged
mqGetBag	String CCSIDs in bag	Not applicable	Copied from string CCSIDs in message
mqExecute	Request-bag CCSID	Used for MQMD of request message (4)	Unchanged
mqExecute	Reply-bag CCSID	Used for data conversion of reply message (4)	Set to CCSID of data returned (3)
mqExecute	String CCSIDs in request bag	Used for request message	Unchanged
mqExecute	String CCSIDs in reply bag	Not applicable	Copied from string CCSIDs in reply message

Notes:

1. Bag CCSID is the system item with selector MQIASY_CODED_CHAR_SET_ID.
2. MQCCSI_Q_MGR is changed to the actual queue manager CCSID.
3. If data conversion is requested, the CCSID of data returned is the same as the output value. If data conversion is not requested, the CCSID of data returned is the same as the message value. Note that no message is returned if data conversion is requested but fails.
4. If the CCSID is MQCCSI_DEFAULT, the queue manager's CCSID is used.

Use of the message descriptor

The PCF command type is obtained from the system item with selector MQIASY_TYPE. When you create your data bag, the initial value of this item is set depending on the type of bag you create:

Table 16. PCF command type

Type of bag	Initial value of MQIASY_TYPE item
MQCBO_ADMIN_BAG	MQCFT_COMMAND
MQCBO_COMMAND_BAG	MQCFT_COMMAND
MQCBO_*	MQCFT_USER

When the MQAI generates a message descriptor, the values used in the *Format* and *MsgType* parameters depend on the value of the system item with selector MQIASY_TYPE as shown in Table 16.

Table 17. Format and MsgType parameters of the MQMD

PCF command type	Format	MsgType
MQCFT_COMMAND	MQFMT_ADMIN	MQMT_REQUEST
MQCFT_REPORT	MQFMT_ADMIN	MQMT_REPORT
MQCFT_RESPONSE	MQFMT_ADMIN	MQMT_REPLY
MQCFT_TRACE_ROUTE	MQFMT_ADMIN	MQMT_DATAGRAM

Table 17. Format and MsgType parameters of the MQMD (continued)

PCF command type	Format	MsgType
MQCFT_EVENT	MQFMT_EVENT	MQMT_DATAGRAM
MQCFT_*	MQFMT_PCF	MQMT_DATAGRAM

Table 17 on page 591 shows that if you create an administration bag or a command bag, the *Format* of the message descriptor is MQFMT_ADMIN and the *MsgType* is MQMT_REQUEST. This is suitable for a PCF request message sent to the command server when a response is expected back.

Other parameters in the message descriptor take the values shown in Table 18.

Table 18. Message descriptor values

Parameter	Value
<i>StrucId</i>	MQMD_STRUC_ID
<i>Version</i>	MQMD_VERSION_1
<i>Report</i>	MQRO_NONE
<i>MsgType</i>	see Table 17 on page 591
<i>Expiry</i>	30 seconds (note 1)
<i>Feedback</i>	MQFB_NONE
<i>Encoding</i>	MQENC_NATIVE
<i>CodedCharSetId</i>	depends on the bag CCSID (note 2)
<i>Format</i>	see Table 17 on page 591
<i>Priority</i>	MQPRI_PRIORITY_AS_Q_DEF
<i>Persistence</i>	MQPER_NOT_PERSISTENT
<i>MsgId</i>	MQML_NONE
<i>CorelId</i>	MQCI_NONE
<i>BackoutCount</i>	0
<i>ReplyToQ</i>	see note 3
<i>ReplyToQMgr</i>	blank

Notes:

1. This value can be overridden on the mqExecute call by using the *OptionsBag* parameter. For information about this, see “mqExecute” on page 514.
2. See “Data conversion” on page 590.
3. Name of the user-specified reply queue or MQAI-generated temporary dynamic queue for messages of type MQMT_REQUEST. Blank otherwise.

Part 3. Appendixes

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.

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.

COPYRIGHT LICENSE:

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

Trademarks

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

AIX	AS/400	CICS
DB2	IMS	RACF
IBM	IBMLink™	iSeries
MQSeries	MVS/ESA™	OS/2
OS/390®	OS/400	Presentation Manager
S/390	VTAM	WebSphere
z/OS	zSeries	

Lotus[®] and LotusScript 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.

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

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

Intel[®] is a registered trademark of Intel Corporation in the United States, other countries, or both. (For a complete list of Intel trademarks, see www.intel.com/tradmarx.htm.)

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 trademarks or service marks of others.

Index

A

AccountingConnOverride parameter
 Change Queue Manager
 command 100
 Inquire Queue Manager (Response)
 command 300

AccountingInterval parameter
 Change Queue Manager
 command 100
 Inquire Queue Manager (Response)
 command 300

Action parameter, Reset Cluster
 command 376

Action parameter, Reset Queue Manager
 command 377

ActiveChannels parameter
 Inquire Channel Initiator
 (Response) 190

ActiveChannelsMax parameter
 Inquire Channel Initiator
 (Response) 190

ActiveChannelsPaused parameter
 Inquire Channel Initiator
 (Response) 190

ActiveChannelsRetrying parameter
 Inquire Channel Initiator
 (Response) 190

ActiveChannelsStarted parameter
 Inquire Channel Initiator
 (Response) 190

ActiveChannelsStopped parameter
 Inquire Channel Initiator
 (Response) 190

ActivityRecording parameter
 Change Queue Manager
 command 100
 Inquire Queue Manager (Response)
 command 300

Adapter parameter
 Change, Copy, Create Channel
 Listener command 71
 Inquire Channel Listener (Response)
 command 195
 Inquire Channel Listener Status
 (Response) command 200

AdaptersMax parameter
 Inquire Channel Initiator
 (Response) 190

AdaptersStarted parameter
 Inquire Channel Initiator
 (Response) 191

adding 64-bit integer items 469
adding byte string filter items 469
adding byte string items 469
adding character-string items 469
adding data items to bags 469
adding inquiry command 469
adding integer filter items 469
adding integer items 469
adding string filter items 469

AdminBag parameter, mqExecute
 call 515

administration bag 467

AdminQ parameter, mqExecute call 516

AdoptNewMCACheck parameter
 Change Queue Manager
 command 101
 Inquire Queue Manager (Response)
 command 301

AdoptNewMCAType parameter
 Change Queue Manager
 command 101
 Inquire Queue Manager (Response)
 command 301

advanced topics
 data conversion 590
 indexing 589

AllocPrimary parameter
 Inquire Archive (Response) 149
 Set Archive command 386

AllocSecondary parameter
 Inquire Archive (Response) 149
 Set Archive command 386

AllocUnits parameter
 Inquire Archive (Response) 149
 Set Archive command 387

AlterationDate parameter
 Inquire Authentication Information
 Object (Response) command 154
 Inquire CF Structure (Response) 167
 Inquire Channel (Response)
 command 181
 Inquire Channel Listener (Response)
 command 195
 Inquire Cluster Queue Manager
 (Response) command 231
 Inquire Namelist (Response)
 command 262
 Inquire Process (Response)
 command 268
 Inquire Queue (Response)
 command 281
 Inquire Queue Manager (Response)
 command 301
 Inquire Service (Response)
 command 341
 Inquire Storage Class (Response) 349

AlterationTime parameter
 Inquire Authentication Information
 Object (Response) command 155
 Inquire CF Structure (Response) 167
 Inquire Channel (Response)
 command 181
 Inquire Channel Listener (Response)
 command 195
 Inquire Cluster Queue Manager
 (Response) command 231
 Inquire Namelist (Response)
 command 263
 Inquire Process (Response)
 command 268

AlterationTime parameter (*continued*)
 Inquire Queue (Response)
 command 281
 Inquire Queue Manager (Response)
 command 301
 Inquire Service (Response)
 command 341
 Inquire Storage Class (Response) 349
amqsacq.c, sample programs 571
amqsaiem.c, sample programs 580
amqsailq.c, sample programs 575
ApplId parameter
 Change, Copy, Create Process
 command 77
 Inquire Process (Response)
 command 269

ApplTag parameter
 Inquire Connection (Response) 244
 Inquire Queue Status (Response)
 command 332

ApplType parameter
 Change, Copy, Create Process
 command 77
 Inquire Connection (Response) 244
 Inquire Process (Response)
 command 269
 Inquire Queue Status (Response)
 command 332

ArchivePrefix1 parameter
 Inquire Archive (Response) 149
 Set Archive command 387

ArchivePrefix2 parameter
 Inquire Archive (Response) 149
 Set Archive command 387

ArchiveRetention parameter
 Inquire Archive (Response) 149
 Set Archive command 387

ArchiveUnit1 parameter
 Inquire Archive (Response) 149
 Set Archive command 387

ArchiveUnit2 parameter
 Inquire Archive (Response) 149
 Set Archive command 387

ArchiveWTOR parameter
 Inquire Archive (Response) 150
 Set Archive command 387

ASId parameter
 Inquire Queue Status (Response)
 command 333

ASID parameter
 Inquire Connection (Response) 244

AuthInfoAttrs parameter
 Inquire authentication information
 command 152

AuthInfoConnName parameter
 Inquire Authentication Information
 Object (Response) command 155

AuthInfoConnName, Create and Copy
 authentication information
 command 36

AuthInfoConnName, Create authentication information command 36
AuthInfoDesc parameter
 Inquire Authentication Information Object (Response) command 155
AuthInfoDesc, Create authentication information command 36
AuthInfoName parameter
 Change, Copy, Create authentication information command 35
 Change, Create authentication information command 36
 Delete Authentication Information Object 132
 Inquire Authentication Information Object (Response) command 155
 Inquire Authentication Information Object command 152
 Inquire Authentication Information Object Names command 156
AuthInfoNames parameter
 Inquire Authentication Information Object Names (Response) 158
AuthInfoType parameter
 Change, Copy, Create authentication information command 36
 Inquire Authentication Information Object (Response) command 155
authority checking (PCF)
 Compaq NonStop Kernel 20
 HP OpenVMS 20
 iSeries 18
 UNIX systems 19
 Windows NT 19
 z/OS systems 23
AuthorityAdd parameter
 Set Authority Record 391, 392
AuthorityEvent parameter
 Change Queue Manager command 101
 Inquire Queue Manager (Response) command 301
 Inquire Queue Manager Status (Response) command 321
AuthorizationList parameter
 Inquire Authority Records (Response) 162
 Inquire Entity Authority (Response) 251

B

Backlog parameter
 Change, Copy, Create Channel Listener command 71
 Inquire Channel Listener (Response) command 195
 Inquire Channel Listener Status (Response) command 200
BackoutRequeueName parameter
 Change, Copy, Create Queue command 83
 Inquire Queue (Response) command 281

BackoutThreshold parameter
 Change, Copy, Create Queue command 83
 Inquire Queue (Response) command 281
Backup CF Structure 34
BackupDate parameter
 Inquire CF Structure Status (Response) 171
BackupEndRBA parameter
 Inquire CF Structure Status (Response) 171
BackupSize parameter
 Inquire CF Structure Status (Response) 171
BackupStartRBA parameter
 Inquire CF Structure Status (Response) 172
BackupTime parameter
 Inquire CF Structure Status (Response) 172
Bag parameter
 mqAddBag call 484
 mqAddByteString call 486
 mqAddByteStringFilter call 487
 mqAddInteger call 492
 mqAddInteger64 call 493
 mqAddIntegerFilter call 495
 mqAddString call 496
 mqAddStringFilter call 498
 mqClearBag call 504
 mqCountItems call 505
 mqCreateBag call 510
 mqDeleteBag call 511
 mqGetBag call 519
 mqInquireBag call 520
 mqInquireByteString call 523
 mqInquireByteStringFilter call 526
 mqInquireInteger call 529
 mqInquireInteger64 call 531
 mqInquireIntegerFilter call 533
 mqInquireItemInfo call 535
 mqInquireString call 538
 mqInquireStringFilter call 541
 mqPutBag call 546
 mqSetByteString call 547
 mqSetByteStringFilter call 550
 mqSetInteger call 553
 mqSetInteger64 call 555
 mqSetIntegerFilter call 558
 mqSetString call 560
 mqSetStringFilter call 563
 mqTruncateBag call 567
bags
 adding 64-bit integer items to 469
 adding byte string filter items to 469
 adding byte string items to 469
 adding character-string items to 469
 adding data items to 469
 adding inquiry command to 469
 adding integer filter items to 469
 adding integer items to 469
 adding string filter items to 469
 changing 64-bit integer items within 471
 changing byte string filter items within 471
bags (continued)
 changing byte string items within 471
 changing character-string items within 471
 changing information within 470
 changing integer filter items within 471
 changing integer items within 471
 changing string filter items within 471
 converting 479
 converting to PCF messages 479
 creating 467
 creating and deleting 467
 deleting 468
 inquiring within 473
 putting 480
 receiving 480
 types of 467
 using 467
BaseQName parameter
 Change, Copy, Create Queue command 83
 Inquire Queue (Response) command 282
Batch Heartbeat parameter
 Channel commands 44
 Inquire Channel (Response) command 181
 Inquire Cluster Queue Manager (Response) command 231
Batches parameter, Inquire Channel Status (Response) command 217
BatchInterval parameter
 Channel commands 45
 Inquire Channel (Response) command 182
 Inquire Cluster Queue Manager (Response) command 231
BatchSize parameter
 Channel commands 45
 Inquire Channel (Response) command 182
 Inquire Channel Status (Response) command 217
 Inquire Cluster Queue Manager (Response) command 231
BatchSizeIndicator parameter
 Inquire Channel Status (Response) command 217
BlockSize parameter
 Inquire Archive (Response) 150
 Set Archive command 388
BridgeEvent parameter
 Change Queue Manager command 101
 Inquire Queue Manager (Response) command 301
Buffer parameter
 mqAddByteString call 486
 mqAddByteStringFilter call 488
 mqAddString call 497
 mqAddStringFilter call 499
 mqBagToBuffer call 501
 mqBufferToBag call 503
 mqInquireByteString call 524

