

# **MAINVIEW® AutoOPERATOR™ for MQSeries Installation and User Guide**

**Version 6.2**

**March 15, 2002**



Copyright © 2002 BMC Software, Inc., as an unpublished work. All rights reserved.

BMC Software, the BMC Software logos, and all other BMC Software product or service names are registered trademarks or trademarks of BMC Software, Inc. IBM and DB2 are registered trademarks of International Business Machines Corp. All other registered trademarks or trademarks belong to their respective companies.

THE USE AND CONTENTS OF THIS DOCUMENTATION ARE GOVERNED BY THE SOFTWARE LICENSE AGREEMENT ENCLOSED AT THE BACK OF THIS DOCUMENTATION.

## Restricted Rights Legend

U.S. GOVERNMENT RESTRICTED RIGHTS. UNPUBLISHED—RIGHTS RESERVED UNDER THE COPYRIGHT LAWS OF THE UNITED STATES. Use, duplication, or disclosure by the U.S. Government is subject to restrictions set forth in FAR Section 52.227-14 Alt. III (g)(3), FAR Section 52.227-19, DFARS 252.227-7014 (b), or DFARS 227.7202, as amended from time to time. Send any contract notices to Contractor/Manufacturer:

**BMC Software, Inc.**  
2101 CityWest Blvd.  
Houston TX 77042-2827  
USA

---

## Contacting BMC Software

You can access the BMC Software Web site at <http://www.bmc.com>. From this Web site, you can obtain general information about the company, its products, special events, and career opportunities. For a complete list of all BMC Software offices and locations, go to <http://www.bmc.com/corporate/offices.html>.

### USA and Canada

**Address** BMC Software, Inc.  
2101 CityWest Blvd.  
Houston TX 77042-2827

**Telephone** 713 918 8800 or  
800 841 2031

**Fax** 713 918 8000

### Outside USA and Canada

**Telephone** (01) 713 918 8800

**Fax** (01) 713 918 8000

---

---

## Customer Support

You can obtain technical support by using the Support page on the BMC Software Web site or by contacting Customer Support by telephone or e-mail. To expedite your inquiry, please see “Before Contacting BMC Software,” below.

### Support Web Site

You can obtain technical support from BMC Software 24 hours a day, seven days a week by accessing the technical support Web site at <http://www.bmc.com/support.html>. From this site, you can

- read overviews about support services and programs that BMC Software offers
- find the most current information about BMC Software products
- search a database for problems similar to yours and possible solutions
- order or download product documentation
- report a problem or ask a question
- subscribe to receive e-mail notices when new product versions are released
- find worldwide BMC Software support center locations and contact information, including e-mail addresses, fax numbers, and telephone numbers

### Support via Telephone or E-mail

In the USA and Canada, if you need technical support and do not have access to the Web, call 800 537 1813. Outside the USA and Canada, please contact your local support center for assistance. To find telephone and e-mail contact information for the BMC Software support center that services your location, refer to the Contact Customer Support section of the Support page on the BMC Software Web site at [www.bmc.com/support.html](http://www.bmc.com/support.html).

### Before Contacting BMC Software

Before you contact BMC Software, have the following information available so that a technical support analyst can begin working on your problem immediately:

- product information
  - product name
  - product version (release number)
  - license number and password (trial or permanent)
- operating-system and environment information
  - machine type
  - operating system type, version, and service pack or program temporary fix (PTF)
  - system hardware configuration
  - serial numbers
  - related software (database, application, and communication) including type, version, and service pack or PTF
- sequence of events leading to the problem
- commands and options that you used
- messages received (and the time and date that you received them)
  - product error messages
  - messages from the operating system, such as `file system full`
  - messages from related software



---

# Contents

<b>Chapter 1. What AutoOPERATOR for MQSeries Is</b> . . . . .	1
Understanding AutoOPERATOR for MQSeries . . . . .	1
Understanding Automation for MQSeries Events . . . . .	1
Understanding Terms and Concepts . . . . .	2
What Messaging and Queuing Is . . . . .	2
What a Queue Manager Is . . . . .	2
How Queue Managers Communicate . . . . .	3
What MQSeries Events Are . . . . .	3
Interactions between AutoOPERATOR and MQSeries . . . . .	7
<b>Chapter 2. Implementing MAINVIEW AutoOPERATOR for MQSeries</b> . . . . .	9
How AutoOPERATOR Connects to MQSeries . . . . .	9
Required Software . . . . .	10
Migration Considerations . . . . .	11
Installation Overview . . . . .	12
Step 1. Activating MAINVIEW AutoOPERATOR for MQSeries . . . . .	15
Specifying Product Option Password Keys . . . . .	15
Adding MQSeries APF-Authorized Libraries to the STEPLIB DD Card . . . . .	17
Modifying AAOPRM00 Parameter Settings . . . . .	17
Modifying AAOMQL00 Parameter Settings (Optional) . . . . .	17
Enabling Distributed Diagnostic Rules . . . . .	19
MAINVIEW AutoOPERATOR for MQSeries Security Considerations . . . . .	19
Restarting the BBI-SS PAS . . . . .	20
Define Queue for Non-OS/390 Instrumentation Events . . . . .	20
Step 2. Enabling Instrumentation Events . . . . .	21
Enabling Instrumentation Events for Non-OS/390 Queue Managers . . . . .	21
Enabling Instrumentation Events for OS/390 Queue Managers . . . . .	21
Step 3. Defining Connectivity . . . . .	22
Making Queue Manager Connections Using Command MQ On Ramp . . . . .	22
Hyperlinking to Command MQ On Ramp in Your Web Browser . . . . .	23
Accessing Command MQ On Ramp Interface . . . . .	23
Specifying the OS/390 Queue Manager . . . . .	24
Specifying Non-OS/390 Queue Managers . . . . .	24
Completing the Queue Manager Connections . . . . .	25
Logging Off . . . . .	27
Diagnostics . . . . .	27
Connecting Queue Managers Manually . . . . .	28
Setting Up OS/390 Queue Manager Connectivity . . . . .	28
Setting Up Non-OS/390 Queue Manager Connectivity . . . . .	28
Step 4. Using the Installation Verification Procedure . . . . .	32
Step 5. Setting up AutoOPERATOR for MQSeries to Co-Exist with PATROL for MQ Products . . . . .	35
MAINVIEW AutoOPERATOR for MQSeries BBI Commands . . . . .	37
<b>Chapter 3. Customizing AutoOPERATOR for MQSeries</b> . . . . .	39
Parameters for BBPARM Member AAOPRMxx . . . . .	39
Generating Instrumentation Events . . . . .	40
Encountering Queues Set to Get(Disabled) . . . . .	41
Encountering Queues Set as NOSHARE . . . . .	41
Parameters for BBPARM Member AAOMQLxx . . . . .	42
Specifying Queues . . . . .	42

Example of Specifying Queues in AAOMQLxx .....	44
<b>Chapter 4. Automating MQSeries Events .....</b>	<b>47</b>
How MAINVIEW AutoOPERATOR for MQSeries Automates MQSeries Events .....	47
What a Rule Is .....	47
What MQS Events Are .....	48
Rule Processor MQSeries Variables .....	49
Enabling MQS Events .....	50
Creating Rules for Event Messages and Non-Event Messages .....	51
Creating a Rule for an MQFMT_EVENT Message .....	51
Creating a Rule for a Non-MQFMT_EVENT Message .....	63
MQSeries Instrumentation Events .....	73
What the Variables for MQSeries Events Are .....	97
SHARED Variables .....	97
AutoOPERATOR-Supplied Variables for all MQSeries Messages .....	98
What the MQS Selection Criteria and Action Specification Panel Fields Are .....	105
Selection Criteria for MQS Panel .....	105
Action Specification for MQS Panel .....	110
Tracking Automation Statistics for MQS Events .....	114
<b>Chapter 5. Viewing AutoOPERATOR for MQSeries Automation Statistics .....</b>	<b>115</b>
Using the AutoOPERATOR for MQSeries Workstation .....	115
Primary Commands .....	116
MQSeries Workstation Fields .....	118
Performance Statistics .....	118
Viewing Queue Management .....	120
<b>Chapter 6. Using AutoOPERATOR for MQSeries Solutions .....</b>	<b>121</b>
Implementation Considerations .....	121
System Queue Archival .....	121
Using the Dead Letter Queue Solution .....	122
Security Considerations .....	122
Implementing the Dead Letter Queue Solution .....	122
Stopping the Dead Letter Queue Solution .....	125
Using the System Queue Archival Solution .....	126
Security Considerations .....	126
Implementing the System Queue Archival Solution .....	127
Enabling the System Queue Archival Rules .....	127
Stopping the System Queue Archival Solution .....	129
Using the Service Interval Performance Solution .....	130
Implementing the Service Interval Performance Solution .....	130
Stopping the Service Interval Performance Solution .....	131
Using the Queue Depth Management Solution .....	132
Implementing the Queue Depth Management Solution .....	132
Stopping the Queue Depth Management Solution .....	133
Using the Channel Availability Solution .....	134
Implementing the Channel Availability Solution .....	135
Stopping the Channel Availability Solution .....	136
Using the Basic Intercommunication Solution .....	137
Security Considerations .....	137
Before You Begin .....	137
Invoking the Basic Intercommunication Solution .....	140
Enabling the Basic Intercommunication Solution .....	140
Stopping the Basic Intercommunication Solution .....	141

<b>Chapter 7. Command MQ Automation Power Line (APL)</b> .....	143
Variables Returned from APL .....	146
Examples .....	148
Example 1– Displaying Attributes of Queue Manager .....	148
Example 2 – Requesting Names of Queue Managers .....	148
Example 3 – Start Queue Manager .....	148
Example 4 – Stop Queue Manager .....	148
Example 5 – Passing Multiple Commands .....	149
<b>Chapter 8. Securing AutoOPERATOR for MQSeries</b> .....	151
<b>Chapter 9. Using MAINVIEW AutoOPERATOR for MQSeries</b>	
<b>IMFEXEC Statements</b> .....	153
MQI Commands .....	154
IMFEXEC MQI Variables .....	154
What the AutoOPERATOR MQI EXEC Interface Is .....	156
How Applications Communicate with MQSeries Objects .....	156
How Applications Interact with MQSeries Objects .....	156
How Completion and Reason Codes Are Returned .....	156
Creating Messages from Variables .....	157
MQI BACK .....	163
MQI CLOSE .....	164
MQI CMIT .....	167
MQI CONN .....	169
MQI DISC .....	171
MQI GET .....	173
MQI OPEN .....	183
MQI PUT .....	187
MQI PUT1 .....	197
CMD (Issue IMFEXEC Command to MQSeries with Response) .....	209
COPY MQI .....	214
DISPLAY MQI .....	217
<b>Appendix A. Diagnosing AutoOPERATOR for MQSeries Errors</b> .....	219
Where You Can Find Additional Information .....	219
Which Tools Are Available to Diagnose AutoOPERATOR for MQSeries Problems .....	220
Diagnostic Checklist .....	222
Frequently Asked Questions .....	223
Examples of Specific Automation Problems and Resolutions .....	233
Dead Letter Queue Solution .....	233
Non-OS/390 Command Times Out .....	237
Rule Will Not Enable a Queue .....	239
<b>Appendix B. MQI Sample Coding</b> .....	241
<b>Appendix C. Rules Variables</b> .....	245
EVENT Variables – Rules and Rule-Invoked EXECs .....	245
Dead Letter Header Variables – Rules and Rule-Invoked EXECs .....	247
Trigger Message Variables – Rules and Rule-Invoked EXECs .....	249
CICS Information Header Variables – Rules and Rule-Invoked EXECs .....	250
IMS Information Header Variables – Rules and Rule-Invoked EXECs .....	254
Transmission Queue Header Variables – Rules and Rule-Invoked EXECs .....	256
Message Descriptor Extension Variables – Rules and Rule-Invoked EXECs .....	261
Work Information Header Variables – Rules and Rule-Invoked EXECs .....	263

Reference Message Header Variables – Rules and Rule-Invoked EXECs .....	264
Trigger Message 2 (Character Format) Variables – Rules and Rule-Invoked EXECs .....	266
Rules and Formatting Header Variables – Rules and Rule-Invoked EXECs .....	267
Rules and Formatting Header Variables Version 2 – Rules and Rule-Invoked EXECs .....	268
PCF Variables – Rules and Rule-Invoked EXECs .....	269
<b>Appendix D. EXECs Variables</b> .....	271
General Purpose Variables – EXECs .....	271
Options and Miscellaneous Variables .....	271
EVENT Variables – EXECs .....	272
Dead Letter Header Variables – EXECs .....	274
Trigger Message Variables – EXECs .....	276
CICS Information Header Variables – EXECs .....	277
IMS Information Header Variables – EXECs .....	281
Transmission Queue Header Variables – EXECs .....	283
Message Descriptor Extension Variables – EXECs .....	288
Work Information Header Variables – EXECs .....	290
Reference Message Header Variables – EXECs .....	291
Trigger Message 2 (Character format) Variables – EXECs .....	293
Rules and Formatting Header Variables – EXECs .....	294
Rules and Formatting Header Variables Version 2 – EXECs .....	295
PCF Variables – EXECs .....	296
<b>Appendix E. Conversion Checklist</b> .....	297
Converting to the Current Release of MAINVIEW AutoOPERATOR .....	297
Rule Processing .....	297
EXEC Manager .....	298
<b>Glossary</b> .....	301
<b>Index</b> .....	313

---

## Tables

1.	Installation Activities — Summary . . . . .	12
2.	Output for .DISPLAY BBI Control Command . . . . .	37
3.	BBI Control Command . . . . .	37
4.	BBPARM Member AAOPRMxx Parameters . . . . .	39
5.	AAOMQLxx Parameters . . . . .	42
6.	Message Format and Description . . . . .	48
7.	MQSeries Instrumentation Events . . . . .	74
8.	SHARED Variables for MAINVIEW AutoOPERATOR for MQSeries . . . . .	97
9.	AutoOPERATOR-Supplied Variables for all MQSeries Messages . . . . .	98
10.	Selection Criteria - MQS Panel's Field Names . . . . .	106
11.	Action Specification - MQS Panel's Field Names . . . . .	110
12.	MQSeries Workstation Panel Primary Commands . . . . .	116
13.	Performance Statistics Field Names . . . . .	118
14.	Queue Management Field Names . . . . .	120
15.	Rules from the Rule Set AAORULBR . . . . .	138
16.	APL parameters . . . . .	144
17.	APL Variables IMFRC, APLCC, and APLRC . . . . .	146
18.	IMFEXEC Command Statements . . . . .	154
19.	MQI EXEC Completion Codes . . . . .	157
20.	MQI Code Examples . . . . .	241
21.	EVENT Variables – Rules and Rule-Invoked EXECs . . . . .	245
22.	Dead Letter Header Variables – Rules and Rule-Invoked EXECs . . . . .	247
23.	Trigger Message Variables – Rules and Rule-Invoked EXECs . . . . .	249
24.	CICS Information Header Variables – Rules and Rule-Invoked EXECs . . . . .	250
25.	IMS Information Header Variables – Rules and Rule-Invoked EXECs . . . . .	254
26.	Transmission Queue Header Variables – Rules and Rule-Invoked EXECs . . . . .	256
27.	Message Descriptor Extension Variables – Rules and Rule-Invoked EXECs . . . . .	261
28.	Work Information Header Variables – Rules and Rule-Invoked EXECs . . . . .	263
29.	Reference Message Header Variables – Rules and Rule-Invoked EXECs . . . . .	264
30.	Trigger Message 2 (Character format) Variables – Rules and Rule-Invoked EXECs . . . . .	266
31.	Rules and Formatting Header Variables – Rules and Rule-Invoked EXECs . . . . .	267
32.	Rules and Formatting Header Variables Version 2 – Rules and Rule-Invoked EXECs . . . . .	268
33.	PCF Variables – Rules and Rule-Invoked EXECs . . . . .	269
34.	General Purpose Variables - EXECs . . . . .	271
35.	Options and Miscellaneous Variables . . . . .	271
36.	Event Variables - EXECs . . . . .	272
37.	Dead Letter Header Variables - EXECs . . . . .	274
38.	Trigger Message Variables - EXECs . . . . .	276
39.	CICS Information Header Variables - EXECs . . . . .	277
40.	IMS Information Header Variables - EXECs . . . . .	281
41.	Transmission Queue Header Variables - EXECs . . . . .	283
42.	Message Descriptor Extension Variables - EXECs . . . . .	288
43.	Work Information Header Variables - EXECs . . . . .	290
44.	Reference Message Header Variables - EXECs . . . . .	291
45.	Trigger Message 2 (Character format) Variables – EXECs . . . . .	293
46.	Rules and Formatting Header Variables – EXECs . . . . .	294
47.	Rules and Formatting Header Variables Version 2 – EXECs . . . . .	295
48.	PCF Variables – (Rules and Rule-Invoked EXECs) . . . . .	296



---

# Figures

1.	Example of Channels . . . . .	3
2.	Interaction between AutoOPERATOR and an OS/390 Queue Manager. . . . .	7
3.	Relationship between AutoOPERATOR and an OS/390 and Non-OS/390 Queue Manager. . . . .	8
4.	Rule Processor Variable Dependencies Panel . . . . .	35
5.	Example of Coding in AAOMQLxx Member. . . . .	44
6.	Creating an MQFMT_EVENT Rule: Automation Control Panel (Example 1) . . . . .	52
7.	Creating an MQFMT_EVENT Rule: Rule Set Overview Panel (Example 1) . . . . .	53
8.	Creating an MQFMT_EVENT Rule: Rule Processor Detail Control Panel (Example 1) . . . . .	53
9.	Creating an MQFMT_EVENT Rule: Rule Processor Detail Control Panel (Example 2) . . . . .	54
10.	Creating an MQFMT_EVENT Rule: Selection Criteria - MQS Panel (Example 1) . . . . .	54
11.	Selection Criteria - MQS Panel (Example 2) . . . . .	55
12.	Example of an MQSeries Event Types Panel . . . . .	56
13.	Example of a Help Panel for Q_DEPTH_HIGH. . . . .	57
14.	Selection Criteria - MQS Panel (Example 3) . . . . .	58
15.	Variable Dependencies - MQS Panel (Example 1) . . . . .	58
16.	Creating an MQFMT_EVENT Rule: Action Specification - MQS Panel (Example 1) . . . . .	59
17.	Action Specification - MQS Panel (Example 2) . . . . .	59
18.	Rule Processor Detail Control Panel (Example 3) . . . . .	61
19.	Rule Set Overview Panel (Example 2) . . . . .	61
20.	Confirming Rule Set Modifications Panel (Example 1) . . . . .	62
21.	Creating a Rule for a Message with Format MQFMT_STRING: Automation Control Panel (Example 1) . . . . .	64
22.	Creating a Rule for a Message with Format MQFMT_STRING: Automation Control Panel (Example 2) . . . . .	64
23.	Creating a Rule for a Message with Format MQFMT_STRING: Rule Set Overview Panel (Example 1) . . . . .	65
24.	Creating a Rule for a Message with Format MQFMT_STRING: Rule Processor Detail Control Panel (Example 1) . . . . .	65
25.	Creating a Rule for a Message with Format MQFMT_STRING: Rule Processor Detail Control Panel (Example 2) . . . . .	66
26.	Creating a Rule for a Message with Format MQFMT_STRING: Selection Criteria - MQS Panel (Example 1) . . . . .	66
27.	Selection Criteria - MQS Panel (Example 2) . . . . .	67
28.	Creating a Rule for a Message with Format MQFMT_STRING: Action Specification Panel . . . . .	68
29.	Action Specification - MQS Panel (Example 2) . . . . .	69
30.	Creating a Rule for a Message with Format MQFMT_STRING: Rule Processor Detail Control Panel (Example 3) . . . . .	70
31.	Rule Set Overview Panel (Example 2) . . . . .	71
32.	Confirming Rule Set Modifications Panel (Example 2) . . . . .	72
33.	Selection Criteria - MQS Field Description . . . . .	105
34.	Action Specification - MQS Field Description . . . . .	110
35.	Example of EVNT Automation Reporter Record . . . . .	114
36.	Example of ACTN Automation Reporter Record . . . . .	114
37.	MQSeries Workstation Panel. . . . .	115
38.	MQSeries Workstation Panel Fields . . . . .	118
39.	MQSeries Workstation - Performance Statistics . . . . .	118
40.	MQSeries Workstation - Queue Management. . . . .	120

41.	Example of Information Returned in Variables APLNOL and APLLN1 through APLLN13 .....	147
42.	Interactions between AutoOPERATOR for MQSeries and MQSeries Queue Managers .....	220

---

# How To Use This Book

This is the *MAINVIEW AutoOPERATOR for MQSeries Installation and User Guide*. Use this book with the MAINVIEW AutoOPERATOR for MQSeries (also referred to as simply AutoOPERATOR for MQSeries) option to learn about how MQSeries events can be automated by AutoOPERATOR Rules or through the AutoOPERATOR for MQSeries EXEC interface.

---

## How This Book Is Organized

This book is organized into the following chapters:

- Chapter 1, “What AutoOPERATOR for MQSeries Is”  
Describes AutoOPERATOR for MQSeries and introduces basic terms and concepts key to understanding fundamental MQSeries functions.
- Chapter 2, “Implementing MAINVIEW AutoOPERATOR for MQSeries”  
Describes what needs to be done to implement AutoOPERATOR for MQSeries (for example, specifying product option password keys, what software levels are required, and how to define communications to OS/390 and non-OS/390 queue managers).
- Chapter 3, “Customizing AutoOPERATOR for MQSeries”  
Describes BBPARM member parameters that determine how AutoOPERATOR for MQSeries operates on your system and how to specify queue managers that will be monitored by AutoOPERATOR for automation.
- Chapter 4, “Automating MQSeries Events”  
Describes how AutoOPERATOR for MQSeries automates MQSeries events with Rules and provides two examples that show step-by-step how to create an MQS Rule.
- Chapter 5, “Viewing AutoOPERATOR for MQSeries Automation Statistics”  
Describes how to view automation statistics displayed by the MAINVIEW AutoOPERATOR for MQSeries Workstation.
- Chapter 6, “Using AutoOPERATOR for MQSeries Solutions”  
Describes how to set up, invoke, and disable the MAINVIEW AutoOPERATOR for MQSeries solutions.
- Chapter 7, “Command MQ Automation Power Line (APL)”  
Describes the Command MQ Automation Power Line (QMPOWER) and its parameters.
- Chapter 8, “Securing AutoOPERATOR for MQSeries”  
Describes some security considerations for AutoOPERATOR for MQSeries.
- Chapter 9, “Using AutoOPERATOR for MQSeries IMFEXEC Statements”  
Describes the AutoOPERATOR MQI EXEC interface commands and several other IMFEXEC command statements that allow you to interface with MQSeries through AutoOPERATOR EXECs.

- Appendix A, “Diagnosing AutoOPERATOR for MQSeries Errors”  
Describes resources and procedures to help resolve problems that you may encounter while installing or using MAINVIEW AutoOPERATOR for MQSeries.
- Appendix B, “MQI Sample Coding”  
Describes sample code that contains all the pieces necessary to execute an MQSeries request and can be copied into your EXECs and modified to fit your needs.
- Appendix C, “Rules Variables”  
Describes all of the variables – Rules and Rule invoked EXECs.
- Appendix D, “EXECs Variables”  
Describes all of the EXEC variables.
- Appendix E, “Conversion Checklist”  
Describes the difference between Rules in AutoOPERATOR 5.1 and the Rules in MAINVIEW AutoOPERATOR 6.2.

This book also contains a glossary and an index.

---

## MAINVIEW AutoOPERATOR Product Library

MAINVIEW AutoOPERATOR is available with seven options:

- MAINVIEW AutoOPERATOR for OS/390
- MAINVIEW AutoOPERATOR for IMS
- MAINVIEW AutoOPERATOR for CICS
- MAINVIEW AutoOPERATOR Access NV
- MAINVIEW AutoOPERATOR TapeSHARE
- MAINVIEW AutoOPERATOR for MQSeries
- MAINVIEW AutoOPERATOR Elan Workstation

The base product and these options are documented in the following MAINVIEW AutoOPERATOR manuals:

- *MAINVIEW AutoOPERATOR Customization Guide*
- *MAINVIEW AutoOPERATOR Basic Automation Guide*
- *MAINVIEW AutoOPERATOR Advanced Automation Guide for CLIST EXECs*
- *MAINVIEW AutoOPERATOR Advanced Automation Guide for REXX EXECs*
- *MAINVIEW AutoOPERATOR Options User Guide*
- *MAINVIEW AutoOPERATOR for MQSeries Installation and User Guide*
- *MAINVIEW AutoOPERATOR Reference Summary*
- *MAINVIEW AutoOPERATOR Solutions Guide*

### Recommended Reading

There are many IBM publications for the IBM MQSeries set of products. BMC Software recommends that you review these IBM publications before you attempt to use the MAINVIEW AutoOPERATOR MQI EXEC interface:

- *IBM MQSeries Application Programming Guide, SC33-0807*
- *IBM MQSeries Application Programming Reference, SC33-1673*

For more information about other available MQSeries publications, the section “About this book” in almost any IBM MQSeries book contains a complete list of MQSeries publications.

---

## Related Reading

The following IBM publications are referenced in this book:

- *IBM MQSeries Application Programming Guide*, SC33-0807
- *IBM MQSeries for Application Programming Reference*, SC33-1673
- *IBM MQSeries Programmable System Management*, SC33-1482
- *MQSeries for Windows NT and Windows 2000 V5R2-1 Quick Beginnings*, GC34-5389
- *IBM MQSeries Command Reference*, SC33-1369
- *IBM MQSeries for OS/390 Messages and Codes*, GC34-5891

and the BMC Software publications:

- *MAINVIEW AutoOPERATOR Customization Guide*
- *MAINVIEW AutoOPERATOR Basic Automation Guide*
- *MAINVIEW AutoOPERATOR Advanced Automation Guide for CLIST EXECs*
- *MAINVIEW AutoOPERATOR Advanced Automation Guide for REXX EXECs*
- *Implementing Security for MAINVIEW Products*

---

## What the Conventions Are

The following syntax notation is used in this book:

- Brackets, [ ], and parentheses, ( ), enclose optional parameters or keywords.
- Braces, { }, enclose a list of parameters: one must be chosen.
- A vertical line, |, separates alternative options; one can be chosen.
- An underlined parameter is the default.
- AN ITEM IN CAPITAL LETTERS must be entered exactly as shown
- Items in lowercase letters are values you supply.

---

# Chapter 1. What AutoOPERATOR for MQSeries Is

This chapter provides an introduction to terms and concepts used for the IBM product MQSeries and the BMC Software product component MAINVIEW AutoOPERATOR for MQSeries (also referred to as AutoOPERATOR for MQSeries).