Buffer parameter (*continued*)
 mqInquireByteStringFilter call 527
 mqInquireString call 539
 mqInquireStringFilter call 542
 mqPad call 544
 mqSetByteString call 548
 mqSetByteStringFilter call 551
 mqSetString call 561
 mqSetStringFilter call 564
 mqTrim call 566
 BufferLength parameter
 mqAddByteString call 486
 mqAddByteStringFilter call 488
 mqAddString call 497
 mqAddStringFilter call 499
 mqBagToBuffer call 501
 mqBufferToBag call 503
 mqInquireByteString call 524
 mqInquireByteStringFilter call 527
 mqInquireString call 539
 mqInquireStringFilter call 542
 mqPad call 544
 mqSetByteStringFilter call 551
 mqSetString call 561
 mqSetStringFilter call 564
 mqTrim call 566
 BufferPoolId parameter
 Inquire Usage (Response) 358, 359
 BuffersReceived parameter, Inquire
 Channel Status (Response)
 command 217
 BuffersSent parameter, Inquire Channel
 Status (Response) command 217
 BytesReceived parameter, Inquire
 Channel Status (Response)
 command 217
 BytesSent parameter, Inquire Channel
 Status (Response) command 217
 ByteStringFilterCommand parameter
 Inquire Connection command 240
 Inquire Queue Status command 325
 ByteStringLength parameter,
 mqInquireByteString call 524
 ByteStringLength parameter,
 mqInquireByteStringFilter call 527

C

calls
 data-bag manipulation 483
 detailed description
 mqAddBad 484
 mqAddByteString 485
 mqAddByteStringFilter 487
 mqAddInquiry 489
 mqAddInteger 491
 mqAddInteger64 493
 mqAddIntegerFilter 494
 mqAddString 496
 mqAddStringFilter 498
 mqBagToBuffer 500
 mqBufferToBag 503
 mqClearBag 504
 mqCountItems 505
 mqCreateBag 507
 mqDeleteBag 511
 mqDeleteItem 512

calls (*continued*)
 detailed description (*continued*)
 mqExecute 514
 mqGetBag 518
 mqInquireBag 520
 mqInquireByteString 523
 mqInquireByteStringFilter 525
 mqInquireInteger 528
 mqInquireInteger64 531
 mqInquireIntegerFilter 533
 mqInquireItemInfo 535
 mqInquireString 538
 mqInquireStringFilter 541
 mqPad 544
 mqPutBag 545
 mqSetByteString 547
 mqSetByteStringFilter 550
 mqSetInteger 553
 mqSetInteger64 555
 mqSetIntegerFilter 557
 mqSetString 560
 mqSetStringFilter 563
 mqTrim 566
 mqTruncateBag 567
 mqAddByteString 469
 mqAddByteStringFilter 469
 mqAddInquiry 469
 mqAddInteger 469
 mqAddInteger64 469
 mqAddIntegerFilter 469
 mqAddString 469
 mqAddStringFilter 469
 mqBagToBuffer 479
 mqBufferToBag 479
 mqClearBag 472
 mqCreateBag 467
 mqDeleteBag 468
 mqDeleteItem 472
 mqExecute 475
 mqGetBag 480
 mqPutBag 480
 mqSetByteString 471
 mqSetByteStringFilter 471
 mqSetInteger 471
 mqSetInteger64 471
 mqSetIntegerFilter 471
 mqSetString 471
 mqSetStringFilter 471
 mqTruncateBag 473
 Catalog parameter
 Inquire Archive (Response) 150
 Set Archive command 388
 CFLevel parameter
 Copy, Change, Create CF Structure
 command 39
 Inquire CF Structure (Response) 168
 CFMsgIdentifier parameter
 Inquire Group (Response) 255
 CFStatusType parameter
 Inquire CF Structure Status
 (Response) 172
 Inquire CF Structure Status
 command 170
 CFStrucAttrs parameter
 Inquire CF Structure command 166

CFStrucDesc parameter
 Copy, Change, Create CF Structure
 command 40
 Inquire CF Structure (Response) 168
 CFStrucName parameter
 Backup CF Structure command 34
 Change, Copy, Create CF Structure
 command 39
 Delete CF Structure command 136
 Inquire CF Structure (Response) 168
 Inquire CF Structure command 166
 Inquire CF Structure Names
 command 169
 Inquire CF Structure Status
 (Response) 172
 Inquire CF Structure Status
 command 170
 Recover CF Structure command 366
 CFStrucNames parameter
 Inquire CF Structure Names
 (Response) 169
 CFStructure parameter
 Change, Copy, Create Queue
 command 84
 Inquire Queue (Response)
 command 282
 Inquire Queue command 272
 CFStrucType parameter
 Inquire CF Structure Status
 (Response) 172, 173
 Change Queue Manager 99
 Change Security 124
 Change, Copy and Create Channel 41
 Change, Copy, Create authentication
 information Object 35
 Change, Copy, Create CF structure 38
 Change, Copy, Create Channel
 Listener 70
 Change, Copy, Create Namelist 73
 Change, Copy, Create Process 76
 Change, Copy, Create Queue 80
 Change, Copy, Create Service 125
 Change, Copy, Create Storage Class 128
 changing 64-bit integer items within data
 bags 471
 changing byte string filter items within
 data bags 471
 changing byte string items within data
 bags 471
 changing character-string items within
 data bags 471
 changing information within data
 bags 470
 changing integer filter items within data
 bags 471
 changing integer items within data
 bags 471
 changing string filter items within data
 bags 471
 Channel parameter, Inquire Cluster
 Queue Manager command 226
 ChannelAttrs parameter, Inquire Channel
 command 175
 ChannelAutoDef parameter
 Change Queue Manager
 command 102

ChannelAutoDef parameter (*continued*)
 Inquire Queue Manager (Response)
 command 302

ChannelAutoDefEvent parameter
 Change Queue Manager
 command 102

 Inquire Queue Manager (Response)
 command 302

ChannelAutoDefExit parameter
 Change Queue Manager
 command 102

 Inquire Queue Manager (Response)
 command 302

ChannelDesc parameter
 Channel commands 45

 Inquire Channel (Response)
 command 182

 Inquire Cluster Queue Manager
 (Response) command 231

ChannelDisposition parameter
 Inquire Channel Status (Response)
 command 217

 Inquire Channel Status
 command 207

 Ping Channel command 363

 Reset Channel command 374

 Resolve Channel command 382

 Start Channel command 398

 Stop Channel command 406

ChannelEvent parameter
 Change Queue Manager
 command 102

 Inquire Queue Manager (Response)
 command 302

ChannelInitiatorControl parameter
 Change Queue Manager
 command 103

 Inquire Queue Manager (Response)
 command 302

ChannelInitiatorStatus parameter
 Inquire Channel Initiator (Response)
 command 191

 Inquire Queue Manager Status
 (Response) command 320

ChannelInstanceAttrs parameter
 Inquire Channel Status
 command 209

ChannelInstanceType parameter
 Inquire Channel Status (Response)
 command 217

 Inquire Channel Status
 command 213

ChannelMonitoring parameter
 Change Queue Manager
 command 103

 Channel commands 46

 Inquire Channel (Response)
 command 182

 Inquire Channel Status (Response)
 command 218

 Inquire Cluster Queue Manager
 (Response) command 232

 Inquire Queue Manager (Response)
 command 303

ChannelName parameter
 Change and Create Channel
 command 43

 command 43

 Delete Channel command 136

 Inquire Channel Names
 command 202

 Inquire Channel Status (Response)
 command 219

 Inquire Channel Names
 (Response) 204

 Inquire Channel (Response)
 command 183

 Inquire Channel command 178

 Inquire Channel Names
 command 202

 Inquire Channel Status (Response)
 command 219

 Inquire Channel Names
 (Response) 204

 Inquire System (Response) 353

 Set System command 396

ChinitAdapters parameter
 Change Queue Manager
 command 104

ChinitAdapters parameter (*continued*)
 Inquire Queue Manager (Response)
 command 304

ChinitDispatchers parameter
 Change Queue Manager
 command 104

 Inquire Queue Manager (Response)
 command 304

ChinitServiceParm parameter
 Change Queue Manager
 command 104

 Inquire Queue Manager (Response)
 command 304

ChinitTraceAutoStart parameter
 Change Queue Manager
 command 104

 Inquire Queue Manager (Response)
 command 304

ChinitTraceTableSize parameter
 Change Queue Manager
 command 105

 Inquire Queue Manager (Response)
 command 304

Clear Queue 131

clearing a bag 472

ClusterCacheType parameter
 Inquire System (Response) 353

ClusterDate parameter
 Inquire Cluster Queue Manager
 (Response) command 233

 Inquire Queue (Response)
 command 282

ClusterInfo parameter
 Inquire Cluster Queue Manager
 (Response) command 233

 Inquire Queue command 273

ClusterName parameter
 Change, Copy, Create Queue
 command 85

 Channel commands 46

 Inquire Channel (Response)
 command 183

 Inquire Cluster Queue Manager
 (Response) command 233

 Inquire Cluster Queue Manager
 (command 226)

 Inquire Queue (Response)
 command 282

 Inquire queue command 273

 Refresh Cluster command 367

 Reset Cluster command 376

 Resume Queue Manager Cluster
 command 384

 Suspend Queue Manager Cluster
 command 414

ClusterNamelist parameter
 Change, Copy, Create Queue
 command 85

 Channel commands 47

 Inquire Channel (Response)
 command 183

 Inquire Queue (Response)
 command 282

 Inquire Queue command 273

 Resume Queue Manager Cluster
 command 384

ClusterNamelist parameter (*continued*)

 Suspend Queue Manager Cluster command 414

ClusterQMgrAttrs parameter, Inquire Cluster Queue Manager command 227

ClusterQMgrName parameter

 Inquire Cluster Queue Manager command 226

ClusterQType parameter, Inquire Queue (Response) command 282

ClusterSenderMonitoringDefault parameter

 Change Queue Manager command 105

 Inquire Queue Manager (Response) command 304

ClusterSenderStatistics parameter

 Change Queue Manager command 105

 Inquire Queue Manager (Response) command 305

ClusterTime parameter

 Inquire Cluster Queue Manager (Response) command 233

 Inquire Queue (Response) command 282

ClusterWorkloadData parameter

 Change Queue Manager command 106

 Inquire Queue Manager (Response) command 305

ClusterWorkloadExit parameter

 Change Queue Manager command 106

 Inquire Queue Manager (Response) command 305

ClusterWorkloadLength parameter

 Change Queue Manager command 106

 Inquire Queue Manager (Response) command 305

CLWLChannelPriority parameter

 Channel commands 47

 Inquire Channel (Response) command 183

 Inquire Cluster Queue Manager (Response) command 233

CLWLChannelRank parameter

 Channel commands 47

 Inquire Channel (Response) command 183

 Inquire Cluster Queue Manager (Response) command 233

CLWLChannelWeight parameter

 Channel commands 47

 Inquire Channel (Response) command 183

 Inquire Cluster Queue Manager (Response) command 233

CLWLMRUChannels parameter

 Change Queue Manager command 106

 Inquire Queue Manager (Response) command 305

CLWLQueuePriority parameter

 Change, Copy, Create Queue command 85

CLWLQueuePriority parameter (*continued*)

 Inquire Queue (Response) command 282

CLWLQueueRank parameter

 Change, Copy, Create Queue command 85

 Inquire Queue (Response) command 282

CLWLUseQ parameter

 Change Queue Manager command 106

 Change, Copy, Create Queue command 85

 Inquire Queue (Response) command 282

 Inquire Queue Manager (Response) command 305

CodedCharSetId field

 MQCFSF structure 436

 MQCFSL structure 440

 MQCFST structure 443

CodedCharSetId parameter

 Inquire Queue Manager (Response) command 306

 Inquire System (Response) 354

CodedCharSetId parameter, mqInquireString call 539

CodedCharSetId parameter, mqInquireStringFilter call 542

command

 queue 11

 structures 417

command bag 467

command calls

 utility 483

Command field 419

Command parameter, mqExecute call 514

CommandEvent parameter

 Change Queue Manager command 107

 Inquire Queue Manager (Response) command 306

CommandInformation parameter

 Inquire Group (Response) 255

CommandInputQName parameter

 Inquire Queue Manager (Response) command 306

CommandLevel parameter

 Inquire Group (Response) 254

 Inquire Queue Manager (Response) command 306

commands

 rules for naming objects in 17

Commands parameter

 Change, Copy, Create Channel Listener command 71

 Inquire Channel Listener (Response) command 195

 Inquire Channel Listener Status (Response) command 200

CommandScope parameter

 Backup CF Structure command 34

 Change Queue Manager command 107

 Change Security command 124

CommandScope parameter (*continued*)

 Change, Copy, Create Namelist command 73

 Change, Copy, Create Process command 78

 Change, Copy, Create Queue command 86

 Change, Copy, Create Storage Class command 129

 Channel commands 48

 Clear Queue command 131

 Delete Authentication Information Object 133

 Delete Channel command 136

 Delete Namelist 139

 Delete Process command 140

 Delete Queue command 142

 Delete Storage Class command 144

 Inquire Archive command 147

 Inquire Authentication Information Object command 153, 179

 Inquire Authentication Information Object Names command 156, 270

 Inquire Channel Initiator command 189

 Inquire Channel Names command 203

 Inquire Channel Status command 214

 Inquire Cluster Queue Manager command 230

 Inquire Connection command 240

 Inquire Log command 256

 Inquire Namelist command 260

 Inquire Namelist Names command 264

 Inquire Process command 266

 Inquire Queue command 273

 Inquire Queue Manager command 291

 Inquire Queue Names command 322

 Inquire Queue Status command 325

 Inquire Security command 337

 Inquire Storage Class command 347

 Inquire Storage Class Names command 350

 Inquire System command 352

 Inquire Usgae command 357

 Move Queue command 361

 Ping Channel command 363

 Recover CF Structure command 366

 Refresh Cluster command 368

 Refresh Queue Manager command 369

 Refresh Security command 371

 Reset Channel command 374

 Reset Cluster command 376

 Reset Queue Statistics command 379

 Resolve Channel command 381

 Resume Queue Manager Cluster command 384

 Resume Queue Manager command 383

 Reverify Security command 385

 Set Archive command 388

 Set Log command 394

 Set System command 396

CommandScope parameter (*continued*)
 Start Channel command 398
 Start Channel Initiator command 401
 Start Channel Listener command 403
 Stop Channel command 407
 Stop Channel Initiator command 409
 Stop Channel Listener command 410
 Suspend Queue Manager Cluster
 command 415
 Suspend Queue Manager
 command 414
 CommandScope parameter, Create
 authentication information
 command 37
 CommandServerControl parameter
 Change Queue Manager
 command 108, 308
 CommandServerStatus parameter
 Inquire Queue Manager Status
 (Response) command 320
 CommandUserId parameter
 Inquire System (Response) 354
 Compact parameter
 Inquire Archive (Response) 150
 Set Archive command 388
 CompCode field 420
 CompCode parameter
 mqAddBag call 484
 mqAddByteString call 486
 mqAddByteStringFilter call 488
 mqAddInquiry call 490
 mqAddInteger call 492
 mqAddInteger64 call 493
 mqAddIntegerFilter call 495
 mqAddString call 497
 mqAddStringFilter call 499
 mqBagToBuffer call 501
 mqBufferToBag call 503
 mqClearBag call 505
 mqCountItems call 506
 mqCreateBag call 510
 mqDeleteBag call 511
 mqDeleteItem call 513
 mqExecute call 516
 mqGetBag call 519
 mqInquireBag call 521
 mqInquireByteString call 524
 mqInquireByteStringFilter call 527
 mqInquireInteger call 529
 mqInquireInteger64 call 532
 mqInquireIntegerFilter call 534
 mqInquireItemInfo call 537
 mqInquireString call 539
 mqInquireStringFilter call 542
 mqPad call 544
 mqPutBag call 546
 mqSetByteString call 548
 mqSetByteStringFilter call 551
 mqSetInteger call 554
 mqSetInteger64 call 556
 mqSetIntegerFilter call 559
 mqSetString call 561
 mqSetStringFilter call 564
 mqTrim call 566
 mqTruncateBag call 568

CompressionRate parameter
 Inquire Channel Status (Response)
 command 219
 CompressionTimee parameter
 Inquire Channel Status (Response)
 command 219
 concepts and terminology 463
 ConfigurationEvent parameter
 Change Queue Manager
 command 108
 Inquire Queue Manager (Response)
 command 308
 configuring WebSphere MQ 475
 Connname parameter
 Inquire Queue Status (Response)
 command 333
 ConnectionAttrs parameter
 Inquire Connection command 240
 ConnectionCount parameter
 Inquire Queue Manager Status
 (Response) command 320
 ConnectionId parameter
 Inquire Connection (Response) 244
 Inquire Connection command 239
 Stop Connection command 412
 ConnectionName parameter
 Channel commands 48
 Inquire Channel (Response)
 command 183
 Inquire Channel Status (Response)
 command 219
 Inquire Channel Status
 command 214
 Inquire Cluster Queue Manager
 (Response) command 233
 Inquire Connection (Response) 245
 Stop Channel command 408
 ConnectionOptions parameter
 Inquire Connection (Response) 245
 ConnInfoType parameter
 Inquire Connection (Response) 245
 Control field 420
 control Language, i5/OS 8
 converting bags and buffers 479
 converting bags to PCF messages 479
 converting PCF messages to bag
 form 479
 Count field
 MQCFIL structure 431
 MQCFSL structure 440
 counting data items 471
 creating a local queue, sample
 programs 571
 creating data bags 467
 CreationDate parameter, Inquire Queue
 (Response) command 283
 CreationTime parameter, Inquire Queue
 (Response) command 283
 CurrentChannels parameter
 Inquire Channel Initiator
 (Response) 191
 CurrentChannelsLU62 parameter
 Inquire Channel Initiator
 (Response) 191
 CurrentChannelsMax parameter
 Inquire Channel Initiator
 (Response) 191

CurrentChannelsTCP parameter
 Inquire Channel Initiator
 (Response) 191
 CurrentLog parameter
 Inquire Queue Manager Status
 (Response) command 321
 CurrentLUWID parameter, Inquire
 Channel Status (Response)
 command 220
 CurrentMsgs parameter, Inquire Channel
 Status (Response) command 220
 CurrentQDepth parameter
 Inquire Queue Status (Response)
 command 330
 CurrentQDepth parameter, Inquire Queue
 (Response) command 283
 CurrentSequenceNumber parameter,
 Inquire Channel Status (Response)
 command 220

D

data
 exchanging 479
 receiving 479
 response 15
 sending 479
 data bags
 adding 64-bit integer items to 469
 adding byte string filter items to 469
 adding byte string items to 469
 adding character-string items to 469
 adding data items to 469
 adding inquiry command to 469
 adding integer filter items to 469
 adding integer items to 469
 adding string filter items to 469
 changing 64-bit integer items
 within 471
 changing byte string filter items
 within 471
 changing byte string items
 within 471
 changing character-string items
 within 471
 changing information within 470
 changing integer filter items
 within 471
 changing integer items within 471
 changing string filter items
 within 471
 converting 479
 converting to PCF messages 479
 creating 467
 creating and deleting 467
 deleting 468
 inquiring within 473
 putting 480
 receiving 480
 types of 467
 using 467
 data conversion 590
 data items
 counting 471
 deleting 472
 filtering 469
 querying 469

data items (*continued*)

 types of 468

data-bag manipulation calls

 command 483

DataBag parameter

 mqBagToBuffer call 501

DataConversion parameter

 Channel commands 49

Inquire Channel (Response)

 command 183

Inquire Cluster Queue Manager (Response) command 233

DataCount parameter

 Ping Channel command 363

DataLength parameter, mqBagToBuffer call 501

DataSetName parameter

 Inquire Archive (Response) 151

Inquire Log (Response) 258

Inquire Usage (Response) 360

DataSetType parameter

 Inquire Usage (Response) 360

DB2BlobTasks parameter

 Inquire System (Response) 354

DB2ConnectStatus parameter

 Inquire Group (Response) 254

DB2Name parameter

 Inquire Group (Response) 254

Inquire System (Response) 354

DB2Tasks parameter

 Inquire System (Response) 354

DeadLetterQName parameter

 Change Queue Manager command 108

Inquire Queue Manager (Response) command 308

DeallocateInterval parameter

 Inquire Log (Response) 257

Set Log command 395

default structures 417

DefBind parameter

 Inquire Queue (Response) command 283

DefBind parameter,
 Change, Copy, Create Queue command 86

definitions of PCFs 25

DefinitionType parameter

 Change, Copy, Create Queue command 86

Inquire Queue (Response) command 283

DefInputOpenOption parameter

 Change, Copy, Create Queue command 86

Inquire Queue (Response) command 283

DefPersistence parameter

 Change, Copy, Create Queue command 86

Inquire Queue (Response) command 284

DefPriority parameter

 Change, Copy, Create Queue command 87

DefPriority parameter (*continued*)

 Inquire Queue (Response)

EntityName parameter (*continued*)

 Inquire Entity Authority command 249

EntityName parameter (*continued*)

 Inquire Entity Authority (Response) 252

EntityType parameter

 Inquire Authority Records 160

Inquire Authority Records (Response) 163

Inquire Entity Authority 249

Inquire Entity Authority (Response) 252

EntriesMax parameter

 Inquire CF Structure Status (Response) 173

EntriesUsed parameter

 Inquire CF Structure Status (Response) 173

EnvData parameter

 Change, Copy, Create command 79

Inquire Process (Response) command 269

EnvironmentInfo parameter

 Start Channel Initiator command 402

error

 response 14

Escape 146

Escape (Response) 146

EscapeText parameter

 Escape (Response) command 147

Escape command 146

EscapeType parameter

 Escape (Response) command 147

Escape command 146

event monitor, sample programs 580

example

 using PCFs 447

exchanging data 479

ExcludeInterval parameter

 Backup CF Structure command 34

ExitInterval parameter

 Inquire System (Response) 354

ExitTasks parameter

 Inquire System (Response) 354

ExitTime parameter

 Inquire Channel Status (Response) command 220

ExpandCount parameter

 Inquire Usage (Response) 358

ExpandType parameter

 Inquire Usage (Response) 358

ExpiryInterval parameter

 Change Queue Manager command 109

Inquire Queue Manager (Response) command 309

ExternalUOWId parameter

 Inquire Queue Status (Response) command 333

E

enquire local queue attributes 447

EntityName parameter

 Inquire Authority Records 160

Inquire Authority Records (Response) 163

F

Facility parameter

 Resume Queue Manager command 383

Suspend Queue Manager command 413

FailDate parameter
 Inquire CF Structure Status (Response)
 command 173
FailTime parameter
 Inquire CF Structure Status (Response) 173
 filtering data items 469
FilterValue field
 MQCFBF structure 423
 MQCFIF structure 428
 MQCFSF structure 436
FilterValueLength field
 MQCFBF structure 423
 MQCFSF structure 436
Force parameter
 Change Queue Manager command 109
 Change, Copy, Create Queue command 87
Format field 418
 message descriptor 13
FromAuthInfoName, Copy authentication information command 35
FromCFStrucName parameter
 Copy CF Structure command 39
FromChannelName parameter
 Copy Channel command 43
FromListenerName parameter, Copy Channel Listener command 70
FromNamelistName parameter, Copy Namelist command 73
FromProcessName parameter, Copy Process command 76
FromQName parameter
 Move Queue command 360
FromQName parameter, Copy Queue command 82
FromServiceName parameter, Copy Service command 125
FromStorageClassName parameter
 Copy Storage Class command 128
FullLogs parameter
 Inquire Log (Response) 258

G

generic values 18
GenericConnectionId parameter
 Inquire Connection command 239
GetMsgOpts parameter, mqGetBag call 519
group bag 467
GroupNames parameter
 Delete Authority Record 135
 Set Authority Record 393

H

HandleState parameter
 Inquire Connection (Response) 245, 333
HardenGetBackout parameter
 Change, Copy, Create Queue command 88
 Inquire Queue (Response) command 284

Hbag parameter
 mqAddInquiry call 490
 mqDeleteItem call 512
Hconn parameter
 mqExecute call 514
 mqGetBag call 518
 mqPutBag call 545
HeaderCompression parameter
 Channel commands 50
 Inquire Channel (Response) command 184
 Inquire Channel Status (Response) command 220
 Inquire Cluster Queue Manager (Response) command 233
HeartbeatInterval parameter
 Channel commands 50
 Inquire Channel (Response) command 184
 Inquire Channel Status (Response) command 221
 Inquire Cluster Queue Manager (Response) command 234
HighQDepth parameter, Reset Queue Statistics (Response) command 379
Hobj parameter
 mqGetBag call 518
 mqPutBag call 545

I

i5/OS Control Language 8
IGQPutAuthority parameter
 Change Queue Manager command 109
 Inquire Queue Manager (Response) command 309
IGQUserId parameter
 Change Queue Manager command 110
 Inquire Queue Manager (Response) command 310
InboundDisposition parameter
 Inquire Channel Initiator (Response) 191
 Start ChannelListener command 403
 Stop Channel Listener command 411
indexing 589
IndexType parameter
 Change, Copy, Create Queue command 88
 Inquire Queue (Response) command 284
InDoubt parameter, Resolve Channel command 381
InDoubtStatus parameter, Inquire Channel Status (Response) command 221
InhibitEvent parameter
 Change Queue Manager command 110
 Inquire Queue Manager (Response) command 310
InhibitGet parameter
 Change, Copy, Create Queue command 89