---

## Understanding AutoOPERATOR for MQSeries

MAINVIEW AutoOPERATOR for MQSeries is a password key-activated AutoOPERATOR product component that allows you to automate MQSeries operations and system management.

The IBM product MQSeries uses the concept of *messaging and queuing*, where a message is a collection of data or information sent by one application to another. The message can be sent to a different computer where it resides in a queue until it is retrieved by the other application. The computer can be in a remote location and it can even use a different platform.

AutoOPERATOR for MQSeries can *listen* for MQSeries events and provide automated responses to the events. AutoOPERATOR for MQSeries also provides an EXEC interface where you can issue MQSeries commands and receive responses with a set of IMFEXEC statements created especially for MQSeries.

---

## Understanding Automation for MQSeries Events

The following lists examples where AutoOPERATOR Rules can be used to automate MQSeries events:

- Generate an AutoOPERATOR ALERT whenever a message is put on the dead letter queue.
- Offload messages when a queue has reached its threshold
- Recover MQSeries channels when a failure occurs.
- Generate a command that will move messages from a system queue which has reached its capacity to an OS/390 generation data set.
- Issue an ALERT to notify operations when messages that are set to be received at a specified rate are not received at that rate.
- Issue an ALERT to notify operations whenever the queue size reaches a given threshold.

AutoOPERATOR provides solutions that you can immediately use for these situations. For more information, refer to Chapter 6, “Using AutoOPERATOR for MQSeries Solutions”.

---

## Understanding Terms and Concepts

This section describes the terms and concepts used for MQSeries and AutoOPERATOR for MQSeries.

### What Messaging and Queuing Is

Messaging and queuing allows programs to communicate with each other across a network. With messaging and queuing there is no need for private, dedicated, or logical connections to link the programs. Programs can communicate by putting messages on queues and taking messages from queues. The terms *Message* and *Queue* are described in the following paragraphs.

**Message:** A message is some data or other communication which a program (or application) sends to a queue for another application to use. A message can be as big as 100 megabytes (MB), although not all platforms support messages of this size.

**Queue:** A queue is a type of MQSeries object which stores the message until it can be retrieved by another application. If you are running MQSeries for OS/390 5.2 and using shared queues, messages can reside in the OS/390 Coupling Facility. A queue can exist on the same system as the sending and receiving applications. This is called a local queue. Queues that exist on a separate system (or in another LPAR) are called remote queues. A queue name can be up to 48 characters long.

One of the advantages of using MQSeries for messaging and queuing is that, once an application sends a message to a queue, the program can continue to run because the message is safely stored in the queue. The retrieving application can collect the message whenever it is ready. In other words, messaging and queuing allows applications to operate independently of each other while still passing data back and forth at different speeds and times, to and from different locations, and without requiring a dialog between applications.

### What a Queue Manager Is

In MQSeries, queues on individual systems are managed by a component called a queue manager. The queue manager provides messaging and queuing services to applications. The queue manager's job is to make sure messages are either put on the correct queue or delivered to another queue manager to accomplish the job. Applications do this through Message Queue Interface (MQI) statements to the queue manager.

On certain platforms (such as OS/390) you can have more than one queue manager per system. On other platforms, you may be restricted to only one queue manager per system. To learn more about platform restrictions, refer to the correspondent platform-specific IBM MQSeries manual.

The following describes different types of queues and queue managers:

**Local queue manager:** A local queue manager is a queue manager that is on the same system as the connecting application. For example, in MQSeries for OS/390, the queue manager can be identified by the subsystem ID in the IEFSSN member of SYS1.PARMLIB. On OS/390 platforms, there may be more than one local queue manager on an OS/390 image.

**Remote queue manager:** A remote queue manager is a queue manager other than the queue manager to which the application is currently connected.

**Dead letter queue:** A dead letter queue is a queue where a queue manager (or application) delivers messages when it cannot deliver the message to its correct destination.

**Transmission queue:** A transmission queue is a local queue (to which a channel is connected) where messages that are targeted to a remote queue are temporarily stored.

## How Queue Managers Communicate

The following terms are used to describe connections between queue managers when using distributed queuing:

**Channel (also known as a message channel):** A channel is a mechanism for moving messages from one queue manager to another. A message channel is comprised of two agents (a sender and a receiver) and a communication link. A message channel agent is a program that sends the message from the transmission queue to (or from) a communication link and then to the destination queue.

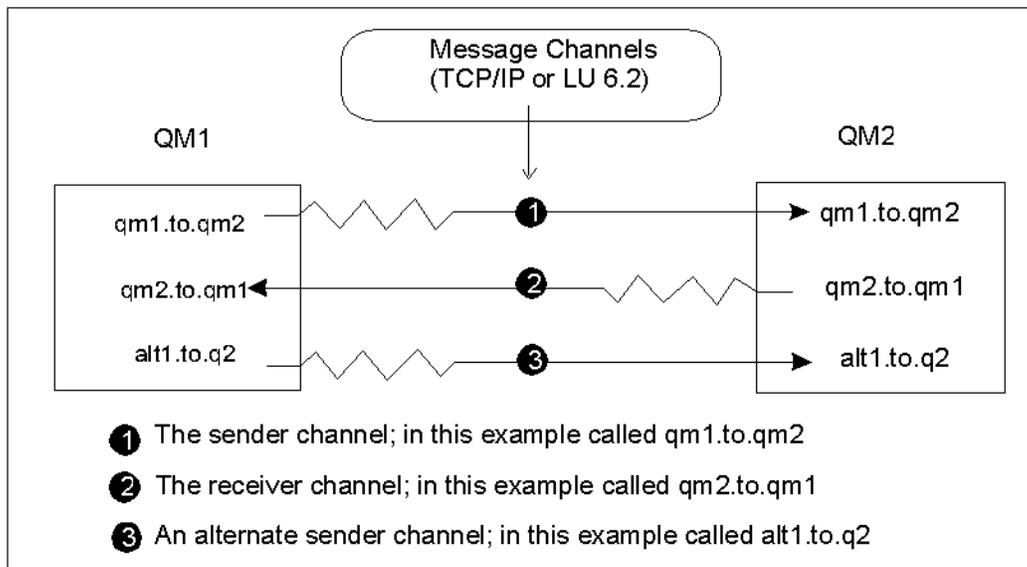


Figure 1. Example of Channels

## What MQSeries Events Are

AutoOPERATOR can listen for events, such as OS/390 commands or messages, and fire off Rules for that event which perform an action, such as message rerouting or creating an AutoOPERATOR ALERT. Messaging and queuing also generates MQSeries instrumentation events for which AutoOPERATOR for MQSeries can listen and fire off Rules.

AutoOPERATOR distinguishes MQSeries messages based on the message format:

- MQFMT\_NONE
- MQFMT\_ADMIN

- MQFMT\_CICS
- MQFMT\_COMMAND\_1
- MQFMT\_COMMAND\_2
- MQFMT\_DEAD\_LETTER\_HEADER
- MQFMT\_EVENT
- MQFMT\_IMS
- MQFMT\_IMS\_VAR\_STRING
- MQFMT\_MD\_EXTENSION
- MQFMT\_PCF
- MQFMT\_REF\_MSG\_HEADER
- MQFMT\_STRING
- MQFMT\_TRIGGER
- MQFMT\_WORK\_INFO\_HEADER
- MQFMT\_XMIT\_Q\_HEADER
- MQFMT\_RF\_HEADER
- MQFMT\_RF\_HEADER\_2
- USER (Used for any format except MQFMT\_EVENT)

When you create a Rule, you must specify which format the message contains, or specify **USER** for any format except MQFMT\_EVENT. A single Rule can recognize *only one* of these two types.

## Messages with Format MQFMT\_EVENT

Messages with format MQFMT\_EVENT are the instrumentation events provided by MQSeries. In MQSeries, an instrumentation event is a logical combination of conditions that is detected by a queue manager or channel instance. The result of such an event is that the queue manager or channel instance puts a special message, called an instrumentation event message, on an event queue.

AutoOPERATOR for MQSeries extracts information appropriate for automation from instrumentation event messages and makes it available to Rules and EXECs in variables. For more information, refer to Chapter 4, “Automating MQSeries Events”, starting on page 47.

There are three categories of MQSeries instrumentation events, and seven types of events within the three categories.

**Queue manager events**

This category of instrumentation events is related to the definitions of resources within queue managers. For example, an application attempts to put a message to a queue that does not exist.

The event messages for queue manager events are put on the SYSTEM.ADMIN.QMGR.EVENT queue.

The following are the event types within the queue manager instrumentation event category, and their associated queue manager attributes:

- Authority (AUTHOREV)
- Inhibit (INHIBTEV)
- Local (LOCALEV)
- Remote (REMOTEEV)
- Start and Stop (STRSTPEV)

For each event type in this list, the associated queue manager attribute enables or disables instrumentation events of that type. For more information about enabling and disabling instrumentation event types, see the IBM publication *MQSeries Command Reference*.

**Performance events**

This category of instrumentation events is a notification that a threshold condition has been reached by a resource. For example, a queue depth limit has been reached.

When a performance event is generated, the queue manager puts the associated event message on the SYSTEM.ADMIN.PERFM.EVENT queue.

All events in the performance event category fall into the performance type.

The queue manager attribute PERFMEV enables or disables the performance event type. For more information about enabling and disabling instrumentation event types, see the IBM publication *MQSeries Command Reference*.

**Channel events**

This category of instrumentation events is reported by channels as a result of conditions detected during their operation; for example, when a channel instance is stopped.

When a channel event is generated, the queue manager puts the associated event message on the SYSTEM.ADMIN.CHANNEL.EVENT queue.

All events in the channel event category fall into the channel type.

Most of the events in the channel event category are enabled automatically and cannot be enabled or disabled by command. The exceptions are the two automatic channel definition events. The queue manager attribute CHADEV enables or disables these two channel events. For more information about enabling and disabling instrumentation event types, see the IBM publication *MQSeries Command Reference*.

## Messages with Format Other Than MQFMT\_EVENT

Non-MQFMT\_EVENT messages are communications sent from one user-written program to another. Several IBM program products use this capability. MAINVIEW AutoOPERATOR for MQSeries can also utilize Non-MQFMT\_EVENT messages. For example, you could have a payroll application that sends data to and from MQSeries user queues. You can create Rules to fire based on the data (or message) that is received.

For additional terms used with AutoOPERATOR for MQSeries, refer to the “Glossary” on page 301.

---

## Interactions between AutoOPERATOR and MQSeries

AutoOPERATOR automates both OS/390 and non-OS/390 MQSeries queue managers and responds to the MQS events. The diagrams in this section illustrate the relationships between the components of AutoOPERATOR and both OS/390 and non-OS/390 queue managers.

Figure 2 shows the interaction between AutoOPERATOR and an OS/390 queue manager.

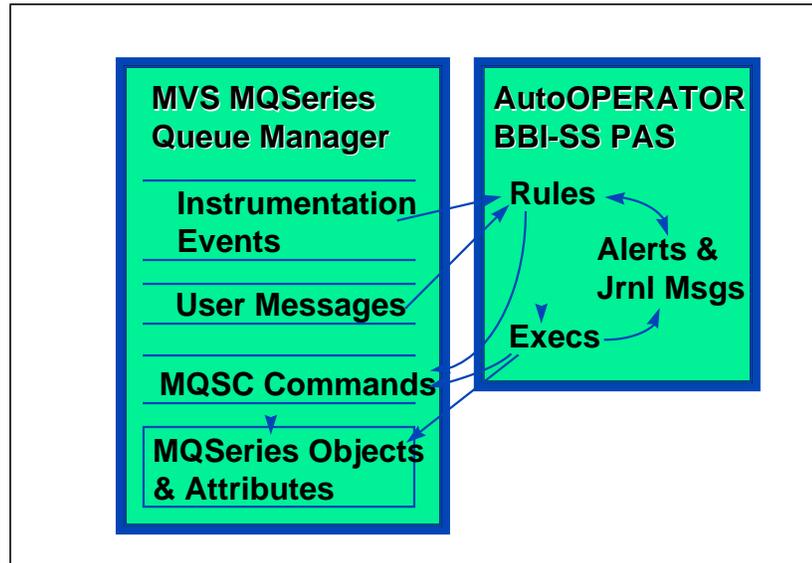


Figure 2. Interaction between AutoOPERATOR and an OS/390 Queue Manager

AutoOPERATOR is connected to an OS/390 MQSeries queue manager, which it listens to for instrumentation events and non-event messages. If the instrumentation event or non-event message is set to trigger a Rule for that event, the Rule can perform one or more of the following actions:

- Create an ALERT or journal message
- Launch an EXEC
- Issue an MQSeries command to manipulate MQSeries objects and attributes

If the Rule issues an ALERT, the ALERT is sent to an operator who performs the appropriate action. A journal message is also logged in the BBI Journal, which contains messages about the queue manager and queues.

If an EXEC is launched, it can create an ALERT, issue an MQSeries command to alter an MQSeries object or attribute, or get and put messages directly using the MQI. The EXEC may also log a message to the BBI Journal.

If the Rule or the EXEC issues an MQSeries command, the MQSeries objects and attributes can be modified to resolve the issue that initially triggered the MQS event.

Figure 3 illustrates the relationship between MAINVIEW AutoOPERATOR and both an OS/390 and non-OS/390 (remote) queue manager.

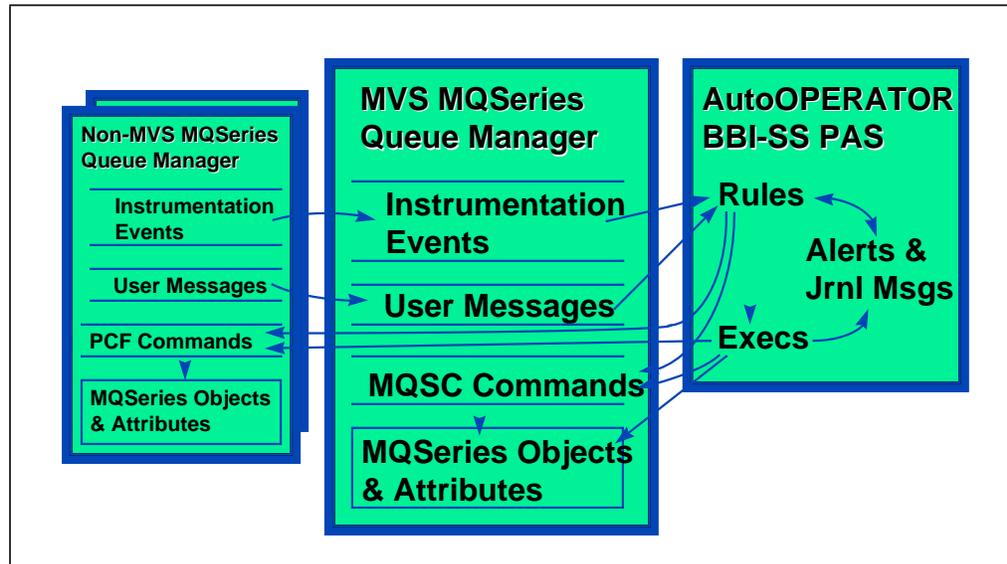


Figure 3. Relationship between AutoOPERATOR and an OS/390 and Non-OS/390 Queue Manager

AutoOPERATOR interacts with a non-OS/390 queue manager in the following ways:

- The non-OS/390 queue manager sends its instrumentation events and non-event messages through the OS/390 queue manager to be listened to by AutoOPERATOR.
- AutoOPERATOR Rules and EXECs can send queue manager commands to the non-OS/390 queue manager in response to these messages and events.
- AutoOPERATOR Rules or EXECs can issue a PCF (Programmable Command Format) command to manipulate the MQSeries objects and attributes in the non-OS/390 queue manager.

---

## Chapter 2. Implementing MAINVIEW AutoOPERATOR for MQSeries

This chapter describes what you need to do to implement MAINVIEW AutoOPERATOR for MQSeries.

---

### How AutoOPERATOR Connects to MQSeries

For MAINVIEW AutoOPERATOR for MQSeries to perform automation of MQSeries events, AutoOPERATOR must make connections to MQSeries local queue managers. MAINVIEW AutoOPERATOR for MQSeries determines which queue managers to connect with automatically and dynamically based on automation needs.

MAINVIEW AutoOPERATOR for MQSeries can connect directly only to MQSeries queue managers running on the same OS/390 image. However, MQSeries instrumentation events on non-OS/390 operating systems can still be automated by MAINVIEW AutoOPERATOR for MQSeries if the non-OS/390 is connected to an OS/390 queue manager. Refer to “Step 3. Defining Connectivity” on page 22 for more information.

Connections between OS/390 queue managers and AutoOPERATOR can be made in one of two ways:

1. For Rules created for event type MQS, AutoOPERATOR searches for the queue manager during Rule enablement. If the queue manager is found and a connection has not already been established, a connection is made.
2. When a queue manager completes initialization, AutoOPERATOR checks for any enabled Rules created for event type MQS that require a connection and if any are found, a connection is made.

At the time of connection, all eligible queue names listed in BBPARM member AAOMQLxx and *only those queues* become eligible for automation. If new queues are added later which require automation, you must refresh the list of eligible queues. For more information about including specific queue managers in BBPARM member AAOMQLxx, refer to “Parameters for BBPARM Member AAOMQLxx” on page 42.

**Note:** At AutoOPERATOR startup time or during a reset situation, the AutoOPERATOR for MQSeries creates the following MQSeries queues for internal processing of commands and replies:

- BBMVAO.COMMAND.REPLY.MODEL
- BBOMVAO.xxxx.RULES.INITIALIZE, BBOMVAO.xxxx.RULES.CMDREPLY
- BBOMVAO.EXEC.REPLY.xxxx.yyyy

where xxxx is the AutoOPERATOR subsystem ID and yyyy is the name of the queue manager.

Users should not be concerned with these queues or events related to these queues, unless error messages are issued regarding these queues.

---

## Required Software

The MAINVIEW AutoOPERATOR for MQSeries component requires a minimum of AutoOPERATOR Version 6, Release 2, Modification 0.

At least one of the following is also required:

- MQSeries for OS/390 Version 1, Release 1, Modification 3
- MQSeries for OS/390 Version 1, Release 1, Modification 4
- MQSeries for OS/390 Version 1, Release 2, Modification 0
- MQSeries for OS/390 Version 2, Release 1, Modification 0
- MQSeries for OS/390 Version 5, Release 2, Modification 0

If you want MAINVIEW AutoOPERATOR for MQSeries to automate MQSeries activity on other operating systems, you need one or more of the following MQSeries products:

- MQSeries for UNIX
- MQSeries for OS/2 Version 2.0.1 or later
- MQSeries for OS/400 Version 3 Release 1 or later
- MQSeries for Windows NT

**Note:** For MAINVIEW AutoOPERATOR for MQSeries to automate MQS events from non-OS/390 event queues, *you must verify that the connectivity between OS/390 and non-OS/390 queue managers is defined*, as described in “Step 3. Defining Connectivity” on page 22.

If you are installing Command MQ On Ramp, you need the following products:

- A Web browser that runs the JDK 1.02 level of Java, i.e. Netscape Navigator 3.04 or Internet Explorer 3.0

Internet Explorer 4.03 does not run Java correctly.

Problems are only supported if they can be reproduced under Netscape Navigator 3.04.

- OS/390 TCP/IP 3.1 or later installed on your OS/390 system

This release of TCP/IP is a base component of OS/390 1.1 and runs on any level of OS/390 currently supported by AutoOPERATOR.

- One PATROL Node Manager for MQ - Administrator product for each non-OS/390 target platform to which you will connect

For more information about this product, refer to the BMC Software documents *PATROL for MQ- Administrator Installation Guide* and *PATROL for MQ- Administrator User Guide*.

---

## Migration Considerations

If you use Rule Set AAORULBQ in your current (5.1) automation, note that this Rule Set has changed. You must make the necessary changes and re-implement this Rule Set to use with MAINVIEW AutoOPERATOR 6.2.

The method for handling remote queue manager Instrumentation Events has changed. Therefore, you need to send instrumentation events from remote queue managers to SYSBMC.DISTRIB.EVENTS in order not to conflict with other products. You must perform “Step 3. Defining Connectivity” on page 22. Additionally, add this queue to your BBPARM member AAOMQL00 and ensure that all rules pertaining to instrumentation events from remote queue managers will process it. These changes will also be required (if not already in effect) for your existing AutoOPERATOR 5.1 system.

**Note:** These changes are not required when migrating from AutoOPERATOR 6.1 to AutoOPERATOR 6.2.

## Installation Overview

This table shows an overview of the MAINVIEW AutoOPERATOR for MQSeries installation activities.

Table 1. Installation Activities — Summary

Summary of Steps	For more information, see
If you are an existing MAINVIEW AutoOPERATOR for MQSeries site, review the migration considerations.	“Migration Considerations” on page 11
Enable MAINVIEW AutoOPERATOR for MQSeries by adding the appropriate Product Option Password Keys.	“Specifying Product Option Password Keys” on page 15
Add MQSeries APF-authorized libraries to AutoOPERATOR STEPLIB.	“Adding MQSeries APF-Authorized Libraries to the STEPLIB DD Card” on page 17
Optional: <ul style="list-style-type: none"> <li>• Activate automatic enabling of MQSeries events</li> <li>• Add MQEV=YES to configuration member AAOPRM00.</li> </ul>	<ul style="list-style-type: none"> <li>• “Modifying AAOPRM00 Parameter Settings” on page 17,</li> <li>• “Step 2. Enabling Instrumentation Events” on page 21</li> <li>• Chapter 3, “Customizing AutoOPERATOR for MQSeries”</li> </ul>
Add BQ to the RULESET= list in AAOPRM00.	“Modifying AAOPRM00 Parameter Settings” on page 17
<ul style="list-style-type: none"> <li>• Modify configuration member AAOMQL00.</li> <li>• Include queue names against which MQSeries Rules are written.</li> </ul>	<ul style="list-style-type: none"> <li>• “Modifying AAOMQL00 Parameter Settings (Optional)” on page 17</li> <li>• Chapter 3, “Customizing AutoOPERATOR for MQSeries”</li> </ul>
Optional. Enable distributed diagnostic Rules.	“Enabling Distributed Diagnostic Rules” on page 19
Review security.	<ul style="list-style-type: none"> <li>• “MAINVIEW AutoOPERATOR for MQSeries Security Considerations” on page 19</li> <li>• “Securing AutoOPERATOR for MQSeries” on page 151</li> </ul>
Enable MQSeries instrumentation events on the OS/390 queue manager that you want monitored.	<ul style="list-style-type: none"> <li>• “Modifying AAOPRM00 Parameter Settings” on page 17,</li> <li>• “Enabling Instrumentation Events for OS/390 Queue Managers” on page 21,</li> <li>• “Generating Instrumentation Events” on page 40,</li> <li>• “Chapter 3. Customizing AutoOPERATOR for MQSeries,”</li> <li>• “Enabling MQS Events” on page 50</li> </ul>

Table 1. Installation Activities — Summary (continued)

Summary of Steps	For more information, see
Optional. Enable MQSeries instrumentation events on non-OS/390 queue manager.	<ul style="list-style-type: none"> <li>• “Step 2. Enabling Instrumentation Events” on page 21</li> <li>• “Step 3. Defining Connectivity” on page 22</li> </ul>
<p>If PATROL® for MQ-Administrator or PATROL® for MQ-Operator is installed on the OS/390 system, you need to do the following steps to allow MAINVIEW AutoOPERATOR for MQSeries to co-exist with these products:</p> <ul style="list-style-type: none"> <li>– Modify member AAOMQL00.</li> <li>– Modify existing rules to use the new event queue name.</li> </ul>	“Step 5. Setting up AutoOPERATOR for MQSeries to Co-Exist with PATROL for MQ Products” on page 35
Optional. Define sender/receiver channel pairs between OS/390 MQ queue managers and non-OS/390 queue managers. This allows MQ to transmit messages and commands between the queue managers.	“Step 3. Defining Connectivity” on page 22
Optional. Define queue SYSBMC.DISTRIB.EVENTS for non-OS/390 events and include this queue in UBBPARAM(AAOMQL00).	<ul style="list-style-type: none"> <li>• “Modifying AAOMQL00 Parameter Settings (Optional)” on page 17</li> <li>• “Define Queue for Non-OS/390 Instrumentation Events” on page 20</li> </ul>
Optional. Route non-OS/390 instrumentation events to SYSBMC.DISTRIB.EVENTS on an OS/390 MQ queue manager to make the events available to MAINVIEW AutoOPERATOR for MQSeries.	“Step 3. Defining Connectivity” on page 22
Optional. Set Up MQ command capability for non-OS/390 MQ queue managers.	“Step 3. Defining Connectivity” on page 22
IVP: verify that MAINVIEW AutoOPERATOR for MQSeries is correctly customized.	“Step 4. Using the Installation Verification Procedure” on page 32
Implement MAINVIEW AutoOPERATOR for MQSeries solutions.	Chapter 6, “Using AutoOPERATOR for MQSeries Solutions”
If SAAREXX/370 is not installed, install REXX/370 alternate library.	Chapter 6, “Using AutoOPERATOR for MQSeries Solutions”
Create site-specific MQ automation Rules to capture MQ messages as AutoOPERATOR events.	<ul style="list-style-type: none"> <li>• Chapter 4, “Automating MQSeries Events”</li> <li>• Appendix C, “Rules Variables”</li> </ul>
Create EXECs to manipulate MQ messages.	<ul style="list-style-type: none"> <li>• Chapter 9, “Using AutoOPERATOR for MQSeries IMFEXEC Statements”</li> <li>• Appendix D, “EXECs Variables”</li> </ul>

Table 1. Installation Activities — Summary (continued)

<b>Summary of Steps</b>	<b>For more information, see</b>
Optional. Create EXECs to manage and maintain non-OS/390 MQ queue managers.	Chapter 7, “Command MQ Automation Power Line (APL)”
Optional. If you are still using release 5.1 and want to take advantage of new features and message data, convert EXECs replacing the old 5.1 statements with the MQI statements that became available in 6.1.	Appendix E, “Conversion Checklist”

---

## Step 1. Activating MAINVIEW AutoOPERATOR for MQSeries

This step describes the initial procedures you must perform to activate AutoOPERATOR for MQSeries.

### Specifying Product Option Password Keys

AutoOPERATOR has features and options that require password keys based upon information about your CPU. BMC Software supplies a utility program called SYSINFO to extract the CPU information.

To extract the CPU information:

1. Run the SYSINFO utility to obtain system information for product password keys.

On the OS/390 system where these products operates, type

**TSO CALL 'hilevel.BBLINK(SYSINFO)'**

where hilevel represents your high-level data set qualifier.

2. Write down the output from the SYSINFO utility in the format that follows:

CPU Type_ _ _ _ CPU Model_ _ CPU Serial Number_ _ _ _ _
---

3. Give the above information to your BMC Software representative who will give you your password key.
4. Edit BBPARM member BBKEYS (create one if you do not have one) and add the key parameters.