InhibitGet parameter (*continued*)
 Inquire Queue (Response) command 284
InhibitPut parameter
 Change, Copy, Create Queue command 89
 Inquire Queue (Response) command 285
InitiationQName parameter
 Change, Copy, Create Queue command 89
 Inquire Queue (Response) command 285
 Start Channel Initiator command 401
InputBufferSize parameter
 Inquire Log (Response) 258
Inquire Archive 147
Inquire Archive (Response) 148
Inquire Authentication Information Object 152
Inquire authentication information object (Response) 154
Inquire Authentication Information Object Names 156
Inquire Authentication Information Object Names (Response) 157
Inquire Authority Records 158
Inquire Authority Records (Response) 161
Inquire Authority Service 164
Inquire Authority Service (Response) 165
Inquire CF Structure 166
Inquire CF Structure (Response) 167
Inquire CF Structure Names 168
Inquire CF Structure Names (Response) 169
Inquire CF Structure Status 169
Inquire CF Structure Status (Response) 171
Inquire Channel 174
Inquire Channel (Response) 181
Inquire Channel Initiator (Response) 190
Inquire Channel Listener 192
Inquire Channel Listener (Response) 195
Inquire Channel Listener Status 197
Inquire Channel Listener Status (Response) 199
Inquire Channel Names 202
Inquire Channel Names (Response) 204
Inquire Channel Status 205
Inquire Channel Status (Response) 216
Inquire Cluster Queue Manager 226
Inquire Cluster Queue Manager (Response) 231
Inquire Connection 239
Inquire Connection (Response) 243
Inquire Entity Authority 248
Inquire Entity Authority (Response) 251
Inquire Group 253
Inquire Group (Response) 253
Inquire Log 255
Inquire Log (Response) 256
Inquire Namelist 259
Inquire Namelist (Response) 262
Inquire Namelist Names 264
Inquire Namelist Names (Response) 265

Inquire Process 266
 Inquire Process (Response) 268
 Inquire Process Names 270
 Inquire Process Names (Response) 271
 Inquire Queue 272
 Inquire Queue (Response) 281
 Inquire Queue Manager 291
 Inquire Queue Manager (Response) 299
 Inquire Queue Manager Status 319
 Inquire Queue ManagerStatus (Response) 320
 Inquire Queue Names 322
 Inquire Queue Names (Response) 324
 Inquire Queue Status 325
 Inquire Queue Status (Response) 330
 Inquire Security 336
 Inquire Security (Response) 337
 Inquire Service 339
 Inquire Service (Response) 340
 Inquire Service Status 342
 Inquire Service Status (Response) 344
 Inquire Storage Class 346
 Inquire Storage Class (Response) 349
 Inquire Storage Class Names 350
 Inquire Storage Class Names (Response) 351
 Inquire System 352
 Inquire System (Response) 353
 Inquire Usage 357
 Inquire Usage (Response) 358
 inquiring queues, sample programs 575
 inquiring within data bags 473
IntegerFilterCommand parameter
 Inquire Authentication Information Object command 153
 Inquire CF Structure command 167
 Inquire CF Structure Status command 170
 Inquire Channel command 179
 Inquire Channel Listener command 193
 Inquire Channel Listener Status command 198
 Inquire Channel Status command 215
 Inquire Cluster Queue Manager command 230
 Inquire Connection command 242, 243
 Inquire Namelist command 260
 Inquire Process command 267
 Inquire Queue command 274
 Inquire Queue Status command 326
 Inquire Service command 339
 Inquire Service Status command 343
 Inquire Storage Class command 347
InterfaceVersion parameter
 Inquire Authority Service (Response) 165
IntraGroupQueuing parameter
 Change Queue Manager command 110
 Inquire Queue Manager (Response) command 310
 introduction 463

IPAddress parameter
 Change, Copy, Create Channel Listener command 71
 Inquire Channel Initiator (Response) 192
 Inquire Channel Listener (Response) command 195
 Inquire Channel Listener Status (Response) command 200
 Start Channel Listener command 403
 Stop Channel Listener command 411
IPAddressVersion parameter
 Change Queue Manager command 111
 Inquire Queue Manager (Response) command 310
ItemCount parameter
 mqCountItems call 506
 mqTruncateBag call 567
ItemIndex parameter
 mqDeleteItem call 513
 mqInquireBag call 521
 mqInquireByteString call 524
 mqInquireByteStringFilter call 526
 mqInquireInteger call 529
 mqInquireInteger64 call 532
 mqInquireIntegerFilter call 534
 mqInquireItemInfo call 536
 mqInquireString call 539
 mqInquireStringFilter call 542
 mqSetByteString call 548
 mqSetByteStringFilter call 551
 mqSetInteger call 554
 mqSetInteger64 call 556
 mqSetIntegerFilter call 559
 mqSetString call 561
 mqSetStringFilter call 564
ItemOperator parameter
 mqAddByteStringFilter call 488
 mqAddStringFilter call 499
items
 counting 471
 deleting 472
 filtering 469
 querying 469
items, types of 468
ItemType parameter
 mqInquireItemInfo call 537
ItemValue parameter
 mqAddBagr call 484
 mqAddInteger call 492
 mqAddInteger64 call 493
 mqAddIntegerFilter call 495
 mqInquireBag call 521
 mqInquireInteger call 529
 mqInquireInteger64 call 532
 mqInquireIntegerFilter call 534
 mqSetInteger call 554
 mqSetInteger64 call 556
 mqSetIntegerFilter call 559

K

KeepAliveInterval parameter
 Channel commands 51
 Inquire Channel (Response) command 184

KeepAliveInterval parameter (*continued*)
 Inquire Cluster Queue Manager (Response) command 234
KeepAliveInterval parameter, Inquire Channel Status (Response) command 221

L

LastGetDate parameter
 Inquire Queue Status (Response) command 330
LastGetTime parameter
 Inquire Queue Status (Response) command 331
LastLUWID parameter, Inquire Channel Status (Response) command 221
LastMsgDate parameter, Inquire Channel Status (Response) command 221
LastMsgTime parameter, Inquire Channel Status (Response) command 221
LastPutDate parameter
 Inquire Queue Status (Response) command 331
LastPutTime parameter
 Inquire Queue Status (Response) command 331
LastSequenceNumber parameter, Inquire Channel Status (Response) command 221
LDAPPassword parameter
 Inquire Authentication Information Object (Response) command 155
LDAPPassword, Create authentication information command 37
LDAPUserName parameter
 Inquire Authentication Information Object (Response) command 155
LDAPUserName, Create authentication information command 37
ListenerAttrs parameter, Inquire Channel Listener command 193
ListenerDesc parameter
 Change, Copy, Create Channel Listener command 71
 Inquire Channel Listener (Response) command 196
 Inquire Channel Listener Status (Response) command 200
ListenerName parameter
 Change, Create Channel Listener command 70
 Delete Listener command 138
 Inquire Channel Listener (Response) command 196
 Inquire Channel Listener command 193
 Inquire Channel Listener Status command 197
 Inquire Channel ListenerStatus (Response) command 200
 Start Channel Listener command 403
 Stop Channel Listener command 410
ListenerStatus parameter
 Inquire Channel Initiator (Response) 192

ListenerStatusAttrs parameter, Inquire Channel Listener Status command 198
 ListenerTimer parameter
 Change Queue Manager command 111
 Inquire Queue Manager (Response) command 310
 Local Address parameter
 Inquire Cluster Queue Manager (Response) command 234
 LocalAddress parameter
 Channel commands 51, 221
 Inquire Channel (Response) command 184
 LocalEvent parameter
 Change Queue Manager command 111
 Inquire Queue Manager (Response) command 311
 LocalName parameter
 Change, Copy, Create Channel Listener command 71
 Inquire Channel Listener (Response) command 196
 Inquire Channel Listener Status (Response) command 200
 LogArchive parameter
 Inquire Log (Response) 258
 LogCopyNumber parameter
 Inquire Log (Response) 258
 LogCorrelId parameter
 Inquire Archive (Response) 151
 LoggerEvent parameter
 Change Queue Manager command 111
 Inquire Queue Manager (Response) command 311
 LogLSRN parameter
 Inquire Usage (Response) 360
 LogQMgrNames parameter
 Inquire CF Structure Status (Response) 174
 LogRBA parameter
 Inquire Log (Response) 258
 Inquire Usage (Response) 360
 LogSuspend parameter
 Inquire Log (Response) 258
 LogUsed parameter
 Inquire Log (Response) 259
 LongRetriesLeft parameter, Inquire Channel Status (Response) command 222
 LongRetryCount parameter
 Channel commands 52
 Inquire Channel (Response) command 184
 Inquire Cluster Queue Manager (Response) command 234
 LongRetryInterval parameter
 Channel commands 53
 Inquire Channel (Response) command 184
 Inquire Cluster Queue Manager (Response) command 234
 LU62ARMSuffix parameter
 Change Queue Manager command 112
 LU62ARMSuffix parameter (*continued*)
 Inquire Queue Manager (Response) command 311
 LU62Channels parameter
 Change Queue Manager command 112
 Inquire Queue Manager (Response) command 311
 LUGroupName parameter
 Change Queue Manager command 112
 Inquire Queue Manager (Response) command 311
 LUName parameter
 Change Queue Manager command 112
 Inquire Channel Initiator (Response) 192
 Inquire Queue Manager (Response) command 311
 Start ChannelListener command 403

M
 MaxActiveChannels parameter
 Change Queue Manager command 112
 Inquire Queue Manager (Response) command 311
 MaxArchiveLog parameter
 Inquire Log (Response) 258
 Set Log command 395
 MaxChannels parameter
 Change Queue Manager command 112
 Inquire Queue Manager (Response) command 312
 MaxConnects parameter
 Inquire System (Response) 354
 Set System command 397
 MaxConnectsBackground parameter
 Inquire System (Response) 354
 Set System command 397
 MaxConnectsForeground parameter
 Inquire System (Response) 354
 Set System command 397
 MaxHandles parameter
 Change Queue Manager command 113
 Inquire Queue Manager (Response) command 312
 MaxMsgLength parameter
 Change Queue Manager command 113
 Change, Copy, Create Queue command 89
 Channel commands 53
 Inquire Channel (Response) command 184
 Inquire Channel Status (Response) command 222
 Inquire Cluster Queue Manager (Response) command 234
 Inquire Queue (Response) command 285
 Inquire Queue Manager (Response) command 312
 MaxPriority parameter
 Inquire Queue Manager (Response) command 312
 MaxQDepth parameter
 Change, Copy, Create Queue command 90
 Inquire Queue (Response) command 285
 MaxReadTapeUnits parameter
 Inquire Log (Response) 258
 Set Log command 395
 MaxUncommittedMsgs parameter
 Change Queue Manager command 113
 Inquire Queue Manager (Response) command 312
 MCAJobName parameter, Inquire Channel Status (Response) command 222
 MCAName parameter
 Channel commands 53
 Inquire Channel (Response) command 184
 Inquire Cluster Queue Manager (Response) command 234
 MCASatus parameter, Inquire Channel Status (Response) command 222
 MCAType parameter
 Channel commands 53
 Inquire Channel (Response) command 184
 Inquire Cluster Queue Manager (Response) command 234
 MCAUserIdentifier parameter
 Channel commands 54
 Inquire Channel (Response) command 184
 Inquire Cluster Queue Manager (Response) command 234
 MCAUserIdentifier parameter, Inquire Channel Status (Response) command 222
 MediaRecoveryLog parameter
 Inquire Queue Manager Status (Response) command 321
 MediaRecoveryLogExtent parameter
 Inquire Queue Status (Response) command 331
 message descriptor
 PCF messages 11
 response 13
 MessageCompression parameter
 Channel commands 54
 Inquire Channel (Response) command 185
 Inquire Channel Status (Response) command 222
 Inquire Cluster Queue Manager (Response) command 234
 Mode parameter
 Stop Channel command 408
 Suspend Queue Manager Cluster command 415
 ModeName parameter
 Channel commands 55
 Inquire Channel (Response) command 185