If you receive more than one key, each key must begin on a separate line. During key validation, AutoOPERATOR attempts to use the most suitable key by validating each key and selecting the one most qualified for your CPU.

Specify

**KEY=ppp-cccc-tt-nnnn-yyddd-xxxx[-ssss]**

where

ppp Is a three-character product ID; such as

ANV AutoOPERATOR Access NV

CAO AutoOPERATOR for CICS

IAO AutoOPERATOR for IMS

MAO AutoOPERATOR for OS/390

QAO MAINVIEW AutoOPERATOR for MQSeries

TSH AutoOPERATOR TapeSHARE

**Note:** To use the AutoOPERATOR TapeSHARE option, AutoOPERATOR for OS/390 must be installed and the MAO key must be specified. Refer to the MAINVIEW *AutoOPERATOR Customization Guide* for more information.

cccc Is a four-character CPU type.

tt Is a two-character CPU internal model group type.

nnnn Is a four- to five-digit CPU serial number in hexadecimal format.

yyddd Is the julian expiration date.

xxxx Is a four-character authorization key created by BMC Software.

**Note:** The values for ppp, cccc, tt, or nnnn can be a generic \* qualifier; however, the Product Key created by BMC Software must have been generated using this same qualifier.

ssss Is the SMF ID of the system where the product option should be activated.

The use of ssss helps prevent an IPL failure because it identifies an invalid product password key and allows for immediate correction. When an invalid key is detected, a WTOR message is issued and the operator can specify another key immediately. If the SMF ID of the target system is not specified, no external warning messages are issued for invalid keys.

**Important**

If your site's CPU serial number changes or an additional CPU is added, BMC Software must receive the new number so that a new Product Key can be generated.

## Adding MQSeries APF-Authorized Libraries to the STEPLIB DD Card

If the MQSeries libraries are not in the Link List, add the following MQSeries APF-authorized program library or libraries to the STEPLIB DD card in the BBI-SS PAS job:

prefix.SCSQAUTH

prefix.SCSQANLE (for MQSeries for ESA v1.1.4 and later)

where prefix was defined when you installed MQSeries.

**Note:** If you are running more than one queue manager on OS/390, use the MQSeries libraries for the highest level of MQSeries.

## Modifying AAOPRM00 Parameter Settings

In BBPARM member AAOPRM00, make the following changes:

1. Specify the setting for the parameter MQEV= as

MQEV=YES

MQEV=YES specifies that MAINVIEW AutoOPERATOR for MQSeries should automatically enable instrumentation events for a queue manager during connection if they are not enabled already.

If the parameter MQEV= is not specified, the default is MQEV=NO.

2. Update the RULESET= parameter to include the AAORUL00 Rule Set suffix BQ; for example,

RULESET=(BC,CM,BQ)

This ensures that the Rule Set AAORULBQ is enabled when the BBI-SS PAS is cold-started.

For information about other parameters in BBPARM member AAOPRM00, refer to “Parameters for BBPARM Member AAOPRMxx” on page 39.

## Modifying AAOMQL00 Parameter Settings (Optional)

Review the parameter settings in BBPARM member AAOMQL00. The parameters in this member specify which MQSeries queues can be monitored (or listened to) by MAINVIEW AutoOPERATOR for MQSeries automation. The parameters are:

- TYPE=(INCL|EXCL)

Specifies whether MAINVIEW AutoOPERATOR for MQSeries should include or exclude this queue in the set of queues that is eligible for automation. Possible values are:

**INCL** Specifies that AutoOPERATOR should make this queue eligible for possible automation. This is the default setting.

**EXCL** Specifies that AutoOPERATOR should not make this queue eligible for automation.

The abbreviation for TYPE is ‘T’ and its values can be abbreviated from ‘INCL’ to ‘I’ and from ‘EXCL’ to ‘E’. For example, T(I) or T(E).

- QMGR(queue manager name)

Specifies the four-character ID for a local OS/390 queue manager that AutoOPERATOR will monitor. You must specify a queue manager name or a partial queue manager name. There is no default value for this parameter.

The abbreviation for QMGR is 'M'. For example, M(CSQ1).

- QUEUE(queue name)

Specifies the 48-character name of the queue that AutoOPERATOR will make eligible for automation. You must specify a queue name or a partial queue name. There is no default value for this parameter.

The abbreviation for QUEUE 'U'. For example, U(SYSTEM.ADMIN.QMGR.EVENT)

- OPEN is a keyword that gives you control over how the queue is opened and what to do with the messages in the queue after it is opened. The abbreviation for 'OPEN' is 'O'. Its subparameters are

EXCLUSIVE indicates the queue should be opened with the MQOO\_INPUT\_EXCLUSIVE open option. This means other applications are unable to open the queue while AutoOPERATOR has it open. It also means that if another application has the queue opened, AutoOPERATOR will be unable to open it. Its abbreviation is 'E'.

SHARED indicates the queue should be opened with the MQOO\_INPUT\_SHARED option. This is the default option. Its abbreviation is 'S'.

PROCESSOLD indicates that all messages found on the queue when it is opened should be routed through the rule processor to allow automation to take place. This is useful for processing messages that were put on the queue while the queue was not opened by AutoOPERATOR. The abbreviation for this option is 'P'.

IGNOREOLD indicates that AutoOPERATOR will not process the existing messages on the queue and no automation takes place for those messages. Its abbreviation is 'I'. This is the default option.

Example 1: OPEN(EXCLUSIVE,PROCESSOLD) or O(E,P)

Indicates the queue is to be opened with the MQOO\_INPUT\_EXCLUSIVE option and all existing messages found on the queue at open time should be routed through the rule processor for automation.

Example 2: OPEN(SHARED,IGNOREOLD) OR O(S,I)

Indicates the queue is to be opened with the MQOO\_INPUT\_SHARED option and all existing messages found on the queue at open time should be ignored.

For more information about setting these parameters and an example of how to set them, refer to "Parameters for BBPARM Member AAOMQLxx" on page 42.

**Refresh the List of Queues Eligible for Automation** To refresh the list of queues eligible for automation, issue the BBI control command:

```
.RESET MQ xx
```

Refer to "MAINVIEW AutoOPERATOR for MQSeries BBI Commands" on page 37 for more information about BBI commands for MAINVIEW AutoOPERATOR for MQSeries.

## Enabling Distributed Diagnostic Rules

Use the AutoOPERATOR Rules Processor application to enable three Rules distributed in BBPARM member AAORULBQ:

- MQDIA001
- MQDIA002
- MQDIA003

These three Rules help detect the most common setup problems, including connectivity and security problems.

Enable each of the three above Rules as follows:

1. Invoke the Rule Processor application by typing RULES from any BBI terminal session screen.
2. Ensure that the AAORULBQ Rule Set is enabled. If not, enable the Rule Set by typing the (E)nable line command next to the Rule Set name.
3. Select the AAORULBQ Rule Set.
4. Select the Rule MQDIAxxx.
5. Edit MQDIAxxx using the Rules Processor application panels.

Press Enter to navigate through the panels and make the following changes:

- a. On the Selection Criteria panel under Queue Identification, change xxx to the queue manager IDs of the queue managers that this Rule manages. Remember that these queue manager names must be specified in BBPARM member AAOMQLxx.

## MAINVIEW AutoOPERATOR for MQSeries Security Considerations

You must ensure that AutoOPERATOR is given the necessary permissions to read and write to and from the appropriate MQSeries queues. For example, AutoOPERATOR must be able to read from the queues that you want it to monitor.

To issue commands and receive responses, AutoOPERATOR must be able to put messages into the command queue and read responses from the response queue. AutoOPERATOR dynamically builds queues for command responses and for its solutions. These queues start with the high-level qualifier BBOMVAO. The external security manager must give alter authority for the resource BBOMVAO.\* to the BBI-SS PAS. (For more information about security, refer to the appropriate MQSeries system management publication for each operating system.)

For example, if security is active for MQSeries on Windows NT, you must define the security ID associated with the AutoOPERATOR BBI-SS PAS to Windows NT and add this security ID to the mqm group. This is required to allow AutoOPERATOR to send commands to MQSeries on Windows NT and retrieve the command response messages. If the commands are issued by AutoOPERATOR users (BBI-TS users) or OS/390 batch jobs, you also need to define these user IDs to Windows NT and add these user IDs to the mqm group.

Users are defined to Windows NT using the User Manager facility. Refer to the publication *IBM MQSeries for Windows NT System Management Guide* and the Windows NT help topic called *User Manager* for additional information about mqm and defining users.

## Restarting the BBI-SS PAS

To complete the activation of AutoOPERATOR for MQSeries, you must restart (or bounce) the BBI-SS PAS.

First, you must stop the subsystem; issue the OS/390 STOP command:

```
P BBI - SS_PAS_NAME
```

Then issue the OS/390 START command:

```
S BBI - SS_PAS_NAME
```

## Define Queue for Non-OS/390 Instrumentation Events

For each OS/390 MQSeries Queue Manager that MAINVIEW AutoOPERATOR monitors, define the queue to receive instrumentation events from non-OS/390 queue managers as follows:

```
DEFINE QL(SYSBMC.DISTRIB.EVENTS) LIKE(SYSTEM.ADMIN.QMGR.EVENT)
```

---

## Step 2. Enabling Instrumentation Events

This step explains how to enable MQSeries instrumentation events for OS/390 queue managers and non-OS/390 queue managers (if applicable) so that AutoOPERATOR for MQSeries can listen for and automate responses to these events.

You must manually enable instrumentation events for

- All non-OS/390 queue managers
- OS/390 queue managers if you do not choose to set MQEV=YES in AAOPRM00

### Enabling Instrumentation Events for Non-OS/390 Queue Managers

There are seven types of non-OS/390 instrumentation events, six of which can be enabled and disabled. The six non-OS/390 instrumentation event types which can be enabled are Author, Inhibit, Local, Performance, Remote, and Start/Stop. The channel seventh event type is always enabled and cannot be disabled. For more information on enabling or disabling the six events, refer to the IBM manual *IBM MQSeries Programmable System Management*.

These events are queued to the instrumentation queues that have limited capacities. MAINVIEW AutoOPERATOR for MQSeries provides a set of solutions to help you manage these queues. Refer to Chapter 6, “Using AutoOPERATOR for MQSeries Solutions” on page 121 for more information.

### Enabling Instrumentation Events for OS/390 Queue Managers

To specify that MAINVIEW AutoOPERATOR for MQSeries automatically enable instrumentation events for all OS/390 queue managers to which it connects, specify MQEV=YES in BBPARM member AAOPRM00.

For more information about the MQEV= parameter setting, refer to “Modifying AAOPRM00 Parameter Settings” on page 17 and “Parameters for BBPARM Member AAOPRMxx” on page 39.

---

## Step 3. Defining Connectivity

There are two methods to define intercommunication for OS/390 and non-OS/390 queue managers so that AutoOPERATOR can listen for instrumentation events and issue commands to queue managers:

- Define and activate the queue manager connections using Command MQ On Ramp

For more information on using Command MQ On Ramp to make connections between OS/390 and non-OS/390 queue managers, refer to “Making Queue Manager Connections Using Command MQ On Ramp” on page 22.

For more information about how Command MQ On Ramp diagnoses and repairs connection problems, refer to “Using the Basic Intercommunication Solution” on page 137.

- Manually define OS/390 and non-OS/390 queue managers

To manually configure the queue manager connections for AutoOPERATOR, you must define the queue managers, transmission queues, and sender and receiver channels so AutoOPERATOR can listen for non-OS/390 instrumentation events and issue commands to non-OS/390 queue managers.

For more information on manually connecting OS/390 and non-OS/390 queue managers, refer to “Connecting Queue Managers Manually” on page 28.

## Making Queue Manager Connections Using Command MQ On Ramp

Command MQ On Ramp is a Web-based application that inspects and optionally completes the connections between an OS/390 queue manager and non-OS/390 queue managers so that Command MQ products can detect MQSeries events and issue MQSeries commands.

**Setting Up OS/390 Queue Manager Connectivity** To enable AutoOPERATOR to issue commands to the OS/390 queue manager, you must ensure that the queue manager’s command server is running and the command input queue is usable. Command MQ On Ramp ensures that the command server and its command input queue are operational. For information about the command server and starting it, refer to the IBM manual, *IBM MQSeries Command Reference*.

**Setting Up Non-OS/390 Queue Manager Connectivity** MAINVIEW AutoOPERATOR for MQSeries can automate events from non-OS/390 queue managers if the non-OS/390 event messages are directed to local OS/390 queue manager. MAINVIEW AutoOPERATOR for MQSeries is able to detect these non-OS/390 event messages.

Event data is sent to the local OS/390 MQSeries queue manager by subscribing its queue to the PATROL event listener. PATROL automatically transmits the messages from the non-OS/390 queue manager to the OS/390 queue manager. AutoOPERATOR for MQSeries, monitoring the local queues on the OS/390 queue manager, receives the system event messages from the non-OS/390 queue manager and performs whatever automation has been defined.

The following sections explain how to use Command MQ On Ramp to interconnect queue managers:

1. Hyperlinking to Command MQ On Ramp in Your Web Browser
2. Accessing Command MQ On Ramp Interface

3. Specifying the OS/390 Queue Manager
4. Specifying Non-OS/390 Queue Managers
5. Completing the Queue Manager Connections
6. Logging Off

## Hyperlinking to Command MQ On Ramp in Your Web Browser

To invoke Command MQ On Ramp:

1. Create a hyperlink in one of your Web pages by adding the following HTML tag:

```
<a href="http://domain name on OS/390:port number/Cor.html">
Welcome to the Command MQ On Ramp</a>
```

where

*domain name on OS/390* Is the network name or IP address of AutoOPERATOR on OS/390.

*port number* Is the TCP/IP port number of AutoOPERATOR's Web server.

**Note:** You set AutoOPERATOR's port number by setting the LSTNPORT=nnnn parameter in AutoOPERATOR's AAOGMExx member of BBIPARM.

2. Open your Web browser.
3. Open the Web page that you modified in Step 1 from which you want to launch Command MQ On Ramp.
4. Click the Command MQ On Ramp hyperlink.

When you click the hyperlink, a panel appears prompting you to enter a userid and password, which are the TSO userid and password on the OS/390 system from which the Command MQ On Ramp applet was loaded.

5. Type your TSO userid and password.
6. Click **Logon** or press Enter.

A status log reports the progress of the applet.

A separate frame opens to run the Command MQ On Ramp session. The browser is free to be used for other purposes, but keep the Command MQ On Ramp page accessible. If you remove the Command MQ On Ramp page from the stack of pages accessible via the Back and Forward buttons in your browser, the browser may terminate the Command MQ On Ramp applet.

## Accessing Command MQ On Ramp Interface

Once you log on to Command MQ On Ramp, use the new window to specify the queue manager connections.

The Command MQ On Ramp window provides Next and Logoff buttons for navigation. After completing the appropriate portions of each panel, click Next to activate the selections. To log off at any time, click the Logoff button.

The Command MQ On Ramp window also contains menu options:

Option	Description
Windows	<p>Allows you to see the following windows:</p> <p>JournalStatus Journal messages logged by the Command MQ On Ramp as it performs its activities.</p> <p>GMEDiagnostic information about the Command MQ On Ramp General Messages Exchange (GME) function. Command MQ On Ramp uses GME to route messages to applications in AutoOPERATOR and to receive messages from them.</p> <p>AppletBrowser window from which you started Command MQ On Ramp.</p>
Help	Describes the version of Command MQ On Ramp you are using.

## Specifying the OS/390 Queue Manager

Command MQ On Ramp discovers the names of OS/390 queue managers and displays them in a selection list:

- If you know the name of the OS/390 queue manager that you want to specify:
  1. Select the name from the list.
  2. Click **Next**.
- If the name of the OS/390 queue manager does not appear in the list:
  1. Type the name in the text field.
  2. Click **Next**.

The text box must contain a name in order to proceed.

## Specifying Non-OS/390 Queue Managers

When you select an OS/390 queue manager for Command MQ products to use, Command MQ On Ramp prompts you to specify one or more non-OS/390 queue managers to connect to it.

**Note:** Unlike specifying the OS/390 queue manager, Command MQ On Ramp does not automatically provide a list from which you can select non-OS/390 queue managers. Command MQ On Ramp could discover several hundred non-OS/390 queue managers, and it could take a considerable amount of time to compile the list.

1. Specify a non-OS/390 queue manager:
  - If you know the name of a queue manager, type the name in the text field.  
You can also specify the network address of the queue manager.

**Note:** When you connect a specified non-OS/390 queue manager to this OS/390 queue manager for the very first time, you must specify its network address.

- If you do not know the name of the queue manager:

- 1) Click **Discover Non-OS/390 Queue Managers**.

Command MQ On Ramp issues an AutoOPERATOR EXEC to obtain the names and network addresses of the non-OS/390 queue managers already referenced by definitions in the OS/390 queue manager and displays a list of the names discovered.

**Important**

One PATROL Node for MQ - Administrator product must be installed for each non-OS/390 target platform to which you will connect.

For more information about this product, refer to the BMC Software documents *PATROL for MQ- Administrator Installation Guide* and *PATROL for MQ- Administrator User Guide*.

- 2) From the generated list, select a non-OS/390 queue manager that you want to connect.

The Command MQ On Ramp automatically displays the corresponding network address in the Specify the Network Address text box.

2. Click **Add**.

The non-OS/390 queue manager and its network address appear in the box at the bottom of the screen.

3. Repeat Steps 1 and 2 for each non-OS/390 queue manager.
4. When you are finished adding non-OS/390 queue managers, click **Next**.

## Removing a Non-OS/390 Queue Manager

To remove a non-OS/390 queue manager from the list of candidates to be configured:

1. Highlight the name.
2. Click **Delete**.
3. When you are satisfied with the list of non-OS/390 queue managers, click **Next**.

If you want to go back and change the non-OS/390 queue manager, click **Back**.

## Completing the Queue Manager Connections

Once OS/390 and non-OS/390 queue managers are specified in the candidate box, the next panel allows you to inspect only or inspect and complete the queue manager connections. If you only inspect the connections, Command MQ On Ramp takes no action to correct any discovered problems. If you inspect and complete the connections, Command MQ On Ramp tries to fix the problem(s) and only reports an error if a solution is not found.

Beneath the list of queue manager connections to be completed is an area that provides the following information about the selected connection:

- Name of the OS/390 queue manager to be connected
- Name of the selected non-OS/390 queue manager to be connected
- Network address of the non-OS/390 queue manager
- Status of the connection

The status continues to change as Command MQ On Ramp inspects and completes each connection.

To check the specified connections:

1. Determine if the queue manager connections should be only inspected or inspected and completed.
2. Check either the **Inspect** or the **Complete** radio button.
3. Click **Inspect Connections** or **Complete Connections**.

Command MQ On Ramp performs the following actions:

1. Schedules EXECs in AutoOPERATOR to process each connection
2. Displays a status panel in the Command MQ On Ramp window

The status panel indicates that the connection-checking EXEC is running and whether any connection errors have been detected or repaired.

During the connection process, one of the following colors highlights the queue manager name and indicates the status of the listed queue manager connections.

<b>Color</b>	<b>Diagnosis</b>
White	Connection has not been checked.
Blue	AutoOPERATOR is in the process of checking the connection.
Yellow	Possible connection error was detected.
Red	Connection error was detected.
Green	Connection is complete and ready for use.

If you click the Back button at the bottom of the status panel, Command MQ On Ramp stops scheduling the configuration EXEC. Clicking the Back button also returns you to the Specify the non-OS/390 Queue Manager(s) configure window. You can select additional queue managers to be configured and validated.

When you click Next and resume interconnection checking, Command MQ On Ramp checks only newly selected or re-selected interconnections (colored white).

## Logging Off

To log off the Command MQ On Ramp session, perform one of the following:

- Click **Logoff**.
- Close the Command MQ On Ramp frame.
- Close the Web browser.

If you do not close the Web browser, Command MQ On Ramp closes the Command MQ On Ramp window and displays its logon panel again in the Web browser.

## Diagnostics

If Command MQ On Ramp is not performing correctly, one of the following situations could be causing the problem:

- The Java Virtual Machine (JVM) does not have enough storage to run the applet.
  - Look in the browser's Java console for Java error messages.
  - Try using a different vendor's JVM, such as Netscape instead of Microsoft Explorer.
- Security settings may be different.

A firewall or proxy server may prevent Java from connecting to AutoOPERATOR on OS/390. Do not access OS/390 from outside a firewall.

- The Java code is not correct.
  - Look in the browser's Java console for Java error messages.
  - Try running the Java Activator. The Activator is a standard JVM from SUN. If Command MQ On Ramp fails in the Activator, there is a Java coding problem.
- There is a communication problem with AutoOPERATOR's GME.
  - Click **Windows' Journal** to look for errors in the Command MQ On Ramp journal.
  - Click **Windows' GME** to verify a connection exists and it has no errors.
  - Verify there are no errors reported in the BBI Journal.
  - Issue the BBI commands **. D GME** and **. D GMECONN** to verify the connection is valid.
  - Turn on **DEBUGMSG=YES** in **AAOGMExx** and enter **RESET GME, RESYNC**.

If you are unable to diagnose the problem, copy the following items and fax or e-mail them to the BMC Software support team:

- Trace messages from the Java console in the browser window
- Journal messages from the Command MQ On Ramp journal
- Screen images of the GME connection details

---

## Connecting Queue Managers Manually

If you do not use Command MQ On Ramp to configure your queue managers, you must manually define OS/390 and non-OS/390 queue managers and set up AutoOPERATOR to listen for non-OS/390 instrumentation events and issue commands to non-OS/390 queue managers.

### Setting Up OS/390 Queue Manager Connectivity

AutoOPERATOR for MQSeries can automate MQSeries instrumentation events from any message queue in an OS/390 queue manager. These OS/390 queue managers must be running on the same OS/390 image as MAINVIEW AutoOPERATOR for MQSeries. Command MQ On Ramp ensures that these events are enabled.

To enable AutoOPERATOR to issue commands to the local queue manager, you must ensure that the command server is started. For information about the command server and starting it, refer to the IBM manual, *IBM MQSeries Programmable System Management*.

### Setting Up Non-OS/390 Queue Manager Connectivity

MAINVIEW AutoOPERATOR for MQSeries can automate events from non-OS/390 queue managers. To do this, you need to send the non-OS/390 event messages to a local queue automated by MAINVIEW AutoOPERATOR for MQSeries.

Event data is sent to the local instrumentation queues on the OS/390 MQSeries queue manager by defining the system event queues on the non-OS/390 operating system as remote queues. MQSeries automatically transmits the messages from the non-OS/390 queue manager to the OS/390 queue manager. MAINVIEW AutoOPERATOR for MQSeries, monitoring the local queues on the OS/390 queue manager, receives from the non-OS/390 queue manager, the system event messages and performs whatever automation has been defined.

To define connectivity between MAINVIEW AutoOPERATOR for MQSeries and non-OS/390 queue managers, you must

- Set up MAINVIEW AutoOPERATOR for MQSeries to listen for non-OS/390 instrumentation events. See “Setting Up MAINVIEW AutoOPERATOR for MQSeries to Listen for Non-OS/390 Instrumentation Events” on page 29.
- Set up MAINVIEW AutoOPERATOR for MQSeries to issue commands to non-OS/390 queue managers. See “Setting Up MAINVIEW AutoOPERATOR for MQSeries to Issue Commands to Non- OS/390 Queue Managers” on page 30.

## Setting Up MAINVIEW AutoOPERATOR for MQSeries to Listen for Non-OS/390 Instrumentation Events

To set up AutoOPERATOR to listen for non-OS/390 instrumentation events, you must define the non-OS/390 sender and receiver channels and the transmission queue. The following is an example of how to define the channels and transmission queue when using TCP as the transmission type:

1. To define the sender channel at the non-OS/390 queue manager, issue the following MQSC command:

---

```
DEFINE CHANNEL(nonOS/390qmgrid. to. OS/390qmgrid) +  
    CHLTYPE(SDR) +  
    CONNAME(OS/390 ip address) +  
    XMITQ(OS/390qmgrid. xmitq) +  
    CONVERT(YES)  
    TRPTYPE(TCP)
```

---

2. To define the transmission queue on the non-OS/390 queue manager, issue the following MQSC command:

---

```
DEFINE QLOCAL(OS/390qmgrid. xmitq) +  
    USAGE(XMITQ)
```

---

3. To define the receiver channel at the OS/390 queue manager, issue the following MQSC command:

---

```
DEFINE CHANNEL(nonOS/390qmgrid. to. OS/390qmgrid) +  
    CHLTYPE(RCVR) +  
    TRPTYPE(TCP)
```

---

4. If PATROL is not running on the non-OS/390 operating system, issue these MQSC commands to delete the local system event queues:

---

```
DELETE QLOCAL (SYSTEM ADMIN. QMGR. EVENT) +  
PURGE  
DELETE QLOCAL (SYSTEM ADMIN. PERFM. EVENT) +  
PURGE  
DELETE QLOCAL (SYSTEM ADMIN. CHANNEL. EVENT) +  
PURGE
```

---

---

```
DEFINE QREMOTE (SYSTEM ADMIN. QMGR. EVENT) +
    RNAME(SYSBMC. DISTRIBUTOR.EVENTS) +
    RQMNAME(OS/390qmgrid) +
    XMITQ(OS/390qmgrid.xmitq)
```

```
DEFINE QREMOTE (SYSTEM ADMIN. PERFM EVENT) +
    RNAME(SYSBMC. DISTRIBUTOR.EVENTS) +
    RQMNAME(OS/390qmgrid) +
    XMITQ(OS/390qmgrid.xmitq)
```

```
DEFINE QREMOTE (SYSTEM ADMIN. CHANNEL. EVENT) +
    RNAME(SYSBMC. DISTRIBUTOR.EVENTS) +
    RQMNAME(OS/390qmgrid) +
    XMITQ(OS/390qmgrid.xmitq)
```

---

5. For a Non-OS/390 system using a PATROL MQ product, do the following:

Using AMQSPUT or another utility, place a message on the BMC.LISTENER.COM command queue with this contents:

```
subscribe A <OS/390_qmgr> SYSTEM.DISTRIBUTOR.EVENTS
```

where <OS/390\_qmgr> is the Queue Manager name from which the MAINVIEW AutoOPERATOR expects the instrumentation events.

For more information on the publish/subscribe function of ServiceView for MQ, see the *ServiceView for MQ User Guide*. For more information on defining system event queues as local queues, see the *Command MQ for D/S Installation and Administration Guide*.

## Setting Up MAINVIEW AutoOPERATOR for MQSeries to Issue Commands to Non-OS/390 Queue Managers

To set up MAINVIEW AutoOPERATOR to issue commands to a non-OS/390 queue manager, you must define the sender and receiver channels, the transmission queue, and both an OS/390 and a non-OS/390 queue manager alias. The following is an example of defining the channels, transmission queue, and queue manager aliases when using TCP as the transmission type:

1. To define the sender channel at the OS/390 queue manager, type:

---

```
DEFINE CHANNEL (OS/390qmgrid.to.nonOS/390qmgrid) +
    CHLTYPE(SDR) +
    CONNAME (non-OS/390 ip address) +
    XMITQ (nonOS/390qmgrid.xmitq) +
    TRPTYPE(TCP)
```

---

2. To define the transmission queue on the OS/390 queue manager, type:

---

```
DEFINE QLOCAL (nonOS/390qmgrid.xmitq) +
    USAGE (XMITQ)
```

---

3. To define the receiver channel at the non-OS/390 queue manager, type:

---

```
DEFINE CHANNEL (OS/390qmgrid. to.nonOS/390qmgrid) +  
          CHLTYPE(RCVR) +  
          TRPTYPE(TCP)
```

---

4. To define the OS/390 queue manager alias name on the non-OS/390 queue manager, type:

---

```
DEFINE QREMOTE (OS/390qmgrid) +  
          RQMNAME (OS/390qmgrid) +  
          XMITQ (OS/390qmgrid.xmitq)
```

---

5. To define the non-OS/390 queue manager alias name on the OS/390 queue manager, type:

---

```
DEFINE QREMOTE (nonOS/390qmgrid) +  
          RQMNAME (nonOS/390qmgrid) +  
          XMITQ (nonOS/390qmgrid.xmitq)
```

---

**Note:** You must define queue manager aliases for AutoOPERATOR to issue commands to non-OS/390 queue managers and for responses to return to OS/390.

If you have not set up AutoOPERATOR for MQSeries to listen for non-OS/390 instrumentation events, you must do steps 1 through 3 in “Setting Up MAINVIEW AutoOPERATOR for MQSeries to Listen for Non-OS/390 Instrumentation Events” on page 29.

---

## Step 4. Using the Installation Verification Procedure

This step describes how to use the installation verification procedure. The installation verification procedure is a compiled REXX EXEC named QMQIVP00 that tests the configuration of the AutoOPERATOR environment and verifies the current status of MQSeries instrumentation event categories and AutoOPERATOR's ability to automate events.

Before using the installation verification procedure, you must complete "Step 3. Defining Connectivity" on page 22. After defining the needed queues and channels, you must start the MQSeries command server and the MQSeries listener tasks. On OS/390, the command server normally starts automatically after the queue manager starts.

After the command servers and listeners are active, the sender channels on both sides of the connection can be started. For more information on starting command servers, listeners, and channels, see the publication *IBM MQSeries Command Reference*.

Use the QMQIVP00 EXEC to verify that you have defined and connected to OS/390 and non-OS/390 queue managers successfully. This EXEC can be invoked from the BBI Log or from within an EXEC.

See Appendix A, "Diagnosing AutoOPERATOR for MQSeries Errors" on page 219 for a description of how to diagnose installation errors.

The syntax for invoking the QMQIVP00 EXEC from the BBI Log is

```
COMMAND ==>> %QMQIVP00 function LM RM CQ where
```

Function	Is a required user-specified value. Possible values are <ul style="list-style-type: none"><li>• <b>IE</b> creates an OS/390 instrumentation event condition. Use this to test AutoOPERATOR's connection to the OS/390 queue manager and AutoOPERATOR's ability to automate events on that queue manager</li><li>• <b>CMD</b> displays the status of OS/390 instrumentation event conditions for the specified queue manager</li><li>• <b>NCMD</b> displays the status of OS/390 instrumentation event conditions for the specified queue manager</li><li>• <b>NIE</b> creates a non-OS/390 instrumentation event condition. Use this to test AutoOPERATOR's connection to the non-OS/390 queue manager and AutoOPERATOR's ability to automate events on that queue manager</li></ul>
LM	Is a required value where you type the local queue manager ID. You must type a local queue manager name to use the EXEC even if you are trying to query the status of non-OS/390 queue managers.

RM	<p>Is the local queue manager's alias name of a non-OS/390 queue manager. This field is required when you are querying a non-OS/390 (or remote) queue manager.</p> <p><b>Note:</b> The local queue manager alias name of a non-OS/390 queue manager is created in Step 5 of "Setting Up MAINVIEW AutoOPERATOR for MQSeries to Issue Commands to Non-OS/390 Queue Managers" on page 30. This is the name that must be used when you are querying a non-OS/390 (or remote) queue manager with this EXEC.</p>
CQ	<p>Is an optional value where you type the name for a non-OS/390 command queue. The default value is SYSTEM.ADMIN.COMMAND.QUEUE. This parameter is not applicable to OS/390 queue managers.</p>

Following are some examples about how to use the QMQIVP00 EXEC to verify that you have successfully completed implementing communications between AutoOPERATOR for MQSeries and MQSeries OS/390 and non-OS/390 queue managers.

**Example 1:** Displaying the status in the BBI Log of OS/390 instrumentation event conditions for a local queue manager named csbe.

Command ==> **%QMQIVP00 cmd csbe**

The results are displayed in the BBI Log.

```
. QMQIVP00 - DISPLAY STATUS OF CURRENT INSTRUMENTATION EVENT CATEGORI
. QMQIVP00 - CSQM409I QMNAME(CSBE)
. QMQIVP00 - INHIBTEV(ENABLED)
. QMQIVP00 - LOCALEV(ENABLED)
. QMQIVP00 - REMOTEEV(ENABLED)
. QMQIVP00 - STRSTPEV(ENABLED)
. QMQIVP00 - PERFMEV(ENABLED)
```

This output shows the name of the local queue manager and the status of the instrumentation events in that queue manager. Use this output to determine which instrumentation events are enabled.

**Example 2:** Creating an OS/390 instrumentation event condition for a local queue manager named csbe.

Command ==> **%QMQIVP00 ie csbe**

The results are displayed in the BBI Log.

```
. QMQIVP00 - OS/390 INSTRUMENTATION CONDITION
. QMQIVP00 - SUCCESSFULLY CREATED
```

This output shows that both AutoOPERATOR and the OS/390 queue manager are properly configured to allow AutoOPERATOR to automate instrumentation events on the specified OS/390 queue manager.

**Example 3:** Displaying the status of non-OS/390 (remote) instrumentation event conditions for a remote queue manager whose local queue manager alias is epesi n.

Command ==> **%QMIVP00 ncmd csbe epsesin**

The results are displayed in the BBI Log.

```
. QMIVP00 - DISPLAY STATUS OF CURRENT INSTRUMENTATION EVENT CATEGORI
. QMIVP00 - AMQ8408: Display Queue Manager details.
. QMIVP00 - QMNAME(EPESIN)
. QMIVP00 - AUTHOREV(DISABLED)
. QMIVP00 - INHIBTEV(DISABLED)
. QMIVP00 - LOCALEV(DISABLED)
. QMIVP00 - REMOTEEV(DISABLED)
. QMIVP00 - PERFMEV(DISABLED)
. QMIVP00 - STRSTPEV(ENABLED)
```

This output shows the name of the non-OS/390 queue manager and the status of the instrumentation events on that remote queue manager. Use this output to determine which instrumentation events are enabled.

**Example 4:** Creating a non-OS/390 instrumentation event condition for a remote queue manager whose local queue manager alias is `epsesin`.

Command ==> **%QMIVP00 nie csbe epsesin**

The results are displayed in the BBI Log.

```
. QMIVP00 - NON OS/390 INSTRUMENTATION EVENT CONDITION
. QMIVP00 - SUCCESSFULLY CREATED
```

This output shows that all of AutoOPERATOR, the OS/390 queue manager, and the remote queue manager are properly configured to allow AutoOPERATOR to automate instrumentation events on that remote queue manager.



MQ - Operator version 2.2.00 or PATROL for MQ - Administrator version 3.1.00 products.

5. Remove BV from the Rule Set list specified in the AAOPRM00 member. This Rule Set is required for older releases of the PATROL products only.

## MAINVIEW AutoOPERATOR for MQSeries BBI Commands

This section describes the new output for current BBI control commands that is displayed when the MAINVIEW AutoOPERATOR for MQSeries component is installed. You can use these commands on any command line when you want to see the status of the MAINVIEW AutoOPERATOR for MQSeries component or when you want to refresh the list of queues to which MAINVIEW AutoOPERATOR for MQSeries is connected.

Table 2. Output for .DISPLAY BBI Control Command

BBI control command	Output description
.DISPLAY ACTIVE or .D ACTIVE	When MAINVIEW AutoOPERATOR for MQSeries is active, issuing this command shows that the MAINVIEW AutoOPERATOR for MQSeries component is active.
.DISPLAY PRODUCTS or .D PRODUCTS	When MAINVIEW AutoOPERATOR for MQSeries is installed, issuing this command shows that the MAINVIEW AutoOPERATOR for MQSeries component is installed.
.DISPLAY KEYS or .D KEYS	When MAINVIEW AutoOPERATOR for MQSeries is installed, issuing this command shows the status of the MAINVIEW AutoOPERATOR for MQSeries component key.

Table 3 describes new BBI commands for MAINVIEW AutoOPERATOR for MQSeries.

Table 3. BBI Control Command

BBI control command	Output description
.RESET MQ	Causes MAINVIEW AutoOPERATOR for MQSeries to read the currently active AAOMQLxx member and scan all the currently existing MQ queues, thereby refreshing the data regarding which MQ queues are eligible for automation.
.RESET MQ xx	Allows you to activate a different AAOMQLxx member where xx is the two-character suffix of an AAOMQLxx member.
.RESET QM nnnn	Causes MAINVIEW AutoOPERATOR for MQSeries to stop and restart the connection to a single MQSeries queue manager where nnnn is the four-character name of the queue manager.  For example:  .RESET QM CSQ1  The queue manager name must be specified.



---

## Chapter 3. Customizing AutoOPERATOR for MQSeries

This chapter describes parameters in BBPARM member AAOPRMxx and the BBPARM member for MQSeries, AAOMQLxx.

Using the parameters in these members you can specify which MQSeries queues will be monitored (or listened to) by AutoOPERATOR and customize which actions AutoOPERATOR for MQSeries will perform on your system. This chapter also describes how to enable instrumentation events.

---

### Parameters for BBPARM Member AAOPRMxx

Table 4 describes the parameters contained in BBPARM member AAOPRMxx for the AutoOPERATOR for MQSeries component.

You can specify that each BBI-SS PAS use its own AAOPRMxx member where xx is a user-specified two-character suffix. In the UBBPARM member CFGssidA (where ssid is the subsystem ID), specify the AAOPRMxx member name for the subsystem. For more information, refer to the *MAINVIEW Common Customization Guide*.

Table 4. BBPARM Member AAOPRMxx Parameters

Parameter	Description
MQEV=YES  <u>NO</u>	Specifies that AutoOPERATOR for MQSeries should automatically enable instrumentation events for a queue manager during connection if it is not already enabled.  The default value is NO.
MQGINHIB=xxxxxxx  <u>JRNL</u>	Specifies the action AutoOPERATOR should take when AutoOPERATOR attempts to listen to a queue which is defined as GET(DISABLED). Possible settings are  <b>JRNL</b> Issue a message to the BBI Journal stating that AutoOPERATOR cannot listen to the queue.  <b>WTO</b> Issue a write-to-operator (WTO) message stating that AutoOPERATOR cannot listen to the queue.  <b>IGNORE</b> Take no action.  <b>ALTER</b> Alter the queue to GET(ENABLED).  The default value is JRNL.

Table 4. BBPARM Member AAOPRMxx Parameters (Continued)

Parameter	Description
MQNSHARE=xxxxxxxx JRNL	<p>Specifies the action AutoOPERATOR should take when AutoOPERATOR attempts to listen to a queue which is defined as NOSHARE. Possible settings are</p> <p><b>JRNL</b> Issue a message to the BBI Journal stating that AutoOPERATOR cannot listen to the queue.</p> <p><b>WTO</b> Issue a write-to-operator (WTO) message stating that AutoOPERATOR cannot listen to the queue.</p> <p><b>IGNORE</b> Take no action.</p> <p><b>ALTER</b> Alter the queue to SHARE.</p> <p>The default value is JRNL.</p>
MQEVLPRC=xxxxxxxx	<p>Specifies the name of the MQSeries Event Listener PROC.</p> <p>Only use this parameter if the MQSeries Event Listener is required. Refer to “Step 5. Setting up AutoOPERATOR for MQSeries to Co-Exist with PATROL for MQ Products” on page 35 for more information about the MQSeries Event Listener and co-existence of AutoOPERATOR and other BMC Software products that require the MQSeries event queues.</p>

The following paragraphs describe which actions are taken, depending on the parameters you specify for AutoOPERATOR for MQSeries, when AutoOPERATOR tries to automate MQSeries events.

## Generating Instrumentation Events

Instrumentation events, recognized by AutoOPERATOR by their format, MQFMT\_EVENT, are generated by queue managers. It may be possible to configure a queue manager to *not generate* instrumentation events. In this case, AutoOPERATOR will not be able to hear them and, therefore, Rules for these events will not fire.

Use the AAOPRMxx parameter MQEV=YES|NO to specify whether or not AutoOPERATOR for MQSeries will automatically alter queue managers to generate instrumentation events. If the queue manager has not been configured to generate instrumentation events and you choose the default setting of MQEV=NO; AutoOPERATOR will not hear the events, and **MQS event Rules will not fire.**

For AutoOPERATOR to hear the events, you must specify MQEV=YES. When MQEV=YES is specified, AutoOPERATOR automatically alters queue managers when it is started, to generate instrumentation events **if they had been originally disabled.**

When the MQEV parameter is left with the default setting of NO (MQEV=NO), AutoOPERATOR **will not** automatically alter queue managers to generate instrumentation events. Warning messages are issued during AutoOPERATOR connection to a queue manager to inform you about any classes of events that are not enabled.

## Encountering Queues Set to Get(Disabled)

You can specify which actions AutoOPERATOR for MQSeries will take when you have set AutoOPERATOR to listen to a queue that has been set to GET(DISABLED). Messages cannot be retrieved (MQGET) from a queue that has been set to GET(DISABLED). Therefore, AutoOPERATOR cannot listen to any messages placed into this queue.

If you have asked AutoOPERATOR to listen to a queue that is GET(DISABLED), then you can use the MQGINHIB parameter in BBPARM member AAOPRMxx to take one of the following actions with these keywords:

**JRNL** Issue a message to the BBI Journal stating that AutoOPERATOR cannot listen to the queue.

**WTO** Issue a write-to-operator (WTO) message stating that AutoOPERATOR cannot listen to the queue. The message will also be issued to the BBI Journal.

**IGNORE** Take no action.

**ALTER** Alter the queue to GET(ENABLED).

## Encountering Queues Set as NOSHARE

You can specify which actions AutoOPERATOR for MQSeries takes when you set AutoOPERATOR to listen to a queue that has been set to NOSHARE. If a queue has been set to NOSHARE, only one application can have the queue open. Therefore, if an application is putting messages onto a queue that is NOSHARE, AutoOPERATOR will not be able to open the queue for monitoring (listening) unless some action is taken.

If you have asked AutoOPERATOR to listen to a queue that is NOSHARE, you can use the MQNSHARE parameter in BBPARM member AAOPRMxx to take one of the following actions with these keywords:

**JRNL** Issue a message to the BBI Journal stating that AutoOPERATOR will not listen to the queue.

**WTO** Issue a write-to-operator (WTO) message stating that AutoOPERATOR will not listen to the queue. The message will also be issued to the BBI Journal.

**IGNORE** Take no action.

**ALTER** Alter the queue to SHARE.

**Note:** The ALTER specification affects all queues eligible for automation as specified in AAOMQLxx.

## Parameters for BBPARM Member AAOMQLxx

AAOMQLxx is a required BBPARM member used by AutoOPERATOR for MQSeries. The AAOMQLxx member contains parameters that specify which MQ queues can be monitored (or listened to) by AutoOPERATOR. You must have at least one valid specification for a queue in an AAOMQLxx member.

### Specifying Queues

Use the BBPARM member AAOMQLxx parameters TYPE, QMGR, and QUEUE to specify MQ queues that will become eligible for automation. A fourth optional parameter, OPEN, indicates how the queue should be processed. *All parameters must be typed on one line (abbreviations are available)*; the syntax is

```
TYPE(INCL|EXCL) QMGR(queue-manager-name) QUEUE(queue-name) OPEN(S, I)
```

The following table describes the AAOMQLxx parameters.

Table 5. AAOMQLxx Parameters

Parameter	Description
TYPE(INCL EXCL)	<p>Specifies whether AutoOPERATOR for MQSeries should include or exclude this queue in the set of queues that are eligible for automation. This parameter is abbreviated with 'T'. Possible values are</p> <p>INCL      Specifies that AutoOPERATOR for MQSeries should make this queue eligible for possible automation. Abbreviated with 'I', this setting is the default.</p> <p>EXCL      Specifies that AutoOPERATOR for MQSeries should not make this queue eligible for automation. This parameter is abbreviated with 'E'.</p>
QMGR(queue-manager-name)	<p>Specifies the four-character ID for a local OS/390 queue manager that AutoOPERATOR for MQSeries will monitor. Abbreviated with 'M', this parameter supports asterisk (*) and plus (+) wildcard characters.</p> <p>This is a required parameter and there is no default value. <i>You must specify a queue manager name</i> or a partial queue manager name with wildcard characters (a plus (+) represents one character, and an asterisk (*) represents one or more characters).</p>
QUEUE(queue-name)	<p>Specifies the 48-character name of the queue that AutoOPERATOR for MQSeries makes eligible for automation. Abbreviated with 'U', this parameter supports asterisk (*) and plus (+) wildcard characters.</p> <p>This is a required parameter and there is no default value. <i>You must specify a queue name</i> or a partial queue name with wildcard characters (a plus (+) represents one character, and an asterisk (*) represents one or more characters).</p>

Table 5. AAOMQLxx Parameters (Continued)

Parameter	Description
OPEN(option1,option2)	<p>Gives you control over how the queue is opened and what action to take against the messages in the opened queue. Abbreviated with 'O', the subparameters are</p> <p><b>EXCLUSIVE</b>    Open queue with MQOO_INPUT_EXCLUSIVE option. This option means that other applications will be unable to open the queue while AutoOPERATOR has it open. It also means that if another application has the queue opened, AutoOPERATOR will be unable to open it. This parameter is abbreviated with 'E'.</p> <p><b>SHARED</b>        Open queue with MQOO_INPUT_SHARED option. Abbreviated with 'S'. This option is the default.</p> <p><b>PROCESSOLD</b>    Route all existing messages found on the queue at open time through the Rule Processor so that they are automated. This option is useful for processing messages that were put on the queue while AutoOPERATOR was unable to open the queue. This parameter is abbreviated with 'P'.</p> <p><b>IGNOREOLD</b>     Ignore (do not process) any messages found on the queue at open time. Abbreviate with 'I', this option is the default.</p> <p>Example 1: OPEN(EXCLUSIVE,PROCESSOLD) or O(E,P) Indicates that the queue is to be opened with the QOO_INPUT_EXCLUSIVE option and all existing messages found on the queue at open time should be rerouted through the Rule Processor for automation.</p> <p>Example 2: OPEN(SHARED,IGNOREOLD) or O(S,I) Indicates that the queue is to be opened with the MQOO_INPUT_SHARED option and all existing messages found on the queue at open time should be ignored.</p>

You may specify multiple sets of these three parameters. Continuations are not supported.

## Example of Specifying Queues in AAOMQLxx

BMC Software recommends that you use Figure 5 as a model for your specifications.

```

-----
* PARAMETER(DEFAULT)  VALUES ----- DESCRIPTION -----
* -----
*
* TYPE(INCL)
*           INCL  TYPE ABBREVIATED WITH 'T'.
*                NCLUDE THIS QUEUE FOR RULES (INCL) OR
*                ABBREVIATE WITH 'I'.
*                EXAMPLE: T(I)
*           EXCL  EXCLUDE (EXCL) THIS QUEUE FOR RULES.
*                ABBREVIATE WITH 'E'.
*                EXAMPLE: T(E)
*
* QMGR
*           QMGR ABBREVIATED WITH 'M'.
*                NAME OF MQSERIES QUEUE MANAGER. NO
*                DEFAULT, THIS PARAMETER MUST BE CODED.
*                EXAMPLE: M(CSQ1)
*
* QUEUE
*           QUEUE ABBREVIATED WITH 'U'.
*                NAME OF MQSERIES QUEUE. NO DEFAULT,
*                THIS PARAMETER MUST BE CODED.
*                EXAMPLE: U(SYSTEM ADMIN. QMGR. EVENT)
*
* OPEN(S, I)
*           EXCLUSIVE  QUEUE OPENED WITH MQOO_INPUT_EXCLUSIVE
*                OPTION.  ABBREVIATE WITH 'E'.
*
*           SHARED  QUEUE OPENED WITH MQOO_INPUT_SHARED
*                OPTION. ABBREVIATE WITH 'S'. THIS IS A DEFAULT OPTION.
*
*           PROCESSOLD  ROUTE ALL EXISTING MESSAGES FOUND ON
*                THE QUEUE AT OPEN TIME THROUGH THE RULE
*                PERFORMED ON THEM  ABBREVIATE WITH 'P'.
*
*           IGNOREOLD  IGNORE (DO NOT PROCESS) ANY EXISTING
*                MESSAGES FOUND ON THE QUEUE AT OPEN TIME.
*                ABBREVIATE WITH 'I'. THIS IS A DEFAULT
*                OPTION.
*
*                EXAMPLE 1: OPEN(EXCLUSIVE, PROCESSOLD) OR
*                O(E, P) INDICATES THE QUEUE IS TO BE
*                OPENED WITH THE MQOO_INPUT_EXCLUSIVE
*                OPTION AND ALL EXISTING MESSAGES FOUND ON
*                THE QUEUE AT OPEN TIME SHOULD BE ROUTED
*                THROUGH THE RULE PROCESSOR FOR
*                AUTOMATION.
*
*                EXAMPLE 2: OPEN(SHARED, IGNOREOLD) OR
*                O(S, I) INDICATES THE QUEUE IS TO BE
*                OPENED WITH THE MQOO_INPUT_SHARED OPTION
*                AND ALL EXISTING MESSAGES FOUND ON THE
*                QUEUE AT OPEN TIME SHOULD BE IGNORED.
*
* -----
* MQSERIES SYSTEM EVENT QUEUES
* -----
* TYPE(INCL) QMGR(*) QUEUE(SYSTEM ADMIN. *. EVENT)
* -----
* REMAINDER OF QUEUES *DO NOT OPEN*
* -----
* TYPE(EXCL) QMGR(*) QUEUE(*)

```

Figure 5. Example of Coding in AAOMQLxx Member

**Note:** In the list, you must separately list the queues eligible for automation from the queues that will be excluded from automation.

For each MQSeries queue manager, AutoOPERATOR matches the list of existing queues to the contents of the active AAOMQLxx member. It sequentially scans the AAOMQLxx

member and evaluates the QMGR(queue manager name) and QUEUE(queue name) specifications until a match is found.

When a match is found AutoOPERATOR will either include or exclude the queue based on the TYPE(INCL|EXCL) setting. If two (or more) statements overlap or conflict, the first definition encountered will be used.

***Any queue not included in the AAOMQLxx member will not be eligible for automation.*** If you code a Rule for an excluded queue, the Rule will never fire.

The AAOMQLxx is processed

- At BBI-SS PAS startup
- When the BBI commands .RESET MQ or .RESET QM are issued

For more information about BBI commands for AutoOPERATOR for MQSeries, refer to “MAINVIEW AutoOPERATOR for MQSeries BBI Commands” on page 37.

**Note:** While AutoOPERATOR has a queue opened, no other application can open exclusively or delete that queue.



---

## Chapter 4. Automating MQSeries Events

This chapter describes

- How MAINVIEW AutoOPERATOR for MQSeries automates MQSeries events with Rules
- Two procedures that show how to create Rules for MQSeries events
- Variables that are available to support the MQS event type
- Panel and field definitions for the Rule Processor Selection Criteria and Action Specification panels for the MQS event type
- The event type MQS in the Automation Reporter Rules record

For more information about creating Rules and Rules Sets, see Chapter 13, “Implementing AutoRULE” in the *MAINVIEW AutoOPERATOR Customization Guide*.

---

### How MAINVIEW AutoOPERATOR for MQSeries Automates MQSeries Events

AutoOPERATOR uses Rules to perform automation for various system events such as OS/390 messages, commands, and ALERTs. Rules can also be used to provide automated responses to MQSeries events. This section describes important concepts about AutoOPERATOR Rules and MQSeries events.

#### What a Rule Is

A Rule is a two-part statement: when the conditions of the first part of the statement are met (the selection criteria), the automation actions specified in the second part are performed.

When AutoOPERATOR is installed, you can create Rules that provide automated actions such as issuing commands, creating ALERTs, or rerouting messages in response to OS/390 system events. AutoOPERATOR Rules listen for these events. When the event occurs, the Rule fires and the automation action is taken.

When MAINVIEW AutoOPERATOR for MQSeries is installed, you can create Rules for MQSeries events. The MQSeries events can originate either from queues within OS/390 or from non-OS/390 platforms. AutoOPERATOR listens for MQSeries messages and recognizes them as event type MQS. Enabled Rules listen for these MQS messages and when the conditions of the Rule are met, the Rule fires to provide automated responses to the message.

**Note:** For MAINVIEW AutoOPERATOR for MQSeries to listen for and automate events from non-OS/390 queue managers (such as those running on Windows NT, UNIX, OS/2, and OS/400 operating systems), you must define connectivity and activate communications between OS/390 and non-OS/390 queue managers as described in “Step 3. Defining Connectivity” on page 22.

## What MQS Events Are

The AutoOPERATOR Rule Processor listens for MQSeries messages and recognizes them as event type MQS. These messages (or events) can originate from an OS/390 or non-OS/390 queue manager. There are additional customization steps required for listening to MQS events from non-OS/390 queue managers.

AutoOPERATOR distinguishes between different MQSeries messages by their format in the message descriptor. The following table lists these message formats and description:

Table 6. Message Format and Description

Message Format	Description
MQFMT_NONE	No format
MQFMT_ADMIN	Command server request/reply message
MQFMT_CHANNEL_COMPLETED	Channel has ended
MQFMT_CICS	CICS Information Message
MQFMT_COMMAND_1	Type 1 command reply message
MQFMT_COMMAND_2	Type 2 command reply message
MQFMT_DEAD_LETTER_HEADER	Dead Letter Message
MQFMT_EVENT	Event Message
MQFMT_IMS	IMS Information Message
MQFMT_IMS_VAR_STRING	IMS Variable String Message
MQFMT_MD_EXTENSION	Message Descriptor Extension
MQFMT_PCF	Programmable Command Format
MQFMT_REF_MSG_HEADER	Reference Message Header
MQFMT_STRING	Character String
MQFMT_TRIGGER	Trigger Message
MQFMT_WORK_INFO_HEADER	Work Information Message
MQFMT_XMIT_Q_HEADER	Transmission Queue Message
MQFMT_RF_HEADER	Rules and formatting header
MQFMT_RF_HEADER_2	Rules and formatting header version 2

The origin of the message may be from an OS/390 queue manager or a non-OS/390 queue manager. When you create a Rule, you must specify the format of the message. A single Rule can recognize only one specified format. However, by specifying USER, the Rule can fire for any format except MQFMT\_EVENT.