ModeName parameter (*continued*)
 Inquire Cluster Queue Manager
 (Response) command 235
Move Queue 360
MoveType parameter
 Move Queue command 361
mqAddBag 484
mqAddBag call
 Bag parameter 484
 CompCode parameter 484
 ItemValue parameter 484
 Reason parameter 485
 Selector parameter 484
mqAddByteString 469, 485
mqAddByteString call
 Bag parameter 486
 Buffer parameter 486
 BufferLength parameter 486
 CompCode parameter 486
 Reason parameter 486
 Selector parameter 486
mqAddByteStringFilter 469, 487
mqAddByteStringFilter call
 Bag parameter 487
 Buffer parameter 488
 BufferLength parameter 488
 CompCode parameter 488
 ItemValue parameter 488
 Reason parameter 488
 Selector parameter 488
mqAddInquiry 469, 489
mqAddInquiry call
 CompCode parameter 490
 Hbag parameter 490
 Reason parameter 490
 Selector parameter 490
mqAddInteger 469, 491
mqAddInteger call
 Bag parameter 492
 CompCode parameter 492
 ItemValue parameter 492
 Reason parameter 492
 Selector parameter 492
mqAddInteger64 469, 493
mqAddInteger64 call
 Bag parameter 493
 CompCode parameter 493
 ItemValue parameter 493
 Reason parameter 494
 Selector parameter 493
mqAddIntegerFilter 469, 494
mqAddIntegerFilter call
 Bag parameter 495
 CompCode parameter 495
 ItemValue parameter 495
 Operator parameter 495
 Reason parameter 495
 Selector parameter 495
mqAddString 469, 496
mqAddString call
 Bag parameter 496
 Buffer parameter 497
 BufferLength parameter 497
 CompCode parameter 497
 Reason parameter 497
 Selector parameter 497
mqAddStringFilter 469, 498
mqAddStringFilter call
 Bag parameter 498
 Buffer parameter 499
 BufferLength parameter 499
 CompCode parameter 499
 ItemValue parameter 499
 Reason parameter 499
 Selector parameter 499
MQAI
 concepts and terminology 463
 examples 571
 introduction 463
 overview 465
 sample programs
 creating a local queue 571
 displaying events 580
 inquiring queues 575
 printing information 575
 selectors 568
 use 464
MQAI (WebSphere MQ Administration Interface) 9
mqBagToBuffer 479, 500
mqBagToBuffer call
 Buffer parameter 501
 BufferLength parameter 501
 CompCode parameter 501
 DataBag parameter 501
 DataLength parameter 501
 OptionsBag parameter 501
 Reason parameter 501
mqBufferToBag 479, 503
mqBufferToBag call
 Buffer parameter 503
 BufferLength parameter 503
 CompCode parameter 503
 DataBag parameter 503
 OptionsBag parameter 503
 Reason parameter 503
MQCFBF structure 422
MQCFBS structure 425
MQCFH structure 418
MQCFIF structure 427
MQCFIL structure 430
MQCFIN structure 433
MQCFSF structure 434
MQCFSL structure 439
MQCFST structure 442
MQCFT_* values 418
mqClearBag 472, 504
mqClearBag call
 Bag parameter 504
 CompCode parameter 505
 Reason parameter 505
MQCMDL_* values 306
mqCountItems 505
mqCountItems call
 Bag parameter 505
 CompCode parameter 506
 ItemCount parameter 506
 Reason parameter 506
 Selector parameter 506
mqCreateBag 467, 507
mqCreateBag call
 Bag parameter 510
 CompCode parameter 510
 Options parameter 507
mqCreateBag call (*continued*)
 Reason parameter 510
mqCreateBag options 467
mqDeleteBag 468, 511
mqDeleteBag call
 Bag parameter 511
 CompCode parameter 511
 Reason parameter 511
mqDeleteItem 472, 512
mqDeleteItem call
 CompCode parameter 513
 Hbag parameter 512
 ItemIndex parameter 513
 Reason parameter 513
 Selector parameter 512
mqExecute 475, 514
mqExecute call
 AdminBag parameter 515
 AdminQ parameter 516
 Command parameter 514
 CompCode parameter 516
 Hconn parameter 514
 OptionsBag parameter 515
 Reason parameter 516
 ResponseBag parameter 515
 ResponseQ parameter 516
mqGetBag 480, 518
mqGetBag call
 Bag parameter 519
 CompCode parameter 519
 GetMsgOpts parameter 519
 Hconn parameter 518
 Hobj parameter 518
 MsgDesc parameter 518
 Reason parameter 519
MQI Accounting parameter
 Change Queue Manager
 command 113
 Inquire Queue Manager (Response)
 command 312
mqInquireBag 520
mqInquireBag call
 Bag parameter 520
 CompCode parameter 521
 ItemIndex parameter 521
 ItemValue parameter 521
 Reason parameter 522
 Selector parameter 521
mqInquireByteString 523
mqInquireByteString call
 Bag parameter 523
 Buffer parameter 524
 BufferLength parameter 524
 CompCode parameter 524
 ItemIndex parameter 524
 Reason parameter 524
 Selector parameter 523
 StringLength parameter 524
mqInquireByteStringFilter 525
mqInquireByteStringFilter call
 Bag parameter 526
 Buffer parameter 527
 BufferLength parameter 527
 CompCode parameter 527
 ItemIndex parameter 526
 Operator parameter 527
 Reason parameter 527

mqInquireByteStringFilter call (*continued*)

 Selector parameter 526

 StringLength parameter 527

mqInquireInteger 528

mqInquireInteger call

 Bag parameter 529

 CompCode parameter 529

 ItemIndex parameter 529

 ItemValue parameter 529

 Reason parameter 530

 Selector parameter 529

mqInquireInteger64 531

mqInquireInteger64 call

 Bag parameter 531

 CompCode parameter 532

 ItemIndex parameter 532

 ItemValue parameter 532

 Reason parameter 532

 Selector parameter 531

mqInquireIntegerFilter 533

mqInquireIntegerFilter call

 Bag parameter 533

 CompCode parameter 534

 ItemIndex parameter 534

 ItemValue parameter 534

 Operator parameter 534

 Reason parameter 534

 Selector parameter 533

mqInquireItemInfo 535

mqInquireItemInfo call

 Bag parameter 535

 CompCode parameter 537

 ItemIndex parameter 536

 ItemType parameter 537

 OutSelector parameter 537

 Reason parameter 537

 Selector parameter 536

mqInquireString 538

mqInquireString call

 Bag parameter 538

 Buffer parameter 539

 BufferLength parameter 539

 CodedCharSetId parameter 539

 CompCode parameter 539

 ItemIndex parameter 539

 Reason parameter 540

 Selector parameter 539

 StringLength parameter 539

mqInquireStringFilter 541

mqInquireStringFilter call

 Bag parameter 541

 Buffer parameter 542

 BufferLength parameter 542

 CodedCharSetId parameter 542

 CompCode parameter 542

 ItemIndex parameter 542

 Operator parameter 542

 Reason parameter 542

 Selector parameter 542

 StringLength parameter 542

MQIStatistics parameter

 Change Queue Manager command 114

 Inquire Queue Manager (Response) command 312

mqPad 544

mqPad call

 Buffer parameter 544

 BufferLength parameter 544

 CompCode parameter 544

 Reason parameter 544

 String parameter 544

mqPutBag 480, 545

mqPutBag call

 Bag parameter 546

 CompCode parameter 546

 Hconn parameter 545

 Hobj parameter 545

 MsgDesc parameter 546

 PutMsgOpts parameter 546

 Reason parameter 546

mqSetByteString 471, 547

mqSetByteString call

 Bag parameter 547

 Buffer parameter 548

 CompCode parameter 548

 ItemIndex parameter 548

 Reason parameter 548

 Selector parameter 548

mqSetByteStringFilter 471, 550

mqSetByteStringFilter call

 Bag parameter 550

 Buffer parameter 551

 BufferLength parameter 551

 CompCode parameter 551

 ItemIndex parameter 551

 Operator parameter 551

 Reason parameter 551

 Selector parameter 550

mqSetInteger 471, 553

mqSetInteger call

 Bag parameter 553

 CompCode parameter 554

 ItemIndex parameter 554

 ItemValue parameter 554

 Reason parameter 554

 Selector parameter 553

mqSetInteger64 471, 555

mqSetInteger64 call

 Bag parameter 555

 CompCode parameter 556

 ItemIndex parameter 556

 ItemValue parameter 556

 Reason parameter 556

 Selector parameter 556

mqSetIntegerFilter 471, 557

mqSetIntegerFilter call

 Bag parameter 558

 CompCode parameter 559

 ItemIndex parameter 559

 ItemValue parameter 559

 Operator parameter 559

 Reason parameter 559

 Selector parameter 558

mqSetString 471, 560

mqSetString call

 Bag parameter 560

 Buffer parameter 561

 BufferLength parameter 561

 CompCode parameter 561

 ItemIndex parameter 561

 Reason parameter 561

 Selector parameter 561

mqSetStringFilter 471, 563

mqSetStringFilter call

 Bag parameter 563

 Buffer parameter 564

 BufferLength parameter 564

 CompCode parameter 564

 ItemIndex parameter 564

 Operator parameter 564

 Reason parameter 564

 Selector parameter 564

mqTrim 566

mqTrim call

 Buffer parameter 566

 BufferLength parameter 566

 CompCode parameter 566

 Reason parameter 566

 String parameter 566

mqTruncateBag 473, 567

mqTruncateBag call

 Bag parameter 567

 CompCode parameter 568

 ItemCount parameter 567

 Reason parameter 568

MsgDeliverySequence parameter

 Change, Copy, Create Queue command 90

 Inquire Queue (Response) command 285

MsgDeqCount parameter, Reset Queue Statistics (Response) command 380

MsgDesc parameter

 mqGetBag call 518

 mqPutBag call 546

MsgEnqCount parameter, Reset Queue Statistics (Response) command 380

MsgExit parameter

 Channel commands 55

 Inquire Channel (Response) command 185

 Inquire Cluster Queue Manager (Response) command 235

MsgRetryCount parameter

 Channel commands 55

 Inquire Channel (Response) command 185

 Inquire Cluster Queue Manager (Response) command 235

MsgRetryExit parameter

 Channel commands 56

 Inquire Channel (Response) command 185

 Inquire Cluster Queue Manager (Response) command 235

MsgRetryInterval parameter

 Channel commands 56

 Inquire Channel (Response) command 186

 Inquire Cluster Queue Manager (Response) command 235

MsgRetryUserData parameter

 Channel commands 56

 Inquire Channel (Response) command 186

 Inquire Cluster Queue Manager (Response) command 235

Msgs parameter, Inquire Channel Status (Response) command 222

MsgsAvailable parameter
 Inquire Channel Status (Response) command 223
MsgSeqNumber field 419
MsgSeqNumber parameter
 Reset Channel command 375
MsgUserData parameter
 Channel commands 56
 Inquire Channel (Response) command 186
 Inquire Cluster Queue Manager (Response) command 235

N

name spaces 17
NameCount parameter
 Inquire Namelist (Response) command 263
NamelistAttrs parameter, Inquire Namelist command 261
NamelistDesc parameter
 Change, Copy, Create Namelist command 74
 Inquire Namelist (Response) command 263
NamelistName parameter
 Change, Create Namelist command 73
 Delete Namelist command 138
 Inquire Namelist (Response) command 263
 Inquire Namelist command 260
 Inquire Namelist Names command 264
NamelistNames parameter
 Inquire Namelist Names (Response) command 265
NamelistType parameter
 Change, Copy, Create Namelist command 74, 263
NamelistType parameter, Inquire Namelist command 261
Names parameter
 Change, Copy, Create Namelist command 74
 Inquire Namelist (Response) command 263
NetbiosNames parameter
 Change, Copy, Create Channel Listener command 71
 Inquire Channel Listener (Response) command 196
 Inquire Channel Listener Status (Response) command 200
NetTime parameter
 Inquire Channel Status (Response) command 223
NetworkPriority parameter
 Channel commands 57
 Inquire Channel (Response) command 186
NonPersistentDataPages parameter
 Inquire Usage (Response) 359
NonPersistentMessageClass parameter
 Change, Copy, Create Queue command 90

NonPersistentMessageClass parameter (*continued*)
 Inquire Queue (Response) command 285
NonPersistentMsgSpeed parameter
 Channel commands 57
 Inquire Channel (Response) command 186
 Inquire Channel Status (Response) command 223
 Inquire Cluster Queue Manager (Response) command 235

O

ObjectName parameter
 Inquire Connection (Response) 245
 Inquire Entity Authority 249
 Inquire Entity Authority (Response) 252
 Refresh Queue Manager command 370
ObjectType parameter
 Delete Authority Record 134
 Inquire Authority Records 159
 Inquire Authority Records (Response) 163
 Inquire Connection (Response) 245
 Inquire Entity Authority 249
 Inquire Entity Authority (Response) 252
 Refresh Queue Manager command 370
 Set Authority Record 390
OffloadStatus parameter
 Inquire Log (Response) 259
OK response 14
OldestMsgAge parameter
 Inquire Queue Status (Response) command 331
OnQTime parameter
 Inquire Queue Status (Response) command 331
OpenBrowse parameter
 Inquire Queue Status (Response) command 334
OpenInputCount parameter
 Inquire Queue Status (Response) command 332
OpenInputCount parameter, Inquire Queue (Response) command 285
OpenInputType parameter
 Inquire Queue Status (Response) command 334
OpenInquire parameter
 Inquire Queue Status (Response) command 334
OpenOptions parameter
 Inquire Connection (Response) 246
 Inquire Queue Status (Response) command 334
OpenOutput parameter
 Inquire Queue Status (Response) command 334
OpenOutputCount parameter
 Inquire Queue Status (Response) command 332