## What Event Messages Are

MQSeries event messages (or instrumentation events) are provided by MQSeries and allow AutoOPERATOR, through Rules, to monitor queue manager conditions, performance

conditions, and channel conditions. MQSeries events originate from the three administration queues; SYSTEM.ADMIN.QMGR.EVENT queue, the SYSTEM.ADMIN.PERFM.EVENT queue, and the SYSTEM.ADMIN.CHANNEL.EVENT queue. However, AutoOPERATOR is not restricted to receiving those messages from those queues only.

You can create a Rule to listen for MQSeries events (format MQFMT\_EVENT) and subsequently take automation action. The automation action the Rule performs could be to create an ALERT to warn the operator or invoke a command to copy the data in the queue to a data set.

## What Non-Event Messages Are

Non-Event messages are any MQSeries messages (except messages with format MQFMT\_EVENT) that arrive on a queue. For example, you could have an accounts payable (AP) application that sends data to and from MQSeries user queues. You can create Rules to fire based on the data from the Message Buffer or the various MQSeries structures that make up the message.

## Rule Processor MQSeries Variables

MAINVIEW AutoOPERATOR provides special variables for MQSeries Rules. Using these variables, the Rule can fire based on any of many possible values that are within an MQSeries message, whether it is an instrumentation event or an application message. These variables are created from MQSeries structures and data in the message buffer, and can be used on panels that make up the Rule Processor user interface. These variables can be used to determine if the Rule should fire and, if so, what resources, if any, should be acted upon by the Rule. The following list describes general properties for variables available in MQSeries Rules:

Variable names consist of a

- Prefix name (for example, IMFQ)
- Structure name (for example, MD, DLH)
- Field name (for example, MsgId or CorrelId in the message descriptor)

Each node of the variable name is delimited by an underscore (\_)

Some field-name variables have associated constant names defined for them by MQSeries. In these cases, the variables contain the constant name rather than the value within the data structure. For example: in the case of a Dead Letter message, the variable IMFQ\_MD\_FORMAT, which contains the format field from the Message Descriptor, is set to MQFMT\_DEAD\_LETTER\_HEADER. Likewise, the variable IMFQ\_MD\_VERSION, which contains the version of the message descriptor, is set to MQMD\_VERSION\_1 instead of x'1'. The following rules apply for variables set by AutoOPERATOR for MQSeries:

- If the value for a character field does not have a defined constant value, the field value is used.
- If the value for a numeric field that has a defined range does not have a defined constant, the numeric value converted to zoned decimal is used.
- Variables for fields that are defined as hexadecimal fields (MQBYTExx in MQSeries) are not converted and are set 'as is.'

- Variables that are created for flag fields in an MQSeries structure may contain multiple constant values, delimited by blanks. For example, the variable for the Message Descriptor Report field, IMFQ\_MD\_REPORT, can have several values. The resulting value might look like this: 'MQRO\_COA MQRO\_COD'. Any other valid combination could be seen as well.
- Variables that are created for MQSeries fields where the value has more than one defined constant will contain both constants. For example, the variable IMFQ\_MD\_PUTAPPLTYPE, which contains the PutApplType value from the message descriptor, would contain the value 'MQAT\_OS390 MQAT\_MVS' if the message originated on an OS/390 system. This result is because both constants contain the value '2'.

## Enabling MQS Events

Each MQS event falls into a category of events that must be enabled within the queue manager before AutoOPERATOR Rules can *hear* them.

Refer to the *IBM MQSeries Command Reference* for information about enabling these categories. During MAINVIEW AutoOPERATOR for MQSeries initialization, warning messages can be issued for each event category that is not enabled, or AutoOPERATOR can enable them automatically.

To enable MQS events so that AutoOPERATOR Rules can hear them, ensure that the BBPARM member AAOMQLxx lists the names of all the queues for which you want to hear events.

For information about listing the queue names in AAOMQLxx, refer to “Specifying Queues” on page 42.

**Note:** If you use the default AAOMQLxx member, AutoOPERATOR listens for only MQFMT\_EVENT messages that originate from administration queues such as the SYSTEM.ADMIN.QMGR.EVENT queue, the SYSTEM.ADMIN.PERFM.EVENT queue, and the SYSTEM.ADMIN.CHANNEL.EVENT queue. To listen for MQFMT\_EVENT and User messages on any other queue, you must modify the AAOMQLxx member of BBPARM and include the names (or name pattern) of the queues to be monitored.

---

## Creating Rules for Event Messages and Non-Event Messages

This section shows how to use the Rule Processor application panels to create Rules for MQSeries event messages (format MQFMT\_EVENT) or non-event messages (format other than MQFMT\_EVENT). For more information about Rules, other events you can create Rules for, and examples, refer to the *AutoOPERATOR Basic Automation Guide*.

### Creating a Rule for an MQFMT\_EVENT Message

This section shows you how to create an AutoOPERATOR Rule that will listen and fire for an MQSeries event that occurs when a specific queue belonging to the queue manager (named CSQ1 in this example) becomes full. When the queue becomes full, the queue manager issues an instrumentation event (format MQFMT\_EVENT) with a type of Q\_FULL.

By following the example, you will learn how to create a Rule to listen for this event and when it fires, the Rule will update a variable that indicates what happened.

To create the Rule, you must enter the correct Selection Criteria information. The following table describes the information you need to enter:

You need to know	Which is	To enter it on this panel
The Rule event type	MQS	Rule Processor Detail Control
The queue manager name	CSQ1, in this example	Selection Criteria - MQS
The message format	MQFMT_EVENT	Selection Criteria - MQS
The MQSeries event type	Q_FULL	Selection Criteria - MQS
The name of the queue for which the event occurred	PROD.AP	Variable Dependencies

For more information about creating Rules and Rules Sets, see Chapter 13, “Implementing AutoRULE” in the *MAINVIEW AutoOPERATOR Customization Guide*.

## Example: Creating an MQFMT\_EVENT Rule

To create a Rule that fires whenever a Q\_FULL event occurs for the PROD.AP queue:

1. From the Automation Control panel, type the E line command, for (E)nable, and press Enter to enable the Rule Set to which you want to add the Rule. You can add Rules only to enabled Rule Sets; see Figure 6.

```

BMC Software ----- Automation Control ----- AutoOPERATOR
COMMAND ==> TGT ==> SYSE
Primary commands: Add, Statshow, Cmdshow DATE --- 00/11/15
TIME --- 15:56:01

Automation Status ==> ACTIVE (Active, Inactive)
Automation Strategy ==> INDIVIDUAL (Individual, All, First, Qualified)
Honor MPF Suppression ==> YES (NO/YES)

Automation Library
LC CMDS --- (S)elect, (E)nable, (D)isable, (T)est, (S)ave
(M)ove, (B)efore or (A)fter, (F)ilter Criteria
LC Rule-Set Status Rules Fired Filtered Date Time Strategy
___ AAORULMI DI SABLED N/A N/A N/A N/A N/A
___ AAORULQ2 DI SABLED N/A N/A N/A N/A N/A
___ AAORULXX DI SABLED N/A N/A N/A N/A N/A
___ AAORUL00 DI SABLED N/A N/A N/A N/A N/A
___ AAORUL01 DI SABLED N/A N/A N/A N/A N/A
___ AAORUL02 DI SABLED N/A N/A N/A N/A N/A
___ RULBCPSM DI SABLED N/A N/A N/A N/A N/A
e_ RULMQS01 DI SABLED N/A N/A N/A N/A N/A
***** END OF DATA *****

```

Figure 6. Creating an MQFMT\_EVENT Rule: Automation Control Panel (Example 1)

For more information about creating Rules and Rules Sets, see Chapter 13, “Implementing AutoRULE” in the *MAINVIEW AutoOPERATOR Customization Guide*.

2. To select the Rule Set, type the S line command, for (S)elect, in the LC field next to the Rule Set name.

```

BMC Software ----- Automation Control ----- RULMQS01 ENABLED
COMMAND ==> TGT ==> SYSE
Primary commands: Add, Statshow, Cmdshow DATE --- 00/11/15
TIME --- 15:56:14

Automation Status ==> ACTIVE (Active, Inactive)
Automation Strategy ==> INDIVIDUAL (Individual, All, First, Qualified)
Honor MPF Suppression ==> YES (NO/YES)

Automation Library
LC CMDS --- (S)elect, (E)nable, (D)isable, (T)est, (S)ave
(M)ove, (B)efore or (A)fter, (F)ilter Criteria
LC Rule-Set Status Rules Fired Filtered Date Time Strategy
___ AAORULMQ ENABLED 83 0 72,180 15-NOV-00 11:50:08 ALL
s_ RULMQS01 ENABLED 79 0 1 15-NOV-00 15:56:10 ALL
___ AAORULBA DI SABLED N/A N/A N/A N/A N/A
___ AAORULBB DI SABLED N/A N/A N/A N/A N/A
___ AAORULBC DI SABLED N/A N/A N/A N/A N/A
___ AAORULBD DI SABLED N/A N/A N/A N/A N/A
___ AAORULBE DI SABLED N/A N/A N/A N/A N/A
___ AAORULBF DI SABLED N/A N/A N/A N/A N/A
___ AAORULBG DI SABLED N/A N/A N/A N/A N/A
___ AAORULBH DI SABLED N/A N/A N/A N/A N/A
___ AAORULBP DI SABLED N/A N/A N/A N/A N/A
___ AAORULBQ DI SABLED N/A N/A N/A N/A N/A
___ AAORULBR DI SABLED N/A N/A N/A N/A N/A

```

The Rule Set Overview panel is displayed. Figure 7 shows an example of a Rule Set named RULMQS01 and the Rules it contains.

3. To add a new Rule, use the ADD primary command on the command line.

```

BMC Software ----- Rule Set Overview ----- AutoOPERATOR
COMMAND ==> ADD                                     TGT --- SYSE
Rule Set ID: RULMQS01   Ruleset Strategy ==> ALL       DATE --- 00/11/15
Primary commands: Add, Save, Sort, Unsort, Reset, Filter  TIME --- 15:56:28
LC CMDS --- (S)elect, (E)nable, (D)isable, (T)est, (DE)lete, (I)nsert
              (C)opy/(CC)opy, (M)ove/(MM)ove, (B)efore or (A)fter, (R)peat
Sort Criterion:                                       right/left

LC  Rule-id Stat Text-id      Type  Fired EXEC      Changed      ID
---  ---  ---  ---  ---  ---  ---  ---  ---
___  MQQDPHI2 DIS Q_DEPTH_HIGH  MQS    0 MQVARS 2 97/01/24 20:30 JDB3
___  MQQDPLOW ENA Q_DEPTH_LOW   MQS    0 MQVARS 1 00/11/15 15:56 BAOJDB4
___  MQQDPLO2 ENA Q_DEPTH_LOW   MQS    0 MQVARS 2 00/11/15 15:56 BAOJDB4
___  MQSVINTH ENA Q_SERVICE_INTERV MQS    0 MQVARS 1 00/11/15 15:56 BAOJDB4
___  MQSVINT2 ENA Q_SERVICE_INTERV MQS    0 MQVARS 2 00/11/15 15:56 BAOJDB4
___  MQSVINOK ENA Q_SERVICE_INTERV MQS    0 MQVARS 1 00/11/15 15:56 BAOJDB4
___  MQSVINO2 ENA Q_SERVICE_INTERV MQS    0 MQVARS 2 00/11/15 15:56 BAOJDB4
___  MQPUTENA ENA PUT_INHIBITED  MQS    0 MQVER MQ 97/02/19 14:51 JDB3
___  MQPUTEN2 SUS PUT_INHIBITED  MQS    0 MQVER MQ 97/02/19 14:51 JDB3
___  MQQFULL  ENA Q_FULL        MQS    0 MQVER MQ 97/02/19 20:49 JDB3
___  QUEUE1   ENA              MQS    0 VARDISP 00/11/15 15:56 BAOJDB4
***** END OF DATA *****

```

Figure 7. Creating an MQFMT\_EVENT Rule: Rule Set Overview Panel (Example 1)

The Rule Processor Detail Control panel is displayed.

```

BMC Software ----- Rule Processor Detail Control ----- AutoOPERATOR
COMMAND ==>
                                               TGT --- SYSE
                                               TIME --- 15:57:17
                                               DATE --- 00/11/15

The following options are displayed in sequence, or may
be selected by entering the two-character code

    S1 - Selection Criteria           A1 - Action Specification
    SV - Variable Dependencies        AA - Alert Action(s) I
                                       AD - Alert Action(s) II

Rule ID      ==>
Event Type   ==>      Type of event ( ? for list)
Initial Mode ==> ENABLED (ENABLED/DISABLED/TEST)

Criteria match rate:
If matched   ==>      (Maximum # times matched within INTERVAL, 1-100)
in seconds   ==>      (Interval length, 1-99999 seconds)
then status  ==>      (SUSPEND, DISABLE)

Application information:
Group        ==>      Function      ==>      Code      ==>
Author       ==> BMCUSER Description  ==>
Last Modified by on at

Press ENTER to continue, END to apply changes, CANCEL to cancel changes

```

Figure 8. Creating an MQFMT\_EVENT Rule: Rule Processor Detail Control Panel (Example 1)

4. Type the following information:

- A unique name in the Rule ID field (for example, APQHI); see Figure 9.
- MQS in the Event Type field; see Figure 9.

**Note:** The Event Type MQS will be accepted only if the MAINVIEW AutoOPERATOR for MQSeries key is present. If the key is not present, you will receive a short message in the upper right corner of the panel.

```

BMC Software ----- Rule Processor Detail Control ----- AutoOPERATOR
COMMAND ==>>                                         TGT --- SYSE
                                                    TIME --- 15:58:13
                                                    DATE --- 00/11/15

The following options are displayed in sequence, or may
be selected by entering the two-character code

    S1 - Selection Criteria          A1 - Action Specification
    SV - Variable Dependencies      AA - Alert Action(s) I
                                    AD - Alert Action(s) II

Rule ID      ==>> APQHI
Event Type   ==>> MQS      Type of event ( ? for list)
Initial Mode ==>> ENABLED (ENABLED/DISABLED/TEST)
Criteria match rate:
If matched   ==>>          (Maximum # times matched within INTERVAL, 1-100)
in seconds   ==>>          (Interval length, 1-99999 seconds)
then status  ==>>          (SUSPEND, DISABLE)

Application information:
Group        ==>>          Function      ==>>          Code      ==>>
Author       ==>> BMCUSER Description  ==>>
Last Modified by on at

Press ENTER to continue, END to apply changes, CANCEL to cancel changes
    
```

Figure 9. Creating an MQFMT\_EVENT Rule: Rule Processor Detail Control Panel (Example 2)

5. Press Enter.

The Selection Criteria - MQS panel is displayed.

```

BMC Software ----- Selection Criteria - MQS ----- AutoOPERATOR
COMMAND ==>>                                         TGT --- SYSE

Rule-set == RULMQS01          Rule-id == APQHI

Queue Identification:                                     (1 to 12 Queue Managers)
Manager(s) ==>>
Queue Id    ==>>

Message Identification:
Format      ==>>          (Value from MD Format field)
Event Type  ==>>          (Enter ? for help)

Msgid       ==>> ___ : ___ ___ _____
CorrelId    ==>> ___ : ___ ___ _____
Msg Buffer   ==>> ___ : ___ ___ _____

Press ENTER to continue, END return to Detail Control, CANCEL to cancel changes
    
```

Figure 10. Creating an MQFMT\_EVENT Rule: Selection Criteria - MQS Panel (Example 1)

6. To enter the selection criteria for this event:

- Enter your queue manager name in the Manager(s) field. In this example, CSQ1 is the queue manager name; see Figure 11.
- In the Format field, type **MQFMT\_EVENT**.

This is the MQSeries message format. When you create Rules for messages with format MQFMT\_EVENT, you can also use the Event Type field to specify the type of MQSeries event for which you can listen. This is optional. You cannot use the MsgId, CorrelId or Msg Buffer fields for messages of format MQFMT\_EVENT.

- For a selectable list of MQ instrumentation Events, type **?**.

```
BMC Software ----- Selection Criteria - MQS ----- AutoOPERATOR
COMMAND ==>                                     TGT --- SYSE

          Rule-set === RULMQS01          Rule-id === APQHI

Queue Identification:                            (1 to 12 Queue Managers)
Manager(s)  ==>  CSQ1
Queue Id    ==>

Message Identification:
Format      ==>  MQFMT_EVENT                (Value from MD Format field)
Event Type  ==>  ?                          (Enter ? for help)

          Sub  Len  Op  Value
MsgId      ==>  ___ : ___ ___ _____
CorrelId   ==>  ___ : ___ ___ _____
Msg Buffer  ==>  ___ : ___ ___ _____

Press ENTER to continue, END return to Detail Control, CANCEL to cancel changes
```

Figure 11. Selection Criteria - MQS Panel (Example 2)

The MQSeries Event Types panel is displayed. See Figure 12.

This panel shows a scrollable list of (MQS) Event Types.

```
BMC Software ----- MQSeries Event Types ----- AutoOPERATOR
COMMAND ==>>
Line commands: (S)elect, (B)rowse
LC Queue      Type
___ QMGR      Alias_base_Q_type_error
___ QMGR      Get_inhi_bited
___ QMGR      Put_inhi_bited
___ QMGR      Q_type_error
___ QMGR      Remote_Q_name_error
___ QMGR      Xmit_Q_type_error
___ QMGR      Xmit_Q_usage_error
___ QMGR      Unknown_alias_base_Q
___ QMGR      Unknown_object_name
___ QMGR      Unknown_remote_Q_mgr
___ Channel  Channel_acti_vated
___ Channel  Channel_not_acti_vated
___ Channel  Channel_started
___ Channel  Channel_stopped
B Perform  Q_depth_hi gh
___ Perform  Q_depth_lo w
___ Perform  Q_ful l
___ Perform  Q_servi ce_i nterval_hi gh
___ Perform  Q_servi ce_i nterval_ok
```

Figure 12. Example of an MQSeries Event Types Panel

**Note:** You can type the B (for (B)rowse) for more than one Type. The additional selected panels will be displayed one at a time. To proceed from one panel to the next, press PF3.

7. To browse descriptive information about a specific Type, type B in the LC (line command) column.

A help panel is displayed. See Figure 13.

```
BMC Software ----- MQSeries Event Types ----- AutoOPERATOR
COMMAND ===>

Type          Q_DEPTH_HIGH

Description   An MQPUT or MQPUT1 call caused the queue depth to be incremented
              to or above the queue depth high limit.

Variables     IMFQ_EVENT_QMGRNAME      : Name of Queue Manager that generated
              the event, whether local or remote.

              IMFQ_EVENT_QNAME      : Name of user or system queue that
              caused the event.

              IMFQ_EVENT_TIMESINCERESET: Time (in seconds) since the
              statistics were last reset.

              IMFQ_EVENT_HIQDEPTH    : Max number of message on the queue
              since the statistics were last reset.

              IMFQ_EVENT_MSGENQCOUNT : Number of messages enqueued since
              statistics were last reset.

              IMFQ_EVENT_MSGDEQCOUNT : Number of messages dequeued since
              statistics were last reset.

              QMGR Attribute        : PERFMEV
```

Figure 13. Example of a Help Panel for Q\_DEPTH\_HIGH

8. To return to the previous panel, press PF3.

The MQSeries Event Types panel is displayed again (see Figure 12 on page 56).

9. To select an Event Type, type S in the LC column next to the Event Type on the MQSeries Event Types panel and press Enter.



The Action Specification - MQS panel is displayed.

```

BMC Software ----- Action Specification - MQS ----- AutoOPERATOR
COMMAND ==>                                           TGT --- SYSE

          Rule-set === RULMQS01           Rule-id === APQHI

Automation Actions:
Journal          ==>
EXEC Name/Parms ==>
Cmd (Type MQ ) ==>

Set Variable     ==>                               ==>
Notify           ==>                               Outboard Pager ID
Info            ==>

DOM Id          ==>                               Delete Operator Message
Issue WTO Msg   ==>

MQSeries Automation Actions:
Keep Message     ==> YES   (Yes or NO)   Remove DLH ==>   (Yes or No)
Destination Que ==>
Destination QMgr ==>

Press ENTER to continue, END return to Detail Control, CANCEL to cancel changes

```

Figure 16. Creating an MQFMT\_EVENT Rule: Action Specification - MQS Panel (Example 1)

12. Use the Action Specification - MQS panel to specify the action for the Rule to take when it fires. For this example, the action taken is to set the value of the STAT.PROD.AP variable to FULL and retain the message in the queue.

To specify the action for this Rule:

- In the Set Variable fields, type **STAT. &IMFQ\_EVENT\_BASEQNAME** and **FULL**.
- Verify that YES is in the Keep Message field (this tells AutoOPERATOR to leave the instrumentation event message in the event queue).

```

BMC Software ----- Action Specification - MQS ----- AutoOPERATOR
COMMAND ==>                                           TGT --- SYSE

          Rule-set === RULMQS01           Rule-id === APQHI

Automation Actions:
Journal          ==>
EXEC Name/Parms ==>
Cmd (Type MQ ) ==>

Set Variable     ==> STAT. &IMFQ_EVENT_BASEQNAME   ==> FULL
Notify           ==>                               Outboard Pager ID
Info            ==>

DOM Id          ==>                               Delete Operator Message
Issue WTO Msg   ==>

MQSeries Automation Actions:
Keep Message     ==> YES   (Yes or NO)   Remove DLH ==>   (Yes or No)
Destination Que ==>
Destination QMgr ==>

Press ENTER to continue, END return to Detail Control, CANCEL to cancel changes

```

Figure 17. Action Specification - MQS Panel (Example 2)

13. Press Enter.

The Alert Action(s) I -MQS panel is displayed.

```
BMC Software ----- Alert Action(s) I - MQS ----- AutoOPERATOR
COMMAND ==> TGT --- SYSE
      Rule-set == RULMQS01      Rule-id == APQHI
Function ==> (ADD, DELETE, DELETED)
Key ==>
Text ==>
Queues ==> Alert Queue Name(s)
Priority ==> (CRIT, MAJ, MIN, WARN, INFO, CLEAR)
Color ==> RED, PINK, YEL, DKBL, LTBL, GRE, WHI
PCMD ==>
System ==> Return to target after PCMD
EXEC ==> Follow-up EXEC
Help ==> Associated HELP Panel
Targets ==> Target System
Udata ==> User Data
Origin ==> Origin
User ==> Userid
Alarm ==> Sound Alarm (YES/NO)
Press ENTER to continue, END return to Detail Control, CANCEL to cancel changes
```

14. The Rule in this example will not issue an ALERT; press Enter.

The Alert Action(s) II - MQS panel is displayed.

```
BMC Software ----- Alert Action(s) II - MQS ----- AutoOPERATOR
COMMAND ==> TGT --- SYSE
      Rule-set == RULMQS01      Rule-id == APQHI
Retain ==> Yes/No
Escalate Direction ==> Up/Down
      Interval ==>
      ==>
      ==>
      ==>
      ==>
      ==>
      Disposition ==> Keep/Delete
      EXEC ==>
Press ENTER to continue, END return to Detail Control, CANCEL to cancel changes
```

15. The Rule in this example will not issue an ALERT; press Enter.

The Rule Processor Detail Control panel is redisplayed.

```

BMC Software ----- Rule Processor Detail Control ----- AutoOPERATOR
COMMAND ==>
TGT --- SYSE
TIME --- 18:24:24
DATE --- 00/11/15

The following options are displayed in sequence, or may
be selected by entering the two-character code

S1 - Selection Criteria           A1 - Action Specification
SV - Variable Dependencies       AA - Alert Actions(s)

Rule ID      ==> APQHI
Event Type   ==> MQS           Type of event ( ? for list)
Initial Mode ==> ENABLED      (ENABLED/DISABLED/TEST)

Criteria match rate:
If matched   ==>             (Maximum # times matched within INTERVAL, 1-100)
in seconds   ==>             (Interval length, 1-99999 seconds)
then status  ==>             (SUSPEND, DISABLE)

Application information:
Group        ==> MQ           Function      ==> MONTRMQ Code ==> R2
Author       ==> BMCUSER     Description   ==> MONITOR AP QUEUES
Last Modified by on at


```

Figure 18. Rule Processor Detail Control Panel (Example 3)

16. Press PF3 to apply changes. The Rule Set Overview Panel is redisplayed.

```

BMC Software ----- Rule Set Overview ----- AutoOPERATOR
COMMAND ==>
TGT --- SYSE
DATE --- 00/11/16
TIME --- 18:32:54

Rule Set ID: RULMQS01   Ruleset Strategy ==> ALL
Primary commands: Add, Save, Sort, Unsort, Reset, Filter
LC CMDS --- (S)elect, (E)nable, (D)isable, (T)est, (DE)lete, (I)nsert
              (C)opy/(CC)opy, (M)ove/(MM)ove, (B)efore or (A)fter, (R)peat
Sort Criterion:
right/left

LC  Rule-id Stat Text-id      Type   Fired EXEC      Changed      ID
---  ---  ---  ---  ---  ---  ---  ---  ---
___ MQQDPL0W ENA Q_DEPTH_LOW   MQS    0 MQVARS 1 00/11/15 15:56 BAOJDB4
___ MQQDPL02 ENA Q_DEPTH_LOW   MQS    0 MQVARS 2 00/11/15 15:56 BAOJDB4
___ MQSVINTH ENA Q_SERVICE_INTERV MQS    0 MQVARS 1 00/11/15 15:56 BAOJDB4
___ MQSVINT2 ENA Q_SERVICE_INTERV MQS    0 MQVARS 2 00/11/15 15:56 BAOJDB4
___ MQSVINOK ENA Q_SERVICE_INTERV MQS    0 MQVARS 1 00/11/15 15:56 BAOJDB4
___ MQSVIN02 ENA Q_SERVICE_INTERV MQS    0 MQVARS 2 00/11/15 15:56 BAOJDB4
___ MQPUTENA ENA PUT_INHIBITED  MQS    0 MQVER MQ 97/02/19 14:51 JDB3
___ MQPUTEN2 SUS PUT_INHIBITED  MQS    0 MQVER MQ 97/02/19 14:51 JDB3
___ MQQFULL  ENA Q_FULL        MQS    0 MQVER MQ 97/02/19 20:49 JDB3
___ QUEUE1   ENA                MQS    0 VARDISP 00/11/15 15:56 BAOJDB4
___ APQHI    ENA Q_DEPTH_HIGH   MQS    0          00/11/15 18:32 BAOJDB4
***** END OF DATA *****

```

Figure 19. Rule Set Overview Panel (Example2)

**Note:** The Text-id for the new Rule APQHI is Q\_DEPTH\_HIGH.

The Rule is enabled and will begin to fire when the Q\_DEPTH\_HIGH event occurs for the queue PROD.AP within the CSQ1 queue manager. However, at this point the Rule is not yet saved to disk.

17. To save the Rule, type the Save primary command on the command line.

If you do not type the Save primary command, the following warning is displayed when you press PF3:

```
BMC Software ----- Confirm Rule Set Modifications ----- AutoOPERATOR
COMMAND ==>>> TGT --- SYSE
+-----+
+ WARNING! Changes made to Rule Set RULMQS01 have not been saved. Those +
+ changes were one or more of the following: +
+ +
+ o A Rule was changed. +
+ o The status of a Rule was modified. +
+ o A Rule was added, deleted, inserted, or copied. +
+ o A Rule was moved. +
+ o The individual Rule Set strategy changed. +
+ +
+ Please do one of the following: +
+ +
+ - Enter SAVE to save RULMQS01 to the BBI PARM dataset. +
+ - Enter NOSAVE to exit WITHOUT saving RULMQS01 to the BBI PARM dataset. +
+ - Press END to return to Rule Set Overview. +
+-----+
```

Figure 20. Confirming Rule Set Modifications Panel (Example 1)

The options are to

- Type SAVE to save the Rule
- Type NOSAVE to cancel saving the newly created Rule to disk, leaving the memory copy intact
- Press PF3 to return to the Rule Set Overview panel

## Creating a Rule for a Non-MQFMT\_EVENT Message

This section shows you how to create an AutoOPERATOR Rule that will listen for and fire when an MQSeries message (with the format MQFMT\_STRING and the message CLOSE ALL included in the first 255 bytes of the message text) is written to the PROD.AP.QUEUE queue.

**Note:** Be aware that the variable IMFTEXT, which is passed to an EXEC when invoked from a Rule, has the same limitations as the MSG Buffer of the Selection Criteria Panel, 255 bytes. For more information regarding IMFTEXT, see Chapter 4 in the *MAINVIEW AutoOPERATOR Advanced Automation Guide for REXX EXECs* and the *MAINVIEW AutoOPERATOR Advanced Automation Guide for CLIST EXECs*. To access data in the message beyond 255 bytes, an EXEC must be used. Using an EXEC, you can access up to 32,767 bytes of the message content.

By following the example, you will learn how to create a Rule to listen for this message. When the Rule fires, it will update a variable that indicates what happened.

To create the Rule, you must enter the correct Selection Criteria information. The following table describes the information you need to enter.

You need to know	Which is	To enter it on this panel
The Rule event type	MQS	Rule Processor Detail Control
The queue manager name	CSQ1, In this example	Selection Criteria - MQS
The message format	MQFMT_STRING	Selection Criteria - MQS
The name of the queue to which the message was PUT	PROD.AP.QUEUE	Selection Criteria - MQS
The message content	CLOSE ALL	Selection Criteria - MQS

## Example: Creating a Non-MQFMT\_EVENT Message Rule MQSeries Message

To create a Rule that fires when a MQSeries message is written to the PROD.AP.QUEUE queue with a message text of CLOSE ALL:

1. From the Automation Control panel, type the E line command, for (E)nable, and press Enter to enable the Rule Set you want to add the Rule to. You can add Rules only to enabled Rule Sets; see Figure 21.

```

BMC Software ----- Automation Control ----- AutoOPERATOR
COMMAND ==>                                     TGT ==> SYSE
Primary commands: Add                             DATE --- 00/11/16
                                                    TIME --- 14:14:39

Automation Status   ==> ACTIVE                   (Active, Inactive)
Automation Strategy ==> INDIVIDUAL              (Individual, All, First, Qualified)
Honor MPF Suppression ==> NO                     (NO/YES)

Automation Statistics
Total Events        48,538 Display suppressed      67
Events Handled      22,407 Hardcopy suppressed     0
Current arrival rate 2 / sec Rule generated Alerts 22,223
Peak arrival rate   99 / sec Rule invoked EXECs    276

Automation Library
LC CMDS --- (S)elect, (E)nable, (D)isable, (T)est, (S)ave
(M)ove, (B)efore or (A)fter, (F)ilter Criteria
LC Rule-Set Status Rules Fired Filtered Date Time Strategy
___ AAORUL00 ENABLED 15 22,223 48,573 00/11/16 08:18:59 FIRST
___ AAORULBC ENABLED 54 188 48,573 00/11/16 08:18:59 FIRST
___ AAORULCM ENABLED 52 59 48,573 00/11/16 08:18:59 FIRST
___ RULCICS ENABLED 25 1 48,573 00/11/16 08:19:00 FIRST
e_ RULMQS01 DISABLED 8 0 38 00/11/16 14:14:38 FIRST
***** END OF DATA *****

```

Figure 21. Creating a Rule for a Message with Format MQFMT\_STRING: Automation Control Panel (Example 1)

2. To select the Rule Set, type the S line command, for (S)elect, in the LC field next to the Rule Set name and press Enter. See Figure 22.

```

BMC Software ----- Automation Control ----- RULMQS01 ENABLED
COMMAND ==>                                     TGT ==> SYSE
Primary commands:                                 DATE --- 00/11/16
                                                    TIME --- 08:35:54

Automation Status   ==> ACTIVE                   (Active, Inactive)
Automation Strategy ==> INDIVIDUAL              (Individual, All, First, Qualified)
Honor MPF Suppression ==> NO                     (NO/YES)

Automation Statistics
Total Events        48,538 Display suppressed      67
Events Handled      22,407 Hardcopy suppressed     0
Current arrival rate 2 / sec Rule generated Alerts 22,223
Peak arrival rate   99 / sec Rule invoked EXECs    276

Automation Library
LC CMDS --- (S)elect, (E)nable, (D)isable, (T)est, (S)ave
(M)ove, (B)efore or (A)fter, (F)ilter Criteria
LC Rule-Set Status Rules Fired Filtered Date Time Strategy
___ AAORUL00 ENABLED 15 22,223 48,573 00/11/16 08:18:59 FIRST
___ AAORULBC ENABLED 54 188 48,573 00/11/16 08:18:59 FIRST
___ AAORULCM ENABLED 52 59 48,573 00/11/16 08:18:59 FIRST
___ RULCICS ENABLED 25 1 48,573 00/11/16 08:19:00 FIRST
s_ RULMQS01 ENABLED 8 0 38 00/11/16 14:14:38 FIRST
***** END OF DATA *****

```

Figure 22. Creating a Rule for a Message with Format MQFMT\_STRING: Automation Control Panel (Example 2)

The Rule Set Overview panel is displayed. Figure 23 shows an example of a Rule Set named RULMQS01 and the Rules it contains.

3. To add a new Rule, use the ADD primary command on the command line.

```

BMC Software ----- Rule Set Overview ----- AutoOPERATOR
COMMAND ==> add                                     TGT --- SYSE
Rule Set ID: RULMQS01   Ruleset Strategy ==> ALL      DATE --- 00/11/16
Primary commands: Add, Save, Sort, Unsort, Reset, Filter  TIME --- 09:34:54
LC CMDS --- (S)elect, (E)nable, (D)isable, (T)est, (DE)lete, (I)nsert
              (C)opy/(CC)opy, (M)ove/(MM)ove, (B)efore or (A)fter, (R)peat
Sort Criterion:                                       right/left

LC  Rule-id Stat Text-id      Type  Fired EXEC      Changed      ID
---  ---  ---  ---  ---  ---  ---  ---  ---
___  MQQDPLOW ENA Q_DEPTH_LOW    MQS    0 MQVARS 1 00/11/15 15:56 BAOJDB4
___  MQQDPLO2 ENA Q_DEPTH_LOW    MQS    0 MQVARS 2 00/11/15 15:56 BAOJDB4
___  MQSVINTH ENA Q_SERVICE_INTERV MQS    0 MQVARS 1 00/11/15 15:56 BAOJDB4
___  MQSVINT2 ENA Q_SERVICE_INTERV MQS    0 MQVARS 2 00/11/15 15:56 BAOJDB4
___  MQSVINOK ENA Q_SERVICE_INTERV MQS    0 MQVARS 1 00/11/15 15:56 BAOJDB4
___  MQSVIN02 ENA Q_SERVICE_INTERV MQS    0 MQVARS 2 00/11/15 15:56 BAOJDB4
___  MQPUTENA ENA PUT_INHIBITED    MQS    0 MQVER MQ 00/11/15 14:51 JDB3
___  MQPUTEN2 SUS PUT_INHIBITED    MQS    0 MQVER MQ 00/11/15 14:51 JDB3
___  MQQFULL  ENA Q_FULL          MQS    0 MQVER MQ 00/11/15 20:49 JDB3
___  QUEUE1   ENA                MQS    0 VARDISP 00/11/15 15:56 BAOJDB4
___  APQHI    ENA Q_DEPTH_HIGH    MQS    0          00/11/15 18:32 BAOJDB4
***** END OF DATA *****

```

Figure 23. Creating a Rule for a Message with Format MQFMT\_STRING: Rule Set Overview Panel (Example 1)

The Rule Processor Detail Control panel is displayed.

```

BMC Software ----- Rule Processor Detail Control ----- AutoOPERATOR
COMMAND ==>
                                     TGT --- SYSE
                                     TIME --- 09:35:54
                                     DATE --- 00/11/16

The following options are displayed in sequence, or may
be selected by entering the two-character code
S1 - Selection Criteria           A1 - Action Specification
SV - Variable Dependencies       AA - Alert Actions(s) I
                                 AD - Alert Actions(s) II

Rule ID      ==>
Event Type   ==>           Type of event ( ? for list)
Initial Mode ==>           (ENABLED/DISABLED/TEST)

Criteria match rate:
If matched   ==>           (Maximum # times matched within INTERVAL, 1-100)
in seconds   ==>           (Interval length, 1-99999 seconds)
then status  ==>           (SUSPEND, DISABLE)

Application information:
Group        ==>           Function      ==>           Code      ==>
Author       ==>           Description  ==>
Last Modified by on at

Press ENTER to continue, END to apply changes, CANCEL to cancel changes

```

Figure 24. Creating a Rule for a Message with Format MQFMT\_STRING: Rule Processor Detail Control Panel (Example 1)

4. Type the following information:

- A unique Rule ID in the Rule ID field (for example, CLOSAP); see Figure 25.

- MQS in the Event Type field; see Figure 25.

**Note:** The Event Type MQS will be accepted only if the MAINVIEW AutoOPERATOR for MQSeries key is present. If the key is not present, you will receive a short error message in the upper right corner of the panel.

```

BMC Software ----- Rule Processor Detail Control ----- AutoOPERATOR
COMMAND ==>                                         TGT --- SYSE
                                                    TIME --- 15:46:37
                                                    DATE --- 00/11/16

The following options are displayed in sequence, or may
be selected by entering the two-character code
  S1 - Selection Criteria           A1 - Action Specification
  SV - Variable Dependencies       AA - Alert Actions(s) I
                                    AD - Alert Actions(s) II

Rule ID      ==> CLOSAP
Event Type   ==> MQS      Type of event (? for list)
Initial Mode ==> ENABLED (ENABLED/DISABLED/TEST)

Criteria match rate:
If matched   ==>          (Maximum # times matched within INTERVAL, 1-100)
in seconds   ==>          (Interval length, 1-99999 seconds)
then status  ==>          (SUSPEND, DISABLE)

Application information:
Group        ==>          Function      ==>          Code      ==>
Author       ==>          Description   ==>
Last Modified by      on          at

Press ENTER to continue, END to apply changes, CANCEL to cancel changes

```

Figure 25. Creating a Rule for a Message with Format MQFMT\_STRING: Rule Processor Detail Control Panel (Example 2)

5. Press Enter.

The Selection Criteria - MQS panel is displayed.

```

BMC Software ----- Selection Criteria - MQS ----- AutoOPERATOR
COMMAND ==>                                         TGT --- SYSE

Rule-set === RULMQS01           Rule-id === CLOSAP

Queue Identification: (1 to 12 Queue Managers)
Manager(s) ==>
Queue Id    ==>

Message Identification:
Format      ==>          (Value from MD Format field)
Event Type  ==>          (Enter ? for help)

      Sub  Len  Op  Value
Msgid    ==> ___ : ___ ___ _____
CorrelId ==> ___ : ___ ___ _____
Msg Buffer ==> ___ : ___ ___ _____

Press ENTER to continue, END return to Detail Control, CANCEL to cancel changes

```

Figure 26. Creating a Rule for a Message with Format MQFMT\_STRING: Selection Criteria - MQS Panel (Example 1)

6. To enter the selection criteria for this event:

- In the Manager(s) field, enter the name of your queue manager (for example,CSQ1); see Figure 27.
- In the Queue ID field, type **PROD. AP. QUEUE**. This is the name of the queue where this message must originate; see Figure 27.
- In the Format field, type **MQFMT\_STRING**; see Figure 27.

This is the message format. When you create Rules for MQSeries messages with a format other than MQFMT\_EVENT, use the fields MsgId, CorrelId and Msg Buffer to add additional selection criteria, if required.

- In the Msg Buffer field, type **CLOSE ALL**.

The message must match this text, see Figure 27.

**Note:** Be aware that the variable IMFTEXT, which is passed to an EXEC when invoked from a Rule, has the same limitations as the MSG Buffer of the Selection Criteria Panel, 255 bytes. For more information regarding IMFTEXT, see Chapter 4 in the *MAINVIEW AutoOPERATOR Advanced Automation Guide for REXX EXECs* and the *MAINVIEW AutoOPERATOR Advanced Automation Guide for CLIST EXECs*. To access data in the message beyond 255 bytes, an EXEC must be used. Using an EXEC, you can access up to 32,767 bytes of the message content.

```

BMC Software ----- Selection Criteria - MQS ----- AutoOPERATOR
COMMAND ==>>>                                     TGT --- SYSE

          Rule-set === RULMQS01                Rule-id === CLOSAP

Queue Identification:                               (1 to 12 Queue Managers)
Manager(s)  ==>>> CSQ1
Queue Id    ==>>> PROD. AP. QUEUE

Message Identification:
Format      ==>>> MQFMT_STRING                    (Value from MD Format field)
Event Type  ==>>>                                (Enter ? for help)

          Sub  Len  Op  Value
Msgid      ==>>> ___ : ___ ___ _____
CorrelId   ==>>> ___ : ___ ___ _____
Msg Buffer  ==>>> ___ : ___ ___ CLOSE ALL _____

Press ENTER to continue, END return to Detail Control, CANCEL to cancel changes

```

Figure 27. Selection Criteria - MQS Panel (Example 2)

7. Press Enter.



- Type **NO** in the Keep Message field which deletes the original message; see Figure 29.

```

BMC Software ----- Action Specification - MQS ----- AutoOPERATOR
COMMAND ==>                                     TGT --- SYSE

          Rule-set === RULMQS01                Rule-id === CLOSAP

Automation Actions:
Journal      ==>
EXEC Name/Parms ==>
Cmd (Type MVS ) ==> #S APCLOSE

Set Variable  ==>                               ==>
Notify        ==>                               Outboard Pager ID
Info          ==>

DOM Id        ==>                               Delete Operator Message
Issue WTO Msg ==>

MQSeries Automation Actions:
Keep Message  ==> NO (Yes or NO)  Remove DLH ==> (Yes or No)
Destination Que ==>
Destination QMgr ==>

Press ENTER to continue, END return to Detail Control, CANCEL to cancel changes

```

Figure 29. Action Specification - MQS Panel (Example 2)

10. Press Enter.

The Alert Action(s) I -MQS panel is displayed.

```

BMC Software ----- Alert Action(s) I - MQS ----- AutoOPERATOR
COMMAND ==>                                     TGT --- SYSE

          Rule-set === RULMQS01                Rule-id === CLOSAP
Function ==>                                     (ADD, DELETE, DELETEQ)
Key        ==>
Text       ==>
Queues     ==>                                     Alert Queue Name(s)
Priority    ==>                                     (CRIT, MAJ, MIN, WARN, INFO, CLEAR)
Color      ==>                                     RED, PINK, YEL, DKBL, LTBL, GRE, WHI
PCMD       ==>
System     ==>                                     Return to target after PCMD
EXEC       ==>                                     Follow-up EXEC
Help       ==>                                     Associated HELP Panel
Targets    ==>                                     Target System
Udata      ==>                                     User Data
Origin     ==>                                     Origin
User       ==>                                     Userid
Alarm      ==>                                     Sound Alarm (YES/NO)
Press ENTER to continue, END return to Detail Control, CANCEL to cancel changes

```

11. This example will not issue an ALERT; press Enter.

The Alert Action(s) II - MQS panel is displayed.

```

BMC Software ----- Alert Action(s) II - MQS ----- AutoOPERATOR
COMMAND ==>                                           TGT --- SYSE
      Rule-set == RULMQS01           Rule-id == CLOSAP
Retain          ==>                                           Yes/No
Escalate Direction ==>                                           Up/Down
      Interval ==>
      ==>
      ==>
      ==>
      ==>
      ==>
      Disposition ==>                                           Keep/Delete
      EXEC ==>
Press ENTER to continue, END return to Detail Control, CANCEL to cancel changes

```

12. This example will not issue an ALERT; press Enter.

The Rule Processor Detail Control Panel is redisplayed.

```

BMC Software ----- Rule Processor Detail Control ----- AutoOPERATOR
COMMAND ==>                                           TGT --- SYSE
                                           TIME --- 15: 23: 36
                                           DATE --- 00/11/16

The following options are displayed in sequence, or may
be selected by entering the two-character code

      S1 - Selection Criteria           A1 - Action Specification
      SV - Variable Dependencies       AA - Alert Action(s) I
                                           AD - Alert Action(s) II

Rule ID      ==> CLOSAP
Event Type   ==> MQS      Type of event ( ? for list)
Initial Mode ==> ENABLED (ENABLED/DISABLED/TEST)

Criteria match rate:
If matched   ==>           (Maximum # times matched within INTERVAL, 1-100)
in seconds   ==>           (Interval length, 1-99999 seconds)
then status  ==>           (SUSPEND, DISABLE)

Application information:
Group        ==>           Function      ==>           Code      ==>
Author       ==> BAOJDB4   Description  ==>
Last Modified by on at

```

Figure 30. Creating a Rule for a Message with Format MQFMT\_STRING: Rule Processor Detail Control Panel (Example 3)

13. Press PF3 to apply the changes.

The Rule Set Overview Panel is redisplayed.

```

BMC Software ----- Rule Set Overview ----- RULE UPDATED
COMMAND ==>                                     TGT --- SYSE
Rule Set ID: RULMQS01   Ruleset Strategy ==> ALL   DATE --- 00/11/16
Primary commands: Add, Save, Sort, Unsort, Reset, Filter   TIME --- 16:06:12
LC CMDS --- (S)elect, (E)nable, (D)isable, (T)est, (DE)lete, (I)nsert
              (C)opy/(CC)opy, (M)ove/(MM)ove, (B)efore or (A)fter, (R)peat
Sort Criterion:                                     right/left

LC  Rule-id Stat Text-id           Type  Fired EXEC           Changed           ID
---  ---  ---  ---  ---  ---  ---  ---  ---  ---
___  MQQDPLOW ENA Q_DEPTH_LOW      MQS    0  MQVARS 1  00/11/15 15:56 BAOJDB4
___  MQQDPLO2 ENA Q_DEPTH_LOW      MQS    0  MQVARS 2  00/11/15 15:56 BAOJDB4
___  MQSVINTH ENA Q_SERVICE_INTERV MQS    0  MQVARS 1  00/11/15 15:56 BAOJDB4
___  MQSVINT2 ENA Q_SERVICE_INTERV MQS    0  MQVARS 2  00/11/15 15:56 BAOJDB4
___  MQSVINOK ENA Q_SERVICE_INTERV MQS    0  MQVARS 1  00/11/15 15:56 BAOJDB4
___  MQSVINO2 ENA Q_SERVICE_INTERV MQS    0  MQVARS 2  00/11/15 15:56 BAOJDB4
___  MQPUTENA ENA PUT_INHIBITED     MQS    0  MQVER MQ  97/02/19 14:51 JDB3
___  MQPUTEN2 SUS PUT_INHIBITED     MQS    0  MQVER MQ  97/02/19 14:51 JDB3
___  MQQFULL  ENA Q_FULL            MQS    0  MQVER MQ  97/02/19 20:49 JDB3
___  QUEUE1   ENA                   MQS    0  VARDISP 00/11/15 15:56 BAOJDB4
___  APQHI    ENA Q_DEPTH_HIGH      MQS    0  00/11/15 18:32 BAOJDB4
___  CLOSAP   ENA                   MQS    0  00/11/16 16:06 BAOJDB4
***** END OF DATA *****

```

Figure 31. Rule Set Overview Panel (Example 2)

The Rule is enabled and fires when a MQFMT\_STRING message with the CLOSE ALL message text is detected for the PROD.AP.QUEUE queue within the CSQ1 queue manager. However, at this point, the Rule is not saved to disk.

**Note:** Be aware that the variable IMFTEXT, which is passed to an EXEC when invoked from a Rule, has the same limitations as the MSG Buffer of the Selection Criteria Panel, 255 bytes. For more information regarding IMFTEXT, see Chapter 4 in the *MAINVIEW AutoOPERATOR Advanced Automation Guide for REXX EXECs* and the *MAINVIEW AutoOPERATOR Advanced Automation Guide for CLIST EXECs*. To access data in the message beyond 255 bytes, an EXEC must be used. Using an EXEC, you can access up to 32,767 bytes of the message content.

14. To save the Rule, type the SAVE primary command on the command line.

If you do not type the Save primary command, the following warning is displayed when you press PF3:

```
BMC Software ----- Confirm Rule Set Modifications ----- AutoOPERATOR
COMMAND ==>>> TGT --- SYSE
+-----+
+ WARNING! Changes made to Rule Set RULMQS01 have not been saved. Those +
+ changes were one or more of the following: +
+ + +
+ o A Rule was changed. +
+ o The status of a Rule was modified. +
+ o A Rule was added, deleted, inserted, or copied. +
+ o A Rule was moved. +
+ o The individual Rule Set strategy changed. +
+ + +
+ Please do one of the following: +
+ + +
+ - Enter SAVE to save RULMQS01 to the BBIPARM dataset. +
+ - Enter NOSAVE to exit WITHOUT saving RULMQS01 to the BBIPARM dataset. +
+ - Press END to return to Rule Set Overview. +
+-----+
```

Figure 32. Confirming Rule Set Modifications Panel (Example 2)

The options are:

- Type SAVE to save the Rule
- Type NOSAVE to cancel saving the newly created Rule to disk, but leaving memory copy intact
- Press End to return to the Rule Set Overview panel.

---

## MQSeries Instrumentation Events

This section describes the MQSeries instrumentation events (MQFMT\_EVENT) for which you can create Rules.

In MQSeries, an instrumentation event is a logical combination of conditions that is detected by a queue manager or channel instance. The result of such an event is that the queue manager or channel instance puts a special message, called an event message, on an event queue.

For more information about MQFMT\_EVENT messages, refer to “What MQSeries Events Are” on page 3.

Table 7 on page 74 shows a list of valid instrumentation events that you can automate through the use of Rules. Enter a ? in the Event Type field to get a list of the supported event types. The third column lists the MQSeries-specific AutoOPERATOR variables that are associated with each instrumentation event; the fourth column shows the instrumentation event category; and the fifth column shows the queue manager attribute that enables or disables generation of the instrumentation event.