OpenOutputCount parameter, Inquire Queue (Response) command 286
OpenSet parameter
 Inquire Queue Status (Response) command 334
OpenType parameter
 Inquire Queue Status command 326, 329
Operator field
 MQCFBF structure 423
 MQCFIF structure 428
 MQCFSF structure 435
Operator parameter
 mqAddIntegerFilter call 495
 mqInquireIntegerFilter call 534
 mqSetByteStringFilter call 551
 mqSetIntegerFilter call 559
 mqSetStringFilter call 564
Operator parameter,
 mqInquireByteStringFilter call 527
Operator parameter,
 mqInquireStringFilter call 542
Options parameter
 Inquire Authority Records 158
 Inquire Authority Records (Response) 163
 Inquire Entity Authority 250
Options parameter, mqCreateBag call 507
OptionsBag parameter
 mqBagToBuffer call 501
 mqBufferToBag call 503
 mqExecute call 515
OriginName parameter
 Inquire Connection (Response) 246
OriginUOWId parameter
 Inquire Connection (Response) 246
OTMADruExit parameter
 Inquire System (Response) 354
OTMAGroup parameter
 Inquire System (Response) 355
OTMAInterval parameter
 Inquire System (Response) 355
OTMAMember parameter
 Inquire System (Response) 355
OTMSTpipePrefix parameter
 Inquire System (Response) 355
OutboundPortMax parameter
 Change Queue Manager command 114
 Inquire Queue Manager (Response) command 313
OutboundPortMin parameter
 Change Queue Manager command 114
 Inquire Queue Manager (Response) command 313
OutputBufferCount parameter
 Inquire Log (Response) 258
 Set Log command 395
OutputBufferSize parameter
 Inquire Log (Response) 258
OutSelector parameter,
 mqInquireItemInfo call 537
overview 465

P

padding strings 544
PageSetID
 Inquire Queue command 274
PageSetId parameter
 Change, Copy, Create Storage Class command 129
 Inquire Storage Class (Response) 349
 Inquire Storage Class command 347
 Inquire Usage (Response) 359
 Inquire Usage command 357
PageSetID parameter
 Inquire Queue (Response) command 286
PagesetStatus parameter
 Inquire Usage (Response) 359
Parameter field
 MQCFBF structure 423
 MQCFBS structure 425
 MQCFIF structure 428
 MQCFIL structure 431
 MQCFIN structure 433
 MQCFSF structure 435
 MQCFSL structure 440
 MQCFST structure 443
ParameterCount field 420
ParameterType parameter
 Inquire Archive (Response) 148, 256
 Set Archive command 386
 Set Log command 394
 Set System command 396
PassTicketApplication parameter
 Change, Copy, Create Storage Class command 129
 Inquire Storage Class (Response) 349
Password parameter
 Channel commands 57
 Inquire Channel (Response) command 186
 Inquire Cluster Queue Manager (Response) command 236
PCF (Programmable Command Format)
 responses 13
PCF definitions
 Backup CF Structure 34
 Change Queue Manager 99
 Change Security 124
 Change, Copy, Create authentication information Object 35
 Change, Copy, Create CF Structure 38
 Change, Copy, Create Channel Listener 70
 Change, Copy, Create Namelist 73
 Change, Copy, Create Process 76
 Change, Copy, Create Queue 80
 Change, Copy, Create Service 125
 Change, Copy, Create Storage Class 128
 Channel commands 41
 Change Channel 41
 Copy Channel 41
 Create Channel 41
 Clear Queue 131
 Delete Authentication Information Object 132
 Delete Authority Record 134

PCF definitions (*continued*)
 Delete CF Structure 135
 Delete Channel 136
 Delete Channel Listener 138
 Delete Namelist 138
 Delete Process 140
 Delete Queue 141
 Delete Service 144
 Delete Storage Class 144
 Escape 146
 Escape (Response) 146
 Inquire Archive 147
 Inquire authentication information object (Response) 154
 Inquire Authentication Information Object command 152
 Inquire Authentication Information Object Names command 156
 Inquire Authority Records 158
 Inquire Authority Service 164
 Inquire CF Structure 166
 Inquire CF Structure (Response) 167
 Inquire CF Structure Names 168
 Inquire CF Structure Status 169
 Inquire CF Structure Status (Response) 171
 Inquire Channel 174
 Inquire Channel Initiator 189
 Inquire Channel Initiator (Response) 190
 Inquire Channel Listener 192
 Inquire Channel Listener Status 197
 Inquire Channel Names 202
 Inquire Channel Status 205
 Inquire Cluster Queue Manager 226
 Inquire Connection (Response) 243
 Inquire Connection command 239
 Inquire Entity Authority 248
 Inquire Group 253
 Inquire Group (Response) 253
 Inquire Log 255
 Inquire Namelist 259
 Inquire Namelist Names 264
 Inquire Process 266
 Inquire Process Names 270
 Inquire Queue 272
 Inquire Queue Manager 291
 Inquire Queue Manager Status 319
 Inquire Queue Names 322
 Inquire Queue Status 325
 Inquire Security 336
 Inquire Security (Response) 337
 Inquire Service 339
 Inquire Service Status 342
 Inquire Storage Class 346
 Inquire Storage Class (Response) 349
 Inquire Storage Class Names 350
 Inquire System 352
 Inquire Usage 357
 Inquire Usage (Response) 358
 Move Queue 360
 Ping Channel 362
 Ping Queue Manager 366
 Recover CF Structure 366
 Refresh Cluster 367
 Refresh Queue Manager 369
 Refresh Security 371

PCF definitions (*continued*)
 Reset Channel 373
 Reset Cluster 375
 Reset Queue Manager 377
 Reset Queue Statistics 378
 Resolve Channel 380
 Resume Queue Manager 383
 Resume Queue Manager Cluster 384
 Reverify Security 385
 Set Archive 386
 Set Authority Record 390
 Set Log 394
 Set System 396
 Start Channel 397
 Start Channel Initiator 401
 Start Channel Listener 402
 Start Service 404
 Stop Channel 405
 Stop Channel Initiator 409
 Stop Channel Listener 410
 Stop Connection 412
 Stop Service 412
 Suspend Queue Manager 413
 Suspend Queue Manager Cluster 414
PCF messages
 converting from bag 480
 converting to bag 480
 receiving 480
 sending 480
PerformanceEvent parameter
 Change Queue Manager command 114
 Inquire Queue Manager (Response) command 313
PersistentDataPages parameter
 Inquire Usage (Response) 359
Ping Channel 362
Ping Queue Manager 366
Platform parameter
 Inquire Queue Manager (Response) command 313
Port parameter
 Change, Copy, Create Channel Listener command 72
 Inquire Channel Initiator (Response) 192
 Inquire Channel Listener (Response) command 196
 Inquire Channel Listener Status (Response) command 200
 Start Channel Listener command 404
 Stop Channel Listener command 411
PrincipalNames parameter
 Delete Authority Record 135
 Set Authority Record 393
printing information, sample programs 575
ProcessAttrs parameter
 Inquire Process command 267
ProcessDesc parameter
 Change, Copy, Create Process command 79
 Inquire Process (Response) command 269
ProcessId parameter
 Inquire Channel Listener Status (Response) command 200

ProcessId parameter (*continued*)
 Inquire Connection (Response) 246
 Inquire Queue Status (Response)
 command 335
 Inquire Service Status (Response)
 command 344
ProcessName parameter
 Change, Copy, Create Queue
 command 90
 Change, Create Process command 76
 Delete Process command 140
 Inquire Process (Response)
 command 269
 Inquire Process command 266
 Inquire Process Names
 command 270
 Inquire Queue (Response)
 command 286
ProcessNames parameter
 Inquire Process Names
 (Response) 272
ProfileAttrs parameter, Inquire Authority Records 160
ProfileAttrs parameter, Inquire Entity Authority 250
PropertyName parameter
 Delete Authority Record 134
 Inquire Authority Records 159
 Inquire Authority Records
 (Response) 163
 Set Authority Record 391
Programmable Command Format (PCF)
 authority checking
 Compaq NonStop Kernel 20
 HP OpenVMS 20
 iSeries 18
 UNIX systems 19
 Windows NT 19
 example program 447
 overview 7
 responses 13
Protect parameter
 Inquire Archive (Response) 150
 Set Archive command 389
PSBName parameter
 Inquire Connection (Response) 246
 Inquire Queue Status (Response)
 command 335
PSTId parameter
 Inquire Connection (Response) 246
 Inquire Queue Status (Response)
 command 335
Purge parameter
 Recover CF Structure command 367
Purge parameter, Delete Queue
 command 142
PutAuthority parameter
 Channel commands 57
 Inquire Channel (Response)
 command 186
 Inquire Cluster Queue Manager
 (Response) command 236
PutMsgOpts parameter, mqPutBag
 call 546
 putting data bags 480

Q

QAttrs parameter, Inquire Queue
 command 274
QDepthHighEvent parameter
 Change, Copy, Create Queue
 command 91
 Inquire Queue (Response)
 command 286
QDepthHighLimit parameter
 Change, Copy, Create Queue
 command 91
 Inquire Queue (Response)
 command 286
QDepthLowEvent parameter
 Change, Copy, Create Queue
 command 91
 Inquire Queue (Response)
 command 286
QDepthLowLimit parameter
 Change, Copy, Create Queue
 command 91
 Inquire Queue (Response)
 command 286
QDepthMaxEvent parameter
 Change, Copy, Create Queue
 command 92
 Inquire Queue (Response)
 command 286
QDesc parameter
 Change, Copy, Create Queue
 command 92
 Inquire Queue (Response)
 command 287
QIndexDefer parameter
 Inquire System (Response) 355
QMgrAttrs parameter
 Inquire Queue Manager
 command 291
QMgrCPF parameter
 Inquire Group (Response) 254
QMgrDefinitionType parameter, Inquire Cluster Queue Manager (Response)
 command 236
QMgrDesc parameter
 Inquire Queue Manager (Response)
 command 313
QMgrDesc parameter
 Change Queue Manager
 command 114
QMgrIdentifier parameter
 Inquire Cluster Queue Manager
 (Response) command 236
 Inquire Queue (Response)
 command 287
 Inquire Queue Manager (Response)
 command 313
 Reset Cluster command 376
QMgrName parameter
 Channel commands 58
 Inquire Authority Records
 (Response) 164
 Inquire CF Structure Status
 (Response) 174
 Inquire Channel (Response)
 command 186
 Inquire Channel Status (Response)
 command 223

QMgrName parameter (*continued*)
 Inquire Cluster Queue Manager
 (Response) command 236
 Inquire Entity Authority
 (Response) 253
 Inquire Group (Response) 255
 Inquire Queue (Response)
 command 287
 Inquire Queue Manager (Response)
 command 313
 Inquire Queue Manager Status
 (Response) command 321
 Reset Cluster command 376
 Stop Channel command 408
QMgrNumber parameter
 Inquire Group (Response) 255
QMgrStartDate parameter
 Inquire Log (Response) 259
QMgrStartRBA parameter
 Inquire Log (Response) 259
QMgrStartTime parameter
 Inquire Log (Response) 259
QMgrStatus parameter
 Inquire Group (Response) 255
 Inquire Queue Manager Status
 (Response) command 321
QMgrType parameter, Inquire Cluster Queue Manager (Response)
 command 236
QMgrUOWId parameter
 Inquire Connection (Response) 246
 Inquire Queue Status (Response)
 command 335
QMStatusAttrs parameter
 Inquire Queue Manager Status
 command 319
QName parameter
 Change, Create Queue command 82
 Clear Queue command 131
 Delete Queue command 141
 Inquire Queue (Response)
 command 287, 332, 335
 Inquire Queue command 272
 Inquire Queue Names command 322
 Inquire Queue Status command 325
 Reset Queue Statistics (Response)
 command 380
 Reset Queue Statistics command 378
QNames parameter
 Inquire Queue Names (Response)
 command 324
QServiceInterval parameter
 Change, Copy, Create Queue
 command 92
 Inquire Queue (Response)
 command 287
QServiceIntervalEvent parameter
 Change, Copy, Create Queue
 command 92
 Inquire Queue (Response)
 command 287
QSGDisposition parameter
 Change, Copy, Create Namelist
 command 74
 Change, Copy, Create Process
 command 79

QSGDisposition parameter (*continued*)
 Change, Copy, Create Queue command 93
 Change, Copy, Create Storage Class command 129
 Channel commands 58
 Clear Queue command 132
 Delete Authentication Information Object 133
 Delete Channel command 137
 Delete Namelist 139
 Delete Process command 140
 Delete Queue command 142
 Delete Storage Class command 145
 Inquire Authentication Information Object (Response) command 155
 Inquire Authentication Information Object command 153, 180
 Inquire Authentication Information Object Names command 156, 271
 Inquire Channel (Response) command 187
 Inquire Channel Names command 203
 Inquire Connection (Response) 246
 Inquire Namelist (Response) command 263
 Inquire Namelist command 261
 Inquire Namelist Names command 264
 Inquire Process (Response) command 269, 332, 335
 Inquire Process command 267
 Inquire Queue (Response) command 287
 Inquire Queue command 279
 Inquire Queue Names command 322
 Inquire Queue Status command 326
 Inquire Storage Class (Response) 349
 Inquire Storage Class command 347
 Inquire Storage Class Names command 351
 Move Queue command 361
 Reset Queue Statistics (Response) command 380
QSGDisposition parameter, Create authentication information command 37
QSGDispositions parameter
 Inquire Authentication Information Object Names (Response) 158
 Inquire Channel Names (Response) 204
 Inquire Namelist Names (Response) 265
 Inquire Process Names (Response) 272
 Inquire Queue Names (Response) command 324
 Inquire Storage Class Names (Response) 352
QSGName parameter
 Inquire Group (Response) 255
 Inquire Queue Manager (Response) command 314
 Inquire System (Response) 355