Table 7. MQSeries Instrumentation Events

Event	Description	Applicable Variables	Event Category	QMGR Attribute
ALIAS_BASE_QTYPE_ERROR	An mqopen or mqput1 call was issued specifying an alias queue as the destination, but the BaseQName in the alias queue definition resolves to a queue that is not a local queue, or local definition of a remote queue.	<p>IMFQ_EVENT_BASEQNAME: Queue manager name (to which the alias resolves)</p> <p>IMFQ_EVENT_QNAME: Name of queue from object descriptor</p> <p>IMFQ_EVENT_QTYPE: Type of queue to which the alias resolves (A for alias, M for model)</p> <p>IMFQ_EVENT_APPLTYPE: Type of application that caused the event. For example:</p> <ul style="list-style-type: none"> <li>MQAT_UNKNOWN</li> <li>MQAT_NO_CONTEXT</li> <li>MQAT_CICS</li> <li>MQAT_OS390</li> <li>MQAT_IMS</li> <li>MQAT_OS2</li> <li>MQAT_DOS</li> <li>MQAT_AIX</li> <li>MQAT_QMGR</li> <li>MQAT_OS400</li> <li>MQAT_WINDOWS</li> <li>MQAT_CICS_VSE</li> <li>MQAT_WINDOWS_NT</li> <li>MQAT_VMS</li> <li>MQAT_GUARDIAN</li> <li>MQAT_VOS</li> <li>MQAT_IMS_BRIDGE</li> <li>MQAT_XCF</li> <li>MQAT_CICS_BRIDGE</li> <li>MQAT_NOTES_AGENT</li> </ul> <ul style="list-style-type: none"> <li>• IMFQ_EVENT_APPLNAME: Name of application</li> <li>• IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote</li> </ul>	Queue Manager	LOCALEV
BRIDGE_STARTED	The IMS bridge has been started.	<ul style="list-style-type: none"> <li>• IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote</li> <li>• IMFQ_EVENT_BRIDGETYPE: Only current value is 1 (OTMA)</li> <li>• IMFQ_EVENT_BRIDGENAME: Name of bridge</li> </ul>	Channel	n/a

Table 7. MQSeries Instrumentation Events (Continued)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
BRIDGE_STOPPED	<p>The IMS bridge has been stopped.</p>	<ul style="list-style-type: none"> <li>• IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote</li> <li>• IMFQ_EVENT_BRIDGETYPE: Only current value is 1 (OTMA)</li> <li>• IMFQ_EVENT_BRIDGENAME: Name of bridge</li> <li>• IMFQ_EVENT_REASONQUALIFIER: Reason code:                             <ul style="list-style-type: none"> <li>– 11 - Bridge stopped, one side or the other issued a normal IXCLEAVE request.</li> <li>– 12 - Bridge stopped, but an error was reported.</li> </ul> </li> <li>• IMFQ_EVENT_ERRORIDENTIFIER: Error identification</li> </ul>	Channel	n/a
CHANNEL_ACTIVATED	<p>A channel which was waiting to become active has now become active. Not generated if the channel did not have to wait for a slot to be released.</p> <p>Not produced for CICS for distributed queue management in MQSeries for OS/390.</p>	<ul style="list-style-type: none"> <li>• IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote</li> <li>• IMFQ_EVENT_CHANNELNAME: Channel name</li> <li>• IMFQ_EVENT_XMITQNAME: Transmission queue name</li> <li>• IMFQ_EVENT_CONNECTIONNAME: Connection name. For TCP/IP, this is the internet address only if the channel has successfully established a connection. Otherwise, it is the contents of the ConnectionName field in the channel definition.</li> </ul>	Channel	n/a

Table 7. MQSeries Instrumentation Events (Continued)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
CHANNEL_CONV_ERROR	This condition is detected when a channel is unable to perform data conversion and the MQGET call to get a message from the transmission queue resulted in a data conversion error. The reason for the failure is identified by IMFQ_EVENT_ConversionReasonCode.	<ul style="list-style-type: none"> <li>• IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote</li> <li>• IMFQ_EVENT_CHANNELNAME: Channel name</li> <li>• IMFQ_EVENT_XMITQNAME: Transmission queue name</li> <li>• IMFQ_EVENT_CONNECTIONNAME: Connection name. For TCP/IP, this is the internet address only if the channel has successfully established a connection. Otherwise, it is the contents of the ConnectionName field in the channel definition.</li> <li>• IMFQ_EVENT_FORMAT: Name of format</li> <li>• IMFQ_EVENT_REASONQUALIFIER: Conversion reason code:            MQRC_FORMAT_ERROR            MQRC_SOURCE_CCSID_ERROR            MQRC_SOURCE_INTEGER_ENC_ERROR            MQRC_SOURCE_DECIMAL_ENC_ERROR            MQRC_SOURCE_FLOAT_ENC_ERROR            MQRC_TARGET_CCSID_ERROR            MQRC_TARGET_INTEGER_ENC_ERROR            MQRC_TARGET_DECIMAL_ENC_ERROR            MQRC_TARGET_FLOAT_ENC_ERROR            MQRC_NOT_CONVERTED            MQRC_CONVERTED_MSG_TOO_BIG            MQRC_TRUNCATED_MSG_ACCEPTED            MQRC_TRUNCATED_MSG_FAILED</li> </ul>	Channel	n/a

Table 7. MQSeries Instrumentation Events (Continued)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
CHANNEL_NOT_ACTIVATED	<p>A channel is waiting to become active because the limit on active channels has been reached. The channel waits until an active slot is released when another channel ceases to be active. A CHANNEL_ACTIVATED event will then be generated.</p> <p><i>Not produced for CICS for distributed queue management in MQSeries for OS/390.</i></p>	<ul style="list-style-type: none"> <li>• IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote</li> <li>• IMFQ_EVENT_CHANNELNAME: Channel name</li> <li>• IMFQ_EVENT_XMITQNAME: Transmission queue name</li> <li>• IMFQ_EVENT_CONNECTIONNAME: Connection name. For TCP/IP, this is the internet address only if the channel has successfully established a connection. Otherwise, it is the contents of the ConnectionName field in the channel definition.</li> </ul>	Channel	n/a
CHANNEL_STARTED	<p>A channel was started because</p> <ul style="list-style-type: none"> <li>• A Start Channel operator command was issued</li> <li>• An instance of a channel connection has been established</li> </ul> <p><i>Not produced for CICS for distributed queue management in MQSeries for OS/390.</i></p>	<ul style="list-style-type: none"> <li>• IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote</li> <li>• IMFQ_EVENT_CHANNELNAME: Channel name</li> <li>• IMFQ_EVENT_XMITQNAME: Transmission queue name</li> <li>• IMFQ_EVENT_CONNECTIONNAME: Connection name. For TCP/IP, this is the internet address only if the channel has successfully established a connection. Otherwise, it is the contents of the ConnectionName field in the channel definition.</li> </ul>	Channel	n/a

Table 7. MQSeries Instrumentation Events (Continued)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
CHANNEL_STOPPED	Channel stopped.	<ul style="list-style-type: none"> <li>• IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote</li> <li>• IMFQ_EVENT_CHANNELNAME: Channel name</li> <li>• IMFQ_EVENT_XMITQNAME: Transmission queue name</li> <li>• IMFQ_EVENT_CONNECTIONNAME: Connection name. For TCP/IP, this is the internet address only if the channel has successfully established a connection. Otherwise, it is the contents of the ConnectionName field in the channel definition.</li> </ul> <p>IMFQ_EVENT_REASONQUALIFIER : Reason qualifier:</p> <p>MQRQ_CHANNEL_STOPPED_OK</p> <p>MQRQ_CHANNEL_STOPPED_ERROR</p> <p>MQRQ_CHANNEL_STOPPED_RETRY</p> <p>MQRQ_CHANNEL_STOPPED_DISABLED</p> <ul style="list-style-type: none"> <li>• IMFQ_EVENT_ERRORIDENTIFIER: See the section regarding the CHANNEL_STOPPED event, field Error identifier, in the IBM publication <i>MQSeries Programmable System Management</i> for a description of this data.</li> </ul>	Channel	n/a
CHANNEL_STOPPED (cont'd)	Channel stopped.	<ul style="list-style-type: none"> <li>• IMFQ_EVENT_AUXERRORDATAINT1: First integer of auxiliary error data for channel errors</li> <li>• IMFQ_EVENT_AUXERRORDATAINT2: Second integer of auxiliary error data for channel errors</li> <li>• IMFQ_EVENT_AUXERRORDATASTR1: First string of auxiliary error data for channel errors</li> <li>• IMFQ_EVENT_AUXERRORDATASTR2: Second string of auxiliary error data for channel errors</li> <li>• IMFQ_EVENT_AUXERRORDATASTR3: Third string of auxiliary error data for channel errors</li> </ul>	Channel	n/a

Table 7. MQSeries Instrumentation Events (Continued)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
DEF_XMIT_Q_TYPE_ERROR	<p>A MQOPEN or MQPUT1 was issued specifying a remote queue as the destination. Either a local definition of the remote queue was specified, or a queue manager alias was being resolved, but in either case the XmitQName attribute in the local definition is blank. No transmission queue is defined with the same name as the destination queue manager, so the local queue manager has attempted to use the default transmission queue. However, although there is a queue defined by the DefXmitQName queue manager attribute, it is not a local queue.</p>	<ul style="list-style-type: none"> <li>• IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote</li> <li>• IMFQ_EVENT_QNAME: Name of queue for that caused the error</li> <li>• IMFQ_EVENT_APPLTYPE: Type of application that caused event. For example:               <ul style="list-style-type: none"> <li>MQAT_UNKNOWN</li> <li>MQAT_NO_CONTEXT</li> <li>MQAT_CICS</li> <li>MQAT_OS390</li> <li>MQAT_IMS</li> <li>MQAT_OS2</li> <li>MQAT_DOS</li> <li>MQAT_AIX</li> <li>MQAT_QMGR</li> <li>MQAT_OS400</li> <li>MQAT_WINDOWS</li> <li>MQAT_CICS_VSE</li> <li>MQAT_WINDOWS_NT</li> <li>MQAT_VMS</li> <li>MQAT_GUARDIAN</li> <li>MQAT_VOS</li> <li>MQAT_IMS_BRIDGE</li> <li>MQAT_XCF</li> <li>MQAT_CICS_BRIDGE</li> <li>MQAT_NOTES_AGENT</li> </ul> </li> <li>IMFQ_EVENT_APPLNAME: Name of application</li> <li>• IMFQ_EVENT_OBJECTQMGRNAME: Name of the object queue manager</li> <li>• IMFQ_EVENT_XMITQNAME: Name of the transmission queue</li> <li>• IMFQ_EVENT_QTYPE: Type of queue (A for alias, R for remote)</li> </ul>	Queue Manager	REMOTEEV

Table 7. MQSeries Instrumentation Events (Continued)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
DEF_XMIT_Q_USAGE_ERROR	<p>A MQOPEN or MQPUT1 was issued specifying a remote queue as the destination. Either a local definition of the remote queue was specified, or a queue manager alias was being resolved, but in either case the XmitQName attribute in the local definition is blank. No transmission queue is defined with the same name as the destination queue manager, so the local queue manager has attempted to use the default transmission queue. However, the queue defined by the XmitQName queue manager attribute does not have a usage attribute of MQUS_TRANSMISSION.</p>	<ul style="list-style-type: none"> <li>• IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote</li> <li>• IMFQ_EVENT_QNAME: Name of queue for which caused the error</li> <li>• IMFQ_EVENT_APPLTYPE: Type of application that caused event. For example: <ul style="list-style-type: none"> <li>MQAT_UNKNOWN</li> <li>MQAT_NO_CONTEXT</li> <li>MQAT_CICS</li> <li>MQAT_OS390</li> <li>MQAT_IMS</li> <li>MQAT_OS2</li> <li>MQAT_DOS</li> <li>MQAT_AIX</li> <li>MQAT_QMGR</li> <li>MQAT_OS400</li> <li>MQAT_WINDOWS</li> <li>MQAT_CICS_VSE</li> <li>MQAT_WINDOWS_NT</li> <li>MQAT_VMS</li> <li>MQAT_GUARDIAN</li> <li>MQAT_VOS</li> <li>MQAT_IMS_BRIDGE</li> <li>MQAT_XCF</li> <li>MQAT_CICS_BRIDGE</li> <li>MQAT_NOTES_AGENT</li> </ul> </li> <li>IMFQ_EVENT_APPLNAME: Name of application</li> <li>• IMFQ_EVENT_OBJECTQMGRNAME: Name of the object queue manager</li> <li>• IMFQ_EVENT_XMITQNAME: Name of the transmission queue</li> </ul>	Queue Manager	REMOTEEV

Table 7. MQSeries Instrumentation Events (Continued)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
GET_INHIBITED	An attempt was made to retrieve a message (using MQGET) but the retrieval was inhibited for the queue.	<ul style="list-style-type: none"> <li>• IMFQ_EVENT_QNAME: Name of queue which GETs are inhibited</li> <li>• IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote</li> <li>• IMFQ_EVENT_APPLNAME: Name of application</li> <li>• IMFQ_EVENT_APPLTYPE: Type of application that caused event. For example:             MQAT_UNKNOWN            MQAT_NO_CONTEXT            MQAT_CICS            MQAT_OS390            MQAT_IMS            MQAT_OS2            MQAT_DOS            MQAT_AIX            MQAT_QMGR            MQAT_OS400            MQAT_WINDOWS            MQAT_CICS_VSE            MQAT_WINDOWS_NT            MQAT_VMS            MQAT_GUARDIAN            MQAT_VOS            MQAT_IMS_BRIDGE            MQAT_XCF            MQAT_CICS_BRIDGE            MQAT_NOTES_AGENT</li> </ul>	Queue Manager	INHIBTEV

Table 7. MQSeries Instrumentation Events (Continued)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
NOT_AUTHORIZED TYPE 1,2,3,4	<p>One of four codes is possible:</p> <ol style="list-style-type: none"> <li>1. On an MQCONN call, the user is not authorized to connect to the queue manager</li> <li>2. On an MQOPEN/MQPUT1: the user is not authorized to open the object for the option(s) specified</li> <li>3. On an MQCLOSE: the user is not authorized to delete the object, which is a permanent dynamic queue, and the HOBJ parameter specified on the MQCLOSE call is not the handle returned by the MQOPEN call which created the queue</li> <li>4. A command has been issued from a user ID that is not authorized to access the object specified in the command</li> </ol>	<ul style="list-style-type: none"> <li>• IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote</li> <li>• IMFQ_EVENT_REASONQUALIFIER: Reason code (type of auth error, 1,2,3 or 4)</li> <li>• IMFQ_EVENT_APPLTYPE: Type of application that caused event. For example: <ul style="list-style-type: none"> <li>MQAT_UNKNOWN</li> <li>MQAT_NO_CONTEXT</li> <li>MQAT_CICS</li> <li>MQAT_OS390</li> <li>MQAT_IMS</li> <li>MQAT_OS2</li> <li>MQAT_DOS</li> <li>MQAT_AIX</li> <li>MQAT_QMGR</li> <li>MQAT_OS400</li> <li>MQAT_WINDOWS</li> <li>MQAT_CICS_VSE</li> <li>MQAT_WINDOWS_NT</li> <li>MQAT_VMS</li> <li>MQAT_GUARDIAN</li> <li>MQAT_VOS</li> <li>MQAT_IMS_BRIDGE</li> <li>MQAT_XCF</li> <li>MQAT_CICS_BRIDGE</li> <li>MQAT_NOTES_AGENT</li> </ul> </li> <li>IMFQ_EVENT_APPLNAME: Name of application</li> <li>• IMFQ_EVENT_OPTIONS: Open options used during the MQOPEN call (type 2 only)</li> <li>• IMFQ_EVENT_QNAME: Name of queue</li> </ul>	Queue Manager	AUTHOREV

Table 7. MQSeries Instrumentation Events (Continued)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
NOT_AUTHORIZED TYPE 1,2,3,4 CONTINUED		<ul style="list-style-type: none"> <li>• IMFQ_EVENT_COMMAND: Identifier for the command (type 4 only). See the discussion for PCF header in the IBM publication <i>MQSeries Programmable System Management</i>.</li> <li>• IMFQ_EVENT_USERIDENTIFIER: User identification that caused the authorization check.</li> <li>• IMFQ_EVENT_OBJECTQMGRNAME: Name of the object queue manager (type 2 only)</li> <li>• IMFQ_EVENT_PROCESSNAME: Name of the process whose attributes have changed (type 2 only)</li> </ul>		

Table 7. MQSeries Instrumentation Events (Continued)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
PUT_INHIBITED	An attempt was made to send a message to a queue (using MQPUT) but the attempt was inhibited for the queue.	<ul style="list-style-type: none"> <li>• IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote</li> <li>• IMFQ_EVENT_APPLTYPE: Type of application that caused event. For example:                             <ul style="list-style-type: none"> <li>MQAT_UNKNOWN</li> <li>MQAT_NO_CONTEXT</li> <li>MQAT_CICS</li> <li>MQAT_OS390</li> <li>MQAT_IMS</li> <li>MQAT_OS2</li> <li>MQAT_DOS</li> <li>MQAT_AIX</li> <li>MQAT_QMGR</li> <li>MQAT_OS400</li> <li>MQAT_WINDOWS</li> <li>MQAT_CICS_VSE</li> <li>MQAT_WINDOWS_NT</li> <li>MQAT_VMS</li> <li>MQAT_GUARDIAN</li> <li>MQAT_VOS</li> <li>MQAT_IMS_BRIDGE</li> <li>MQAT_XCF</li> <li>MQAT_CICS_BRIDGE</li> <li>MQAT_NOTES_AGENT</li> </ul> </li> <li>• IMFQ_EVENT_APPLNAME: Name of application</li> <li>• IMFQ_EVENT_QNAME: Name of queue for which PUTs are inhibited</li> <li>• IMFQ_EVENT_OBJECTQMGRNAME: Name of the object queue manager if the object queue manager (when the queue was opened) is not the queue manager currently connected</li> </ul>	Queue Manager	INHIBTEV

Table 7. MQSeries Instrumentation Events (Continued)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
Q_DEPTH_HIGH	An MQPUT or MQPUT1 call caused the queue depth to increase to the queue depth high limit.	<ul style="list-style-type: none"> <li>• IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote</li> <li>• IMFQ_EVENT_BASEQNAME: Name of queue for which caused the error</li> <li>• IMFQ_EVENT_TIMESINCERESSET: Time (in seconds) since the statistics were last reset</li> <li>• IMFQ_EVENT_HIGHQDEPTH: Max number of message on the queue since the statistics were last reset</li> <li>• IMFQ_EVENT_MSGENQCOUNT: Number of messages enqueued since statistics were last reset</li> <li>• IMFQ_EVENT_MSGDEQCOUNT: Number of messages dequeued since statistics were last reset</li> </ul>	Performance	PERFMDEV
Q_DEPTH_LOW	An MQGET call caused the queue depth to decrease to the queue depth low limit.	<ul style="list-style-type: none"> <li>• IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote</li> <li>• IMFQ_EVENT_BASEQNAME: Name of queue for which caused the error</li> <li>• IMFQ_EVENT_TIMESINCERESSET: Time (in seconds) since the statistics were last reset</li> <li>• IMFQ_EVENT_HIGHQDEPTH: Max number of message on the queue since the statistics were last reset</li> <li>• IMFQ_EVENT_MSGENQCOUNT: Number of messages enqueued since statistics were last reset</li> <li>• IMFQ_EVENT_MSGDEQCOUNT: Number of messages dequeued since statistics were last reset</li> </ul>	Performance	PERFMDEV

Table 7. MQSeries Instrumentation Events (Continued)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
Q_FULL	An MQPUT or MQPUT1 call failed because the queue is full.	<ul style="list-style-type: none"> <li>• IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote</li> <li>• IMFQ_EVENT_BASEQNAME: Name of queue for which caused the error</li> <li>• IMFQ_EVENT_TIMESINCERESET: Time (in seconds) since the statistics were last reset</li> <li>• IMFQ_EVENT_HIGHQDEPTH: Max number of message on the queue since the statistics were last reset</li> <li>• IMFQ_EVENT_MSGENQCOUNT: Number of messages enqueued since statistics were last reset</li> <li>• IMFQ_EVENT_MSGDEQCOUNT: Number of messages dequeued since statistics were last reset</li> </ul>	Performance	PERFMEV
Q_MGR_ACTIVE	Detected when a queue manager becomes active. In MQSeries for OS/390, this event is not generated for the first start of a queue manager, only on subsequent restarts.	<ul style="list-style-type: none"> <li>• IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote</li> </ul>	Queue Manager	STRSTPEV
Q_MGR_NOT_ACTIVE	Detected when a queue manager is requested to stop or quiesce.	<ul style="list-style-type: none"> <li>• IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote</li> <li>• IMFQ_EVENT_REASONQUALIFIER: Reason code (5 for queue manager stopping, 6 for queue manager quiescing)</li> </ul>	Queue Manager	STRSTPEV

Table 7. MQSeries Instrumentation Events (Continued)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
Q_SERVICE_INTERVAL_HIGH	No successful gets or puts have been detected within an interval that is greater than the threshold specified in the queue service interval attribute.	<ul style="list-style-type: none"> <li>• IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote</li> <li>• IMFQ_EVENT_BASEQNAME: Name of queue for which caused the error</li> <li>• IMFQ_EVENT_TIMESINCERESET: Time (in seconds) since the statistics were last reset; for this event, this is greater than the service interval</li> <li>• IMFQ_EVENT_HIGHQDEPTH: Max number of message on the queue since the statistics were last reset</li> <li>• IMFQ_EVENT_MSGENQCOUNT: Number of messages enqueued since statistics were last reset</li> <li>• IMFQ_EVENT_MSGDEQCOUNT: Number of messages dequeued since statistics were last reset</li> </ul>	Performance	PERFMEV

Table 7. MQSeries Instrumentation Events (Continued)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
Q_TYPE_ERROR	<p>Could result for one of the following reasons:</p> <ul style="list-style-type: none"> <li>• On an MQOPEN call, the 'object queue manager name' field specifies the name of a local definition of a remote queue, and in the definition the 'remote queue manager name' the attribute is the name of the local queue manager</li> <li>• An MQPUT1 call specifies the name of a model queue</li> <li>• On a previous MQPUT/MQPUT1 call, the 'reply to queue' specified a model queue</li> </ul>	<ul style="list-style-type: none"> <li>• IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote</li> <li>• IMFQ_EVENT_QNAME: Name of queue for which caused the error</li> <li>• IMFQ_EVENT_APPLTYPE: Type of application that caused event. For example: <ul style="list-style-type: none"> <li>MQAT_UNKNOWN</li> <li>MQAT_NO_CONTEXT</li> <li>MQAT_CICS</li> <li>MQAT_OS390</li> <li>MQAT_IMS</li> <li>MQAT_OS2</li> <li>MQAT_DOS</li> <li>MQAT_AIX</li> <li>MQAT_QMGR</li> <li>MQAT_OS400</li> <li>MQAT_WINDOWS</li> <li>MQAT_CICS_VSE</li> <li>MQAT_WINDOWS_NT</li> <li>MQAT_VMS</li> <li>MQAT_GUARDIAN</li> <li>MQAT_VOS</li> <li>MQAT_IMS_BRIDGE</li> <li>MQAT_XCF</li> <li>MQAT_CICS_BRIDGE</li> <li>MQAT_NOTES_AGENT</li> </ul> </li> <li>• IMFQ_EVENT_APPLNAME: Name of application</li> <li>• IMFQ_EVENT_OBJECTQMGRNAME: Name of the object queue manager</li> </ul>	Queue Manager	REMOTEEV

Table 7. MQSeries Instrumentation Events (Continued)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
REMOTE_Q_NAME_ERROR	<p>During an MQOPEN or MQPUT1 call, one of the following occurred:</p> <ul style="list-style-type: none"> <li>A local definition of a remote queue (or an alias to one) was specified, but the 'remote queue name' attribute in the remote queue definition is blank</li> <li>An MQPUT1 call specifies the name of a model queue</li> <li>The 'object queue manager name' field was not blank, and not the name of the local queue manager, but the 'object name' is blank</li> </ul>	<ul style="list-style-type: none"> <li>IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote</li> <li>IMFQ_EVENT_QNAME: Name of queue for which caused the error</li> <li>IMFQ_EVENT_APPLTYPE: Type of application that caused event. For example: <ul style="list-style-type: none"> <li>MQAT_UNKNOWN</li> <li>MQAT_NO_CONTEXT</li> <li>MQAT_CICS</li> <li>MQAT_OS390</li> <li>MQAT_IMS</li> <li>MQAT_OS2</li> <li>MQAT_DOS</li> <li>MQAT_AIX</li> <li>MQAT_QMGR</li> <li>MQAT_OS400</li> <li>MQAT_WINDOWS</li> <li>MQAT_CICS_VSE</li> <li>MQAT_WINDOWS_NT</li> <li>MQAT_VMS</li> <li>MQAT_GUARDIAN</li> <li>MQAT_VOS</li> <li>MQAT_IMS_BRIDGE</li> <li>MQAT_XCF</li> <li>MQAT_CICS_BRIDGE</li> <li>MQAT_NOTES_AGENT</li> </ul> </li> <li>IMFQ_EVENT_APPLNAME: Name of application</li> <li>IMFQ_EVENT_OBJECTQMGRNAME: Name of the object queue manager</li> </ul>	Queue Manager	REMOTEEV

Table 7. MQSeries Instrumentation Events (Continued)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
UNKNOWN_ ALIAS_BASE_Q	An MQOPEN or MQPUT1 was issued specifying an alias queue as the target, but the 'base queue name' in the alias queue attributes are not recognized.	<ul style="list-style-type: none"> <li>• IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote</li> <li>• IMFQ_EVENT_QNAME: Name of queue for which caused the error</li> <li>• IMFQ_EVENT_APPLTYPE: Type of application that caused event. For example:                       MQAT_UNKNOWN                      MQAT_NO_CONTEXT                      MQAT_CICS                      MQAT_OS390                      MQAT_IMS                      MQAT_OS2                      MQAT_DOS                      MQAT_AIX                      MQAT_QMGR                      MQAT_OS400                      MQAT_WINDOWS                      MQAT_CICS_VSE                      MQAT_WINDOWS_NT                      MQAT_VMS                      MQAT_GUARDIAN                      MQAT_VOS                      MQAT_IMS_BRIDGE                      MQAT_XCF                      MQAT_CICS_BRIDGE                      MQAT_NOTES_AGENT                 </li> <li>• IMFQ_EVENT_APPLNAME: Name of application</li> <li>• IMFQ_EVENT_OBJECTQMGRNAME: Name of the object queue manager</li> <li>• IMFQ_EVENT_BASEQNAME: Queue name to which the alias resolves</li> </ul>	Queue Manager	LOCALEV

Table 7. MQSeries Instrumentation Events (Continued)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
UNKNOWN_DEF_XMIT_Q	<p>An MQOPEN or MQPUT1 was issued specifying a remote queue. If a local definition of the remote queue was specified, or if a queue manager alias is being resolved, the XmitQName attribute in the local definition is blank. No queue is defined with the same name as the destination queue manager. The queue manager has therefore attempted to use the default transmission queue. However, the name defined by the DefXmitQName queue manager attribute is not the name of a locally defined queue.</p>	<ul style="list-style-type: none"> <li>• IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote</li> <li>• IMFQ_EVENT_QNAME: Name of queue for which caused the error</li> <li>• IMFQ_EVENT_APPLTYPE: Type of application that caused event. For example: <ul style="list-style-type: none"> <li>MQAT_UNKNOWN</li> <li>MQAT_NO_CONTEXT</li> <li>MQAT_CICS</li> <li>MQAT_OS390</li> <li>MQAT_IMS</li> <li>MQAT_OS2</li> <li>MQAT_DOS</li> <li>MQAT_AIX</li> <li>MQAT_QMGR</li> <li>MQAT_OS400</li> <li>MQAT_WINDOWS</li> <li>MQAT_CICS_VSE</li> <li>MQAT_WINDOWS_NT</li> <li>MQAT_VMS</li> <li>MQAT_GUARDIAN</li> <li>MQAT_VOS</li> <li>MQAT_IMS_BRIDGE</li> <li>MQAT_XCF</li> <li>MQAT_CICS_BRIDGE</li> <li>MQAT_NOTES_AGENT</li> </ul> </li> <li>• IMFQ_EVENT_APPLNAME: Name of application</li> <li>• IMFQ_EVENT_OBJECTQMGRNAME: Name of the object queue manager</li> <li>• IMFQ_EVENT_XMITQNAME: Name of the transmission queue</li> </ul>	Queue Manager	REMOTEEV

Table 7. MQSeries Instrumentation Events (Continued)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
UNKNOWN_ OBJECT_NAME	<p>An MQOPEN or MQPUT1 was issued and the 'object queue manager name' is set to</p> <ul style="list-style-type: none"> <li>• Blank</li> <li>• Name of the local queue manager</li> </ul> <p>The name of a local definition of a remote queue in which the 'remote queue manager name' attribute is the name of the local queue manager.</p>	<ul style="list-style-type: none"> <li>• IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote</li> <li>• IMFQ_EVENT_QNAME: Name of queue for which caused the error</li> <li>• IMFQ_EVENT_APPLTYPE: Type of application that caused event. For example: <ul style="list-style-type: none"> <li>MQAT_UNKNOWN</li> <li>MQAT_NO_CONTEXT</li> <li>MQAT_CICS</li> <li>MQAT_OS390</li> <li>MQAT_IMS</li> <li>MQAT_OS2</li> <li>MQAT_DOS</li> <li>MQAT_AIX</li> <li>MQAT_QMGR</li> <li>MQAT_OS400</li> <li>MQAT_WINDOWS</li> <li>MQAT_CICS_VSE</li> <li>MQAT_WINDOWS_NT</li> <li>MQAT_VMS</li> <li>MQAT_GUARDIAN</li> <li>MQAT_VOS</li> <li>MQAT_IMS_BRIDGE</li> <li>MQAT_XCF</li> <li>MQAT_CICS_BRIDGE</li> <li>MQAT_NOTES_AGENT</li> </ul> </li> <li>• IMFQ_EVENT_APPLNAME: Name of application</li> <li>• IMFQ_EVENT_OBJECTQMGRNAME: Name of the object queue manager</li> <li>• IMFQ_EVENT_PROCESSNAME: Name of the process (application) issuing the MQI call that caused the event</li> </ul>	Queue Manager	LOCALEV

Table 7. MQSeries Instrumentation Events (Continued)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
UNKNOWN_REMOTE_Q_MGR	<p>An MQOPEN or MQPUT1 was issued, and an error occurred with the queue name resolution. For more information, refer to the IBM publication <i>MQSeries Programmable System Management</i>.</p>	<ul style="list-style-type: none"> <li>• IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote</li> <li>• IMFQ_EVENT_QNAME: Name of queue for which caused the error</li> <li>• IMFQ_EVENT_APPLTYPE: Type of application that caused event. For example: <ul style="list-style-type: none"> <li>MQAT_UNKNOWN</li> <li>MQAT_NO_CONTEXT</li> <li>MQAT_CICS</li> <li>MQAT_OS390</li> <li>MQAT_IMS</li> <li>MQAT_OS2</li> <li>MQAT_DOS</li> <li>MQAT_AIX</li> <li>MQAT_QMGR</li> <li>MQAT_OS400</li> <li>MQAT_WINDOWS</li> <li>MQAT_CICS_VSE</li> <li>MQAT_WINDOWS_NT</li> <li>MQAT_VMS</li> <li>MQAT_GUARDIAN</li> <li>MQAT_VOS</li> <li>MQAT_IMS_BRIDGE</li> <li>MQAT_XCF</li> <li>MQAT_CICS_BRIDGE</li> <li>MQAT_NOTES_AGENT</li> </ul> </li> <li>• IMFQ_EVENT_APPLNAME: Name of application</li> <li>• IMFQ_EVENT_OBJECTQMGRNAME: Name of the object queue manager</li> </ul>	Queue Manager	REMOTEEV

Table 7. MQSeries Instrumentation Events (Continued)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
UNKNOWN_XMIT_Q	<p>On an MQOPEN or MQPUT1, a message is to be sent to a remote queue manager. The ObjectName or the ObjectQMgrName in the object descriptor specifies the name of a local definition of a remote queue (in the latter case queue manager aliasing is being used), but the XmitQName attribute of the definition is not blank and not the name of a locally defined queue.</p>	<ul style="list-style-type: none"> <li>• IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote</li> <li>• IMFQ_EVENT_QNAME: Name of queue for which caused the error</li> <li>• IMFQ_EVENT_APPLTYPE: Type of application that caused event. For example: <ul style="list-style-type: none"> <li>MQAT_UNKNOWN</li> <li>MQAT_NO_CONTEXT</li> <li>MQAT_CICS</li> <li>MQAT_OS390</li> <li>MQAT_IMS</li> <li>MQAT_OS2</li> <li>MQAT_DOS</li> <li>MQAT_AIX</li> <li>MQAT_QMGR</li> <li>MQAT_OS400</li> <li>MQAT_WINDOWS</li> <li>MQAT_CICS_VSE</li> <li>MQAT_WINDOWS_NT</li> <li>MQAT_VMS</li> <li>MQAT_GUARDIAN</li> <li>MQAT_VOS</li> <li>MQAT_IMS_BRIDGE</li> <li>MQAT_XCF</li> <li>MQAT_CICS_BRIDGE</li> <li>MQAT_NOTES_AGENT</li> </ul> </li> <li>• IMFQ_EVENT_APPLNAME: Name of application</li> <li>• IMFQ_EVENT_OBJECTQMGRNAME: Name of the object queue manager</li> <li>• IMFQ_EVENT_XMITQNAME: Name of the transmission queue</li> </ul>	Queue Manager	REMOTEEV

Table 7. MQSeries Instrumentation Events (Continued)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
XMIT_Q_TYPE_ERROR	<p>Occurs during an MQOPEN or MQPUT1 call when a message is directed to be sent to a remote queue manager. The 'object name' or 'object queue manager name' field specifies the local definition of a remote queue, but one of the following applies to the 'xmit name' attribute of the definition:</p> <ul style="list-style-type: none"> <li>• 'Xmit queue name' specifies a queue that is not a local queue</li> <li>• 'Xmit queue name' is blank, but 'remote queue manager name' specifies a queue that is not a local queue</li> </ul>	<ul style="list-style-type: none"> <li>• IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote</li> <li>• IMFQ_EVENT_QNAME: Name of queue for which caused the error</li> <li>• IMFQ_EVENT_APPLTYPE: Type of application that caused event. For example: <ul style="list-style-type: none"> <li>MQAT_UNKNOWN</li> <li>MQAT_NO_CONTEXT</li> <li>MQAT_CICS</li> <li>MQAT_OS390</li> <li>MQAT_IMS</li> <li>MQAT_OS2</li> <li>MQAT_DOS</li> <li>MQAT_AIX</li> <li>MQAT_QMGR</li> <li>MQAT_OS400</li> <li>MQAT_WINDOWS</li> <li>MQAT_CICS_VSE</li> <li>MQAT_WINDOWS_NT</li> <li>MQAT_VMS</li> <li>MQAT_GUARDIAN</li> <li>MQAT_VOS</li> <li>MQAT_IMS_BRIDGE</li> <li>MQAT_XCF</li> <li>MQAT_CICS_BRIDGE</li> <li>MQAT_NOTES_AGENT</li> </ul> </li> <li>• IMFQ_EVENT_APPLNAME: Name of application</li> <li>• IMFQ_EVENT_XMITQNAME: Transmission queue name, not available for some events</li> <li>• IMFQ_EVENT_OBJECTQMGRNAME: Name of the object queue manager</li> <li>• IMFQ_EVENT_QTYPE: Type of queue (A for alias, R for remote)</li> </ul>	Queue Manager	REMOTEEV

Table 7. MQSeries Instrumentation Events (Continued)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
XMIT_Q_USAGE_ERROR	<p>During an MQOPEN or MQPUT1 call a message was directed to a remote queue manager but one of the following occurred:</p> <ul style="list-style-type: none"> <li>• ‘Object queue manager name’ specifies the name of a local queue, but it does not have a ‘usage’ attribute of ‘transmission’</li> <li>• The ‘object name’ or ‘object queue manager name’ specifies the name of a local definition of a remote queue but one of the following applies to the ‘xmit queue name’ attribute definition: <ul style="list-style-type: none"> <li>– ‘Xmit queue name’ is not blank, but specifies a queue that does not have a usage attribute of ‘transmission’</li> <li>– ‘Xmit queue name’ is blank, but ‘remote queue manager name’ specifies a queue that does not have a usage attribute of ‘transmission’</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote</li> <li>• IMFQ_EVENT_QNAME: Name of queue for which caused the error</li> <li>• IMFQ_EVENT_APPLTYPE: Type of application that caused event. For example: <ul style="list-style-type: none"> <li>MQAT_UNKNOWN</li> <li>MQAT_NO_CONTEXT</li> <li>MQAT_CICS</li> <li>MQAT_OS390</li> <li>MQAT_IMS</li> <li>MQAT_OS2</li> <li>MQAT_DOS</li> <li>MQAT_AIX</li> <li>MQAT_QMGR</li> <li>MQAT_OS400</li> <li>MQAT_WINDOWS</li> <li>MQAT_CICS_VSE</li> <li>MQAT_WINDOWS_NT</li> <li>MQAT_VMS</li> <li>MQAT_GUARDIAN</li> <li>MQAT_VOS</li> <li>MQAT_IMS_BRIDGE</li> <li>MQAT_XCF</li> <li>MQAT_CICS_BRIDGE</li> <li>MQAT_NOTES_AGENT</li> </ul> </li> <li>• IMFQ_EVENT_APPLNAME: Name of application</li> <li>• IMFQ_EVENT_OBJECTQMGRNAME: Name of the object queue manager</li> <li>• IMFQ_EVENT_XMITQNAME: Transmission queue name, not available for some events</li> </ul>	Queue Manager	REMOTEEV

---

## What the Variables for MQSeries Events Are

This section describes SHARED variables and AutoOPERATOR-supplied variables that are set by MQSeries events.

### SHARED Variables

Table 8 contains descriptions for SHARED variables available to AutoOPERATOR Rules and EXECs.

Table 8. SHARED Variables for MAINVIEW AutoOPERATOR for MQSeries

<b>SHARED variable name</b>	<b>Description</b>
QAOQAOST	Status of MAINVIEW AutoOPERATOR for MQSeries. Possible values are ACT or INACT.
QQMSxxxx or QQMS.XXXX	Status of connection to queue manager where xxxx is the subsystem ID (SSID) of the queue manager. This variable is valid only for queue managers with a valid connection. The absence of this variable may mean the queue manager is not active, or no Rules are enabled which access this queue manager.
QCPFxxxx or QCPF.XXXX	Command prefix string (CPF) where xxxx is the SSID of the queue manager. This variable is available only if a connection has been established.
QDEDxxxx or QDED.XXXX	Name of the dead letter queue where xxxx is the SSID of the queue manager. This variable is available only if a connection has been established.
QCMDxxxx or QCMD.XXXX	Name of the command input queue where xxxx is the SSID of the queue manager. This variable is available only if a connection has been established.

## AutoOPERATOR-Supplied Variables for all MQSeries Messages

Table 9 contains descriptions of AutoOPERATOR-supplied variables available to Rules and EXECs for all MQSeries messages.

The following table describes the most commonly used variables that are available to a Rule or when an EXEC is invoked from an MQSeries Rule or a combination of both. For other structures not listed here, for example DLH (Dead Letter Header) and others, please refer to Appendix A, “Diagnosing AutoOPERATOR for MQSeries Errors” on page 219. Variables created from structures begin with “IMFQ\_”, followed by the MQSeries structure, followed by the field name within the MQSeries structure, with each node delimited by an underscore. For definitions of field names and constants used in this table, which is derived from MQSeries structure field names, see the *MQSeries Application Programming Reference Guide*.

Table 9. AutoOPERATOR-Supplied Variables for all MQSeries Messages

Variable name	Description
<b>Note:</b>	These variables follow the IBM conventions where possible. Therefore, the variables lists in this table may not be complete because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i> .
IMFOJOB	OS/390 jobname of queue manager.
IMFOQMGR	Subsystem ID of queue manager.
IMFQCPF	OS/360 console command prefix for this MQ queue manager.
IMFQRMT	Origin of message (Y: message is from a remote queue, N: message is not from a remote queue).
IMFQ_STRUCTURES	Contains a blank-delimited list of the MQSeries structures contained in the message. For example, in the case of a Dead Letter message, the contents would contain at least: <i>MD DLH</i>
IMFQ_OFFSETS	Contains blank-delimited list of offsets to structures and application data from the front of the message buffer. For example: if the message buffer contains a Dead Letter Header (DLH) and application data, the variable value may look like this: 'NONE 0 172' indicating NONE for MD (since it is not in the message buffer), 0 for the DLH (first structure in the buffer) and 172 for application data (follows the DLH).
IMFQ_QAO_CATCH_UP_MSG	Indicates whether the current message is a potential Catch-up message (present on the queue when AutoOPERATOR opened it).  Possible values are <ul style="list-style-type: none"> <li>• Y (Yes): the message already existed on the queue</li> <li>• N (No): the message has newly arrived since the queue was opened</li> </ul>
IMFQ_MD_STRUCID	Contains the value MQMD_STRUC_ID.

Table 9. AutoOPERATOR-Supplied Variables for all MQSeries Messages (Continued)

Variable name	Description
IMFQ_MD_VERSION	Contains either MQMD_VERSION_1 or MQMD_VERSION_2
IMFQ_MD_REPORT	<p>Contains one or more of the following delimited by blanks:</p> <p>MQRO_EXCEPTION  MQRO_EXCEPTION_WITH_DATA  MQRO_EXCEPTION_WITH_FULL_DATA  MQRO_EXPIRATION  MQRO_EXPIRATION_WITH_DATA  MQRO_EXPIRATION_WITH_FULL_DATA  MQRO_COA  MQRO_COA_WITH_DATA  MQRO_COA_WITH_FULL_DATA  MQRO_COD  MQRO_COD_WITH_DATA  MQRO_COD_WITH_FULL_DATA  MQRO_PAN  MQRO_NAN  MQRO_NEW_MSG_ID  MQRO_PASS_MSG_ID  MQRO_COPY_MSG_ID_TO_CORREL_ID  MQRO_PASS_CORREL_ID  MQRO_DEAD_LETTER_Q  MQRO_DISCARD_MSG  MQRO_NONE</p> <p><i>See note at the beginning of this table.</i></p>
IMFQ_MD_MSGTYPE	<p>Contains one of the following values:</p> <p>MQMT_DATAGRAM  MQMT_REQUEST  MQMT_REPLY  MQMT_REPORT</p> <p><i>See note at the beginning of this table.</i></p>
IMFQ_MD_EXPIRY	Contains a decimal value in the range 1 to 999999999, or MQEI_UNLIMITED.

Table 9. AutoOPERATOR-Supplied Variables for all MQSeries Messages (Continued)

Variable name	Description
IMFQ_MD_FEEDBACK	<p>Contains one of the following values for report messages:</p> <p>MQFB_NONE  MQFB_COA  MQFB_COD  MQFB_EXPIRATION  MQFB_PAN  MQFB_NAN  MQFB_QUIT</p> <p>In addition, this field can contain any reason code from the following sources:</p> <ul style="list-style-type: none"> <li>• IMS-bridge feedback codes</li> <li>• CICS-bridge feedback codes</li> <li>• MQSeries reason codes</li> </ul> <p><b>Note:</b> The content of this field can be a feedback code or a reason code. You can determine whether it is a feedback code or reason code by its value. For example, feedback codes start with MQFB_xxxx and reason codes start with MQRC_xxxx.</p> <p><i>See note at the beginning of this table.</i></p>
IMFQ_MD_ENCODING	<p>Contains either MQENC_NATIVE or any decimal value up to 999999999.</p>
IMFQ_MD_CODEDCHARSETID	<p>Contains</p> <p>MQCCSI_Q_MGR  MQCCSI_EMBEDDED  or any decimal value up to 999999999.</p> <p><i>See IBM MQSeries Application Programming Reference Guide for details on how the CodedCharSetId field is used as input.</i></p>

Table 9. AutoOPERATOR-Supplied Variables for all MQSeries Messages (Continued)

Variable name	Description
IMFQ_MD_FORMAT	<p>Contains one of the following values:</p> <p>MQFMT_NONE  MQFMT_ADMIN  MQFMT_CHANNEL_COMPLETED  MQFMT_CICS  MQFMT_COMMAND_1  MQFMT_COMMAND_2  MQFMT_DEAD_LETTER_HEADER  MQFMT_EVENT  MQFMT_IMS  MQFMT_IMS_VAR_STRING  MQFMT_MD_EXTENSION  MQFMT_PCF  MQFMT_REF_MSG_HEADER  MQFMT_STRING  MQFMT_TRIGGER  MQFMT_WORK_INFO_HEADER  MQFMT_XMIT_Q_HEADER  MQFMT_RF_HEADER  MQFMT_RF_HEADER_2</p> <p><i>See note at the beginning of this table.</i></p>
IMFQ_MD_PRIORITY	Contains a decimal value in the range 0 to 999999999.
IMFQ_MD_PERSISTENCE	<p>Contains one of the following values:</p> <ul style="list-style-type: none"> <li>• MQPER_PERSISTENT</li> <li>• MQPER_NOT_PERSISTENT</li> </ul>
IMFQ_MD_MSGID	Contains up to a 32-byte Msgid. Processed exactly as received. No conversion of hexadecimal data is done.
IMFQ_MD_CORRELID	Contains up to a 32-byte CorrelId. Processed exactly as received. No conversion of hexadecimal data is done.
IMFQ_MD_BACKOUTCOUNT	Contains a decimal value in the range 0 to 255.
IMFQ_MD_REPLYTOQ	Contains up to a 48-character name of the ReplyToQ.
IMFQ_MD_REPLYTOQMGR	Contains up to a 48-character name of the ReplyToQMgr.
IMFQ_MD_USERIDENTIFIER	Contains up to a 12-character UserIdentifier.
IMFQ_MD_ACCOUNTINGTOKEN	Contains either MQACT_NONE or up to a 32-byte AccountingToken. Processed exactly as received. No conversion of hexadecimal data is done.
IMFQ_MD_APPLIDENTITYDATA	Contains up to a 32-character value for ApplIdentityData.

Table 9. AutoOPERATOR-Supplied Variables for all MQSeries Messages (Continued)

Variable name	Description
IMFQ_MD_PUTAPPLTYPE	<p>Contains a user-defined type or one or more of the following standard types:</p> <ul style="list-style-type: none"> <li>MQAT_AIX</li> <li>MQAT_BROKER</li> <li>MQAT_CICS</li> <li>MQAT_CICS_BRIDGE</li> <li>MQAT_CICS_VSE</li> <li>MQAT_DEFAULT</li> <li>MQAT_DOS</li> <li>MQAT_DQM</li> <li>MQAT_GUARDIAN</li> <li>MQAT_IMS</li> <li>MQAT_IMS_BRIDGE</li> <li>MQAT_JAVA</li> <li>MQAT_NO_CONTEXT</li> <li>MQAT_NOTES_AGENT</li> <li>MQAT_NSK</li> <li>MQAT_OS2</li> <li>MQAT_OS390</li> <li>MQAT_OS400</li> <li>MQAT_QMGR</li> <li>MQAT_UNIX</li> <li>MQAT_VMS</li> <li>MQAT_VOS</li> <li>MQAT_WINDOWS</li> <li>MQAT_WINDOWS_NT</li> <li>MQAT_XCF</li> <li>MQAT_UNKNOWN</li> </ul> <p><i>See note at the beginning of this table.</i></p>
IMFQ_MD_PUTAPPLNAME	Contains up to a 28-character value for PutApplName.
IMFQ_MD_PUTDATE	Contains an 8-character date stamp.
IMFQ_MD_PUTTIME	Contains an 8-character time stamp.
IMFQ_MD_APPLORIGINDATA	Contains a 4-character value for ApplOriginData.
IMFQ_MD_GROUPID	If an MQMD_VERSION_2 MD is present, this variable may contain a 24-byte GroupId. Processed exactly as received. No conversion of hexadecimal data is done.
IMFQ_MD_MSGSEQNUMBER	If an MQMD_VERSION_2 MD is present, this variable may contain a decimal value in the range 1 to 999999999.
IMFQ_MD_OFFSET	If an MQMD_VERSION_2 MD is present, this variable may contain a decimal value in the range 0 to 999999999.

Table 9. AutoOPERATOR-Supplied Variables for all MQSeries Messages (Continued)

Variable name	Description
IMFQ_MD_MSGFLAGS	<p>If an MQMD_VERSION_2 MD is present, this variable may contain one or more of the following character strings delimited by blanks:</p> <p>MQMF_SEGMENTATION_INHIBITED  MQMF_SEGMENTATION_ALLOWED  MQMF_MSG_IN_GROUP  MQMF_LAST_MSG_IN_GROUP  MQMF_SEGMENT  MQMF_LAST_SEGMENT  MQMF_NONE</p> <p><i>See note at the beginning of this table.</i></p>
IMFQ_MD_ORIGINALLENGTH	<p>Contains a decimal value in the range 1 to 999999999, or MQOL_UNDEFINED.</p> <p><i>See note at the beginning of this table.</i></p>
IMFQ_QATTR_BACKOUTREQUEUEQNAME	<p>Contains up to a 48-character value.</p>
IMFQ_QATTR_BACKOUTTHRESHOLD	<p>Contains a decimal number in the range 0 to 999999999.</p>
IMFQ_QATTR_CREATIONDATE	<p>Contains a 12-character date.</p>
IMFQ_QATTR_CREATIONTIME	<p>Contains an 8-character time.</p>
IMFQ_QATTR_QDESC	<p>Contains up to a 48-character value.</p>
IMFQ_QATTR_CURRENTQDEPTH	<p>Contains a decimal number for the current queue depth.</p>
IMFQ_QATTR_DEFPRIORITY	<p>Contains a decimal number in the range 0 to the MaxPriority of the queue manager.</p>
IMFQ_QATTR_DEFPERSISTENCE	<p>Contains one of the following:</p> <p>MQPER_PERSISTENT  MQPER_NOT_PERSISTENT</p>
IMFQ_QATTR_DEFINPUTOPENOPTION	<p>Contains one of the following:</p> <p>MQOO_INPUT_EXCLUSIVE  MQOO_INPUT_SHARED</p>
IMFQ_QATTR_DEFINITIONTYPE	<p>Contains one of the following:</p> <p>MQQDT_PREDEFINED  MQQDT_PERMANENT_DYNAMIC  MQQDT_TEMPORARY_DYNAMIC  MQQDT_SHARED_DYNAMIC</p>
IMFQ_QATTR_INHIBITGET	<p>Contains one of the following:</p> <p>MQQA_GET_ALLOWED  MQQA_GET_INHIBITED</p>
IMFQ_QATTR_HARDENGETBACKOUT	<p>Contains one of the following:</p> <p>MQQA_BACKOUT_HARDENED  MQQA_BACKOUT_NOT_HARDENED</p>
IMFQ_QATTR_INITIATIONQNAME	<p>Contains up to a 48-character value.</p>

Table 9. AutoOPERATOR-Supplied Variables for all MQSeries Messages (Continued)

<b>Variable name</b>	<b>Description</b>
IMFQ_QATTR_OPENINPUTCOUNT	Contains a decimal value.
IMFQ_QATTR_MAXQDEPTH	Contains a decimal number in the range 0 to 999999999.
IMFQ_QATTR_MAXMSGLENGTH	Contains a decimal number in the range 0 to 104857600.
IMFQ_QATTR_MSGDELIVERYSEQUENCE	Contains one of the following: MQMDS_FIFO MQMDS_PRIORITY
IMFQ_QATTR_OPENOUTPUTCOUNT	Contains a decimal value.
IMFQ_QATTR_PROCESSNAME	Contains up to a 48-character value.
IMFQ_QATTR_INHIBITPUT	Contains one of the following: MQQA_PUT_INHIBITED MQQA_PUT_ALLOWED
IMFQ_QATTR_RETENTIONINTERVAL	Contains a decimal number in the range 0 to 999999999.
IMFQ_QATTR_SHAREABILITY	Contains one of the following: MQQA_SHAREABLE MQQA_NOT_SHAREABLE
IMFQ_QATTR_STORAGECLASS	Contains up to a 8-character value.
IMFQ_QATTR_TRIGGERDAT	Contains up to a 64-character value
IMFQ_QATTR_TRIGGERDEPTH	Contains a decimal value.
IMFQ_QATTR_TRIGGERCONTROL	Contains one of the following: MQTC_OFF MQTC_ON
IMFQ_QATTR_TRIGGERMSGPRIORITY	Contains a decimal value in the range of 0 to the maximum priority.
IMFQ_QATTR_TRIGGERTYPE	Contains one of the following: MQTT_NONE MQTT_FIRST MQTT_EVERY MQTT_DEPTH
IMFQ_QATTR_USAGE	Contains one of the following: MQUS_NORMAL MQUS_TRANSMISSION

## What the MQS Selection Criteria and Action Specification Panel Fields Are

This section describes the fields for the panels introduced with MAINVIEW AutoOPERATOR for MQSeries: Selection Criteria - MQS and Action Specification - MQS. Use these panels to specify the selection criteria and the actions to perform with each Rule.

### Selection Criteria for MQS Panel

The Selection Criteria - MQS panel is used for specifying selection criteria, as shown in Figure 33. All criteria must be met before a Rule's actions will be taken.

```

BMC Software ----- Selection Criteria - MQS ----- AutoOPERATOR
COMMAND ==>                                     TGT --- SYSE

          Rule-set === AAORUL00          Rule-id === MQCICS

Queue Identification:                               (1 to 12 Queue Managers)
Manager(s) ==> CSQ1
Queue Id   ==> CICSPRD1.WAREHOUS.I NVENTORY

Message Identification:
Format     ==> MQFMT_CICS                       (Value from MD Format field)
Event Type ==>                                     (Enter ? for help)

          Sub  Len  Op  Value
Msgid     ==> 10_ : 2__ HE  A3C4_____
CorrelId  ==> 20_ : 1__ HG  1_____
Msg Buffer ==> 40_ : 12_ EQ  ITEM#3445_____

Press ENTER to continue, END return to Detail Control, CANCEL to cancel changes

```

Figure 33. Selection Criteria - MQS Field Description

Use the Selection Criteria - MQS panel to specify attributes of an event that the Rule must match before it will fire. Table 10 lists the Selection Criteria - MQS panel's field names and a description of each.

Table 10. Selection Criteria - MQS Panel's Field Names

Field Name	Description
<b>Manager(s)</b>	<p>Type a four-character queue manager name in this field. You can specify up to 12 managers, each separated by either blanks or commas.</p> <p>Other notes about this field:</p> <ul style="list-style-type: none"> <li>• Pattern matching is permitted in this field; <b>variables are not valid.</b></li> <li>• An entry in this field is required.</li> <li>• There is no default for this field.</li> </ul>
<b>Queue ID</b>	<p>Type 1 to 48 characters for the name of a specific MQSeries queue. If not specified, the source of the message can be any queue on which AutoOPERATOR is listening for messages.</p> <p>Other notes about this field:</p> <ul style="list-style-type: none"> <li>• Local and Shared variables are permitted. Pattern matching is also permitted.</li> <li>• An entry in this field is not required.</li> <li>• This field is left blank by default.</li> <li>• The specified queue must have an entry in BBPARM member AAOMQL00 for the Rule to monitor the queue.</li> </ul>
<b>Format</b>	<p>Type the message format in this field. Use to select messages based on the format found in the message descriptor. The following formats are recognized by AutoOPERATOR:</p> <p>MQFMT_EVENT: Event message</p> <p>USER: User defined format, any message format except MQFMT_EVENT; see the following list.</p> <p>MQFMT_NONE: No format name</p> <p>MQFMT_ADMIN: Command server request/reply message</p> <p>MQFMT_CHANNEL_COMPLETED: Channel completed message</p> <p>MQFMT_CICS: CICS information header</p> <p>MQFMT_COMMAND