QStatusAttrs parameter, Inquire Queue Status command 326
QType parameter
 Change, Copy, Create Queue command 83
 Delete Queue command 143
 Inquire Queue (Response) command 287
 Inquire Queue command 279
 Inquire Queue Names command 323
QTypes parameter
 Inquire Queue Names (Response) command 324
 querying data items 469
 queue command 11
 SYSTEM.ADMIN.COMMAND .QUEUE 11
QueueAccounting parameter
 Change Queue Manager command 115
 Inquire Queue (Response) command 94, 288
 Inquire Queue Manager (Response) command 314
QueueMonitoring parameter
 Change Queue Manager command 115
 Change, Copy, Create Queue command 94
 Inquire Queue (Response) command 288
 Inquire Queue Manager (Response) command 314
 Inquire Queue Status (Response) command 332
 queues reserved names 18
QueueStatistics parameter
 Change Queue Manager command 115
 Change, Copy, Create Queue command 95
 Inquire Queue Manager (Response) command 314
QuiesceInterval parameter
 Inquire Archive (Response) 150
 Set Archive command 389

R

Reason field
 MQCFH structure 420
Reason parameter
 Change Queue Manager command 123
 Change, Copy, Create Queue command 99
 Channel commands 67
 Clear Queue command 132
 Delete Channel command 138
 Delete Queue command 143
 Escape command 146
 Inquire Authority Records 161
 Inquire Authority Service 165
 Inquire Channel command 180

Reason parameter (*continued*)
 Inquire Channel Listener Status command 199
 Inquire Channel Names command 204
 Inquire Channel Status command 215
 Inquire Entity Authority 251
 Inquire Queue command 280, 330
 Inquire Service Status command 344
 mqAddBag call 485
 mqAddByteString call 486
 mqAddByteStringFilter call 488
 mqAddInquiry call 490
 mqAddInteger call 492
 mqAddInteger64 call 494
 mqAddIntegerFilter call 495
 mqAddString call 497
 mqAddStringFilter call 499
 mqBagToBuffer call 501
 mqBufferToBag call 503
 mqClearBag call 505
 mqCountItems call 506
 mqCreateBag call 510
 mqDeleteBag call 511
 mqDeleteItem call 513
 mqExecute call 516
 mqGetBag call 519
 mqInquireBag call 522
 mqInquireByteString call 524
 mqInquireByteStringFilter call 527
 mqInquireInteger call 530
 mqInquireInteger64 call 532
 mqInquireIntegerFilter call 534
 mqInquireItemInfo call 537
 mqInquireString call 540
 mqInquireStringFilter call 542
 mqPad call 544
 mqPutBag call 546
 mqSetByteString call 548
 mqSetByteStringFilter call 551
 mqSetInteger call 554
 mqSetInteger64 call 556
 mqSetIntegerFilter call 559
 mqSetString call 561
 mqSetStringFilter call 564
 mqTrim call 566
 mqTruncateBag call 568
 Ping Channel command 364
 Reset Channel command 375
 Reset Cluster command 377, 378
 Reset Queue Statistics command 379
 Resolve Channel command 383
 Resume Queue Manager Cluster command 385
 Set Authority Record 135, 393
 Start Channel command 400
 Start Channel Initiator command 402
 Start Channel Listener command 404
 Start Service command 405, 413
 Stop Channel command 409
 Stop Channel Listener command 412
 Suspend Queue Manager Cluster command 243, 415
ReceiveExit parameter
 Channel commands 59

ReceiveExit parameter (*continued*)
 Inquire Channel (Response)
 command 187
 Inquire Cluster Queue Manager (Response) command 237
ReceiveTimeout parameter
 Change Queue Manager
 command 116
 Inquire Queue Manager (Response)
 command 315
ReceiveTimeoutMin parameter
 Change Queue Manager
 command 116
 Inquire Queue Manager (Response)
 command 315
ReceiveTimeoutType parameter
 Change Queue Manager
 command 116
 Inquire Queue Manager (Response)
 command 315
ReceiveUserData parameter
 Channel commands 60
 Inquire Channel (Response)
 command 187
 Inquire Cluster Queue Manager (Response) command 237
 receiving data 479
 receiving data bags 480
 receiving PCF messages 480
 Recover CF Structure 366
 Recover parameter
 Inquire CF Structure (Response) 168
 Recovery parameter
 Copy, Change, Create CF Structure
 command 40
 Refresh Cluster 367
 Refresh Queue Manager 369
 Refresh Security 371
 RefreshInterval parameter
 Refresh Queue Manager
 command 371
 RefreshRepository parameter
 Refresh Cluster command 368
 RefreshType parameter
 Refresh Queue Manager
 command 369
 RemoteApplTag parameter
 Inquire Channel Status (Response)
 command 223
 RemoteEvent parameter
 Change Queue Manager
 command 117
 Inquire Queue Manager (Response)
 command 315
 RemoteQMGrName parameter
 Change, Copy, Create Queue
 command 95
 Inquire Channel Status (Response)
 command 223
 Inquire Queue (Response)
 command 289
 RemoteQName parameter
 Change, Copy, Create Queue
 command 95
 Inquire Queue (Response)
 command 289

RemoveQueues parameter
 Reset Cluster command 376
Replace parameter
 Copy and Create CF Structure
 command 40
 Copy and Create Channel
 command 60
 Copy Channel Listener command 72
 Copy Namelist command 75
 Copy Service command 126
 Copy Storage Class command 130
 Copy, Create Process command 80
 Copy, Create Queue command 96
 Replace parameter, Create authentication information command 38
RepositoryName parameter
 Change Queue Manager
 command 117
 Inquire Queue Manager (Response)
 command 315
RepositoryNamelist parameter
 Change Queue Manager
 command 117
 Inquire Queue Manager (Response)
 command 316
 reserved names
 queues 18
 Reset Channel 373
 Reset Cluster 375
 Reset Queue Manager 377
 Reset Queue Statistics 378
 Reset Queue Statistics (Response) 379
RESLEVELAudit parameter
 Inquire System (Response) 355
 Resolve Channel 380
response
 data 15
 error 14
 extended 15
 OK 14
 standard 14
 structures 417
ResponseBag parameter, mqExecute call 515
ResponseQ parameter, mqExecute call 516
Responses
 Inquire Archive (Response) 148
 Inquire Authentication Information Object Names (Response) 157
 Inquire Authority Records (Response) 161
 Inquire Authority Service (Response) 165
 Inquire CF Structure Names (Response) 169
 Inquire Channel (Response) 181
 Inquire Channel Listener (Response) 195
 Inquire Channel Listener Status (Response) 199
 Inquire Channel Names (Response) 204
 Inquire Channel Status (Response) 216
 Inquire Cluster Queue Manager (Response) command 231

Responses (*continued*)
 Inquire Entity Authority (Response) 251
 Inquire Log (Response) 256
 Inquire Namelist (Response) 262
 Inquire Namelist Names (Response) 265
 Inquire Process (Response) 268
 Inquire Process Names (Response) 271
 Inquire Queue (Response) 281
 Inquire Queue Manager (Response) 299
 Inquire Queue Manager Status (Response) 320
 Inquire Queue Names (Response) 324
 Inquire Queue Status (Response) 330
 Inquire Service (Response) 340
 Inquire Service Status (Response) 344
 Inquire Storage Class Names (Response) 351
 Inquire System (Response) 353
 Reset Queue Statistics (Response) 379
RestartRecoveryLog parameter
 Inquire Queue Manager Status (Response) command 321
 Resume Queue Manager 383
 Resume Queue Manager Cluster 384
RetentionInterval parameter
 Change, Copy, Create Queue command 96
 Inquire Queue (Response) command 289
 Reverify Security 385
RoutingCode parameter
 Inquire Archive (Response) 151
 Inquire System (Response) 355
 Set Archive command 389

S

sample programs
 creating a local queue 571
 displaying events 580
 inquiring queues 575
 printing information 575
Scope parameter
 Change, Copy, Create Queue command 96
 Inquire Queue (Response) command 289
SecurityAttrs parameter
 Inquire Security command 337
SecurityExit parameter
 Channel commands 60
 Inquire Channel (Response) command 187
 Inquire Cluster Queue Manager (Response) command 237
SecurityInterval parameter
 Change Security command 124
 Inquire Security (Response) 338
SecurityItem parameter
 Refresh Security command 372

SecuritySwitch parameter
 Inquire Security (Response) 338
SecuritySwitchProfile parameter
 Inquire Security (Response) 338
SecuritySwitchSetting parameter
 Inquire Security (Response) 338
SecurityTimeout parameter
 Change Security command 125
 Inquire Security (Response) 339
SecurityType parameter
 Refresh Security command 372
SecurityUserData parameter
 Channel commands 61
 Inquire Channel (Response)
 command 187
 Inquire Cluster Queue Manager
 (Response) command 237
Selector parameter
 mqAddBag call 484
 mqAddByteString call 486
 mqAddByteStringFilter call 488
 mqAddInquiry call 490
 mqAddInteger call 492
 mqAddInteger64 call 493
 mqAddIntegerFilter call 495
 mqAddString call 497
 mqAddStringFilter call 499
 mqCountItems call 506
 mqDeleteItem call 512
 mqInquireBag call 521
 mqInquireByteString call 523
 mqInquireByteStringFilter call 526
 mqInquireInteger call 529
 mqInquireInteger64 call 531
 mqInquireIntegerFilter call 533
 mqInquireItemInfo call 536
 mqInquireString call 539
 mqInquireStringFilter call 542
 mqSetByteString call 548
 mqSetByteStringFilter call 550
 mqSetInteger call 553
 mqSetInteger64 call 556
 mqSetIntegerFilter call 558
 mqSetString call 561
 mqSetStringFilter call 564
 selectors 568
 system 569
 user 569
Selectors parameter, Inquire Authority Service 164
SendExit parameter
 Channel commands 61
 Inquire Channel (Response)
 command 187
 Inquire Cluster Queue Manager
 (Response) command 237
 sending administration commands 475
 sending data 479
 sending PCF messages 480
SendUserData parameter
 Channel commands 62
 Inquire Channel (Response)
 command 188
 Inquire Cluster Queue Manager
 (Response) command 237
SeqNumberWrap parameter
 Channel commands 62
SeqNumberWrap parameter (*continued*)
 Inquire Channel (Response)
 command 188
 Inquire Cluster Queue Manager
 (Response) command 237
Service parameter
 Inquire System (Response) 356
 Set System command 397
ServiceAttrs parameter, Inquire Service command 339
ServiceComponent parameter
 Inquire Authority Records 161
 Inquire Authority Service 164
 Inquire Authority Service
 (Response) 165
 Inquire Entity Authority 250
 Set Authority Record 393
ServiceDesc parameter
 Change, Copy, Create Service
 command 126
 Inquire Service (Response)
 command 341
 Inquire Service Status (Response)
 command 345
ServiceName parameter
 Change, Create Service
 command 125
 Delete Service command 144
 Inquire Service (Response)
 command 341
 Inquire Service command 339
 Inquire Service Status (Response)
 command 345
 Inquire Service Status command 342
 Start Service command 405
 Stop Service command 412
ServiceStatusAttrs parameter, Inquire Service Status command 343
ServiceType parameter
 Change, Copy, Create Service
 command 126
 Inquire Service (Response)
 command 341
Sessions parameter
 Change, Copy, Create Channel
 Listener command 72
 Inquire Channel Listener (Response)
 command 196
 Inquire Channel Listener Status
 (Response) command 200
Set Archive 386
Set Authority Record 390
Set Log 394
Set System 396
Shareability parameter
 Change, Copy, Create Queue
 command 97
 Inquire Queue (Response)
 command 289
SharedChannelRestart parameter
 Stop Channel Initiator command 410
ShortRetriesLeft parameter, Inquire Channel Status (Response)
 command 223
ShortRetryCount parameter
 Channel commands 62
ShortRetryCount parameter (*continued*)
 Inquire Channel (Response)
 command 188
 Inquire Cluster Queue Manager
 (Response) command 237
ShortRetryInterval parameter
 Channel commands 62
 Inquire Channel (Response)
 command 188
 Inquire Cluster Queue Manager
 (Response) command 238
SizeMax parameter
 Inquire CF Structure Status
 (Response) 174
SizeUsed parameter
 Inquire CF Structure Status
 (Response) 174
SMFAccounting parameter
 Inquire System (Response) 356
SMFInterval parameter
 Inquire System (Response) 356
 Set System command 397
SMFStatistics parameter
 Inquire System (Response) 356
Socket parameter
 Change, Copy, Create Channel
 Listener command 72
 Inquire Channel Listener (Response)
 command 196
 Inquire Channel Listener Status
 (Response) command 200
SQQMName parameter
 Change Queue Manager
 command 117
 Inquire Queue Manager (Response)
 command 316
SSLCertRemoteIssuerName parameter,
 Inquire Channel Status (Response)
 command 223
SSLCertUserId parameter, Inquire
 Channel Status (Response)
 command 223
SSLCipherSpec parameter
 Channel commands 63, 188, 238
SSLClientAuthentication parameter
 Channel commands 64, 188, 238
SSLCRLNamelist parameter
 Change Queue Manager
 command 117
 Inquire Queue Manager (Response)
 command 316
SSLCryptoHardware parameter
 Change Queue Manager
 command 118
 Inquire Queue Manager (Response)
 command 316
SSLEvent parameter
 Change Queue Manager
 command 119
 Inquire Queue Manager (Response)
 command 316
SSLFipsRequired parameter
 Change Queue Manager
 command 119
 Inquire Queue Manager (Response)
 command 316

SSLKeyResetCount parameter
 Change Queue Manager command 121
SSLKeyRepository parameter
 Change Queue Manager command 120
 Inquire Queue Manager (Response) command 317
SSLKeyResetCount parameter
 Inquire Queue Manager (Response) command 317
SSLKeyResetDate parameter, Inquire Channel Status (Response) command 224
SSLKeyResetTime parameter, Inquire Channel Status (Response) command 224
SSLPeerName parameter
 Channel commands 65, 188, 238
SSLShortPeerName parameter
 Inquire Channel Status (Response) command 224
SSLTasks parameter
 Change Queue Manager command 121
 Inquire Queue Manager (Response) command 317
SSLTasksMax parameter
 Inquire Channel Initiator (Response) 191
SSLTasksStarted parameter
 Inquire Channel Initiator (Response) 191
 Start Channel 397
 Start Channel Initiator 401
 Start Channel Listener 402
 Start Service 404
StartArguments parameter
 Change, Copy, Create Service command 126
 Inquire Service (Response) command 341
 Inquire Service Status (Response) command 345
StartCommand parameter
 Change, Copy, Create Service command 126
 Inquire Service (Response) command 341
 Inquire Service Status (Response) command 345
StartDate parameter
 Inquire Channel Listener Status (Response) command 201
 Inquire Service Status (Response) command 345
StartMode parameter
 Change, Copy, Create Channel Listener command 72
 Change, Copy, Create Service command 127
 Inquire Channel Listener (Response) command 196
 Inquire Channel Listener Status (Response) command 201
StartMode parameter (*continued*)
 Inquire Service (Response) command 341
 Inquire Service Status (Response) command 345
StartStopEvent parameter
 Change Queue Manager command 121
 Inquire Queue Manager (Response) command 317
StartTime parameter
 Inquire Channel Listener Status (Response) command 201
 Inquire Service Status (Response) command 345
StartUOWLogExtent parameter
 Inquire Connection (Response) 247
StatisticsInterval parameter
 Change Queue Manager command 121
 Inquire Queue Manager (Response) command 317
Status parameter
 Inquire Channel Listener Status (Response) command 201
 Inquire Service Status (Response) command 345
StatusType parameter
 Inquire Queue (Response) command 332, 335
StderrDestination parameter
 Change, Copy, Create Service command 127
 Inquire Service (Response) command 342
 Inquire Service Status (Response) command 346
StdoutDestination parameter
 Change, Copy, Create Service command 127
 Inquire Service (Response) command 342
 Inquire Service Status (Response) command 346
StgClassAttrs parameter
 Inquire Storage Class command 348
StgClassName parameter
 Inquire Storage Class (Response) 350
 Stop Channel 405
 Stop Channel Initiator 409
 Stop Channel Listener 410
 Stop Connection Initiator 412
 Stop Service 412
StopArguments parameter
 Change, Copy, Create Service command 127
 Inquire Service (Response) command 342
 Inquire Service Status (Response) command 346
StopCommand parameter
 Change, Copy, Create Service command 127
 Inquire Service (Response) command 342
 Inquire Service Status (Response) command 346
StopRequested parameter, Inquire Channel Status (Response) command 224
StorageClass parameter
 Change, Copy, Create Queue command 97
 Inquire Queue (Response) command 289
 Inquire Queue command 280
StorageClassDesc parameter
 Change, Copy, Create Storage Class command 130
 Inquire Storage Class (Response) 350
StorageClassName parameter
 Change, Copy, Create Storage Class command 128
 Delete Storage Class command 144
 Inquire Storage Class command 346
 Inquire Storage Class Names command 350
StorageClassNames parameter
 Inquire Namelist Names (Response) command 352
String field
 MQCFBS structure 426
 MQCFST structure 444
String parameter
 mqPad call 544
 mqTrim call 566
StringFilterCommand parameter
 Inquire Authentication Information Object command 154
 Inquire CF Structure command 167
 Inquire CF Structure Status command 170
 Inquire Channel command 180
 Inquire Channel Listener command 194
 Inquire Channel Listener Status command 199
 Inquire Channel Status command 215
 Inquire Cluster Queue Manager command 230
 Inquire Connection command 243
 Inquire Namelist command 262
 Inquire Process command 268
 Inquire Queue command 280
 Inquire Queue Status command 329
 Inquire Service command 340
 Inquire Service Status command 344
 Inquire Storage Class command 348
StringLength field
 MQCFBS structure 426
 MQCFSL structure 440
 MQCFST structure 444
StringLength parameter, mqInquireString call 539
StringLength parameter, mqInquireStringFilter call 542
Strings field
 MQCFSL structure 440
StrucLength field
 MQCFBF structure 422
 MQCFBS structure 425
 MQCFH structure 419
 MQCFIF structure 428

StrucLength field (*continued*)
MQCFIL structure 431
MQCFIN structure 433
MQCFSF structure 435
MQCFSL structure 439
MQCFST structure 443
structures 417
MQCFBF 422
MQCFBS 425
MQCFH 418
MQCFIF 427
MQCFIL 430
MQCFIN 433
MQCFSF 434
MQCFSL 439
MQCFST 442
SubState parameter
Inquire Channel Status (Response) command 224
Suspend parameter, Inquire Cluster Queue Manager (Response) command 238
Suspend Queue Manager 413
Suspend Queue Manager Cluster 414
SyncPoint parameter
Inquire Queue Manager (Response) command 317
SysName parameter
Inquire CF Structure Status (Response) command 174
system bag 467
system selectors 569
SYSTEM.ADMIN.COMMAND .QUEUE 11

T

TaskNumber parameter
Inquire Connection (Response) 247
Inquire Queue Status (Response) command 335
TCPChannels parameter
Change Queue Manager command 121
Inquire Queue Manager (Response) command 317
TCPKeepAlive parameter
Change Queue Manager command 122
Inquire Queue Manager (Response) command 318
TCPName parameter
Change Queue Manager command 122
Inquire Channel Initiator (Response) 191
Inquire Queue Manager (Response) command 318
TCPStackType parameter
Change Queue Manager command 122
Inquire Queue Manager (Response) command 318
ThreadId parameter
Inquire Connection (Response) 247
Inquire Queue Status (Response) command 335

TimeSinceReset parameter, Reset Queue Statistics (Response) command 380
TimeStampFormat parameter
Inquire Archive (Response) 151
Set Archive command 389
ToAuthInfoName parameter, Copy authentication information command 36
ToCFStrucName parameter
Copy CF Structure command 39
ToChannelName parameter
Copy Channel command 44
ToListenerName parameter, Copy Channel Listener command 71
ToNamelistName parameter, Copy Namelist command 73
ToProcessName parameter, Copy Process command 76
ToQName parameter
Move Queue command 362
ToQName parameter, Copy Queue command 83
ToServiceName parameter, Copy Service command 126
ToStorageClassName parameter
Copy Storage Class command 128
TotalBuffers parameter
Inquire Usage (Response) 360
TotalLogs parameter
Inquire Log (Response) 259
TotalPages parameter
Inquire Usage (Response) 359
Tpipename parameter
Inquire Queue (Response) command 289
TpName parameter
Channel commands 66
Inquire Channel (Response) command 188
Inquire Cluster Queue Manager (Response) command 238
TPName parameter
Change, Copy, Create Channel Listener command 72
Inquire Channel Listener (Response) command 196
Inquire Channel Listener Status (Response) command 201
TraceClass parameter
Inquire System (Response) 356
TraceRouteRecording parameter
Change Queue Manager command 122
Inquire Queue Manager (Response) command 318
TraceSize parameter
Inquire System (Response) 356
Set System command 397
TransactionId parameter
Inquire Connection (Response) 247
Inquire Queue Status (Response) command 336
TransportType parameter
Change, Create Channel Listener command 70
Channel commands 66
TransportType parameter (continued)
Inquire Channel (Response) command 188
Inquire Channel Initiator (Response) 192
Inquire Channel Listener (Response) command 197
Inquire Channel Listener Status (Response) command 201
Inquire Cluster Queue Manager (Response) command 238
Start Channel Listener command 404
Stop Channel Listener command 411
TransportType parameter, Inquire Channel Listener command 194
TriggerControl parameter
Change, Copy, Create Queue command 97
Inquire Queue (Response) command 290
TriggerData parameter
Change, Copy, Create Queue command 97
Inquire Queue (Response) command 290
TriggerDepth parameter
Change, Copy, Create Queue command 98
Inquire Queue (Response) command 290
TriggerInterval parameter
Change Queue Manager command 123
Inquire Queue Manager (Response) command 318
TriggerMsgPriority parameter
Change, Copy, Create Queue command 98
Inquire Queue (Response) command 290
TriggerType parameter
Change, Copy, Create Queue command 98
Inquire Queue (Response) command 290
trimming blanks from strings 566
truncating a bag 473
Type field
MQCFBF structure 422
MQCFBS structure 425
MQCFH structure 418
MQCFIF structure 427
MQCFIL structure 430
MQCFIN structure 433
MQCFSF structure 435
MQCFSL structure 439
MQCFST structure 443
types of data bag 467
types of data items 468

U

UncommittedMsgs parameter
Inquire Queue Status (Response) command 332
UnitAddress parameter
Inquire Archive (Response) 151

UnitStatus parameter
 Inquire Archive (Response) 151

UnitVolser parameter
 Inquire Archive (Response) 151

UnusedPages parameter
 Inquire Usage (Response) 359

UOWIdentifier parameter
 Inquire Connection (Response) 247

 Inquire Queue Status (Response)
 command 336

UOWLogStartDate parameter
 Inquire Connection (Response) 247

UOWLogStartTime parameter
 Inquire Connection (Response) 247

UOWStartDate parameter
 Inquire Connection (Response) 247

UOWStartTime parameter
 Inquire Connection (Response) 247

UOWState parameter
 Inquire Connection (Response) 248

UOWType parameter
 Inquire Connection (Response) 248

 Inquire Queue Status (Response)
 command 336

Usage parameter
 Change, Copy, Create Queue
 command 98

 Inquire Queue (Response)
 command 290

UsageType parameter
 Inquire Usage command 357

use of the MQAI 464

user bag 467

user data 13

user selectors 569

UserData parameter
 Change, Copy, Create Process
 command 80

 Inquire Process (Response)
 command 270

Userld parameter
 Inquire Connection (Response) 248

 Reverify Security command 385

UserIdentifier parameter
 Channel commands 67

 Inquire Channel (Response)
 command 189

 Inquire Cluster Queue Manager
 (Response) command 239

 Inquire Queue Status (Response)
 command 336

UserIDSupport parameter
 Inquire Authority Service
 (Response) 165

utility calls 483

V

Value field
 MQCFIN structure 433

Values field
 MQCFIL structure 431

Version field
 MQCFH structure 419

W

WebSphere MQ
 Commands (MQSC) 8

WebSphere MQ Administration Interface
 concepts and terminology 463

 creating a local queue 571

 displaying events 580

 examples 571

 inquiring queues 575

 introduction 463

 printing information 575

 sample programs 571

 selectors 568

 use 464

WebSphere MQ Administration Interface
 (MQAI) 9

WLMIInterval parameter
 Inquire System (Response) 356

WLMIIntervalUnits parameter
 Inquire System (Response) 356

X

XCFGroupName parameter
 Change, Copy, Create Storage Class
 command 130

 Inquire Storage Class (Response) 350

XCFMemberName parameter
 Change, Copy, Create Storage Class
 command 131

XmitQName parameter
 Change, Copy, Create Queue
 command 98

 Channel commands 67

 Inquire Channel (Response)
 command 189

 Inquire Channel Status (Response)
 command 225

 Inquire Channel Status
 command 215

 Inquire Queue (Response)
 command 290

XQTime parameter
 Inquire Channel Status (Response)
 command 225

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 44-1962-816151
 - From within the U.K., use 01962-816151
- 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.

IBM

SC34-6598-01



Spine information:



WebSphere MQ

Programmable Command Formats and Administration
Interface

Version 6.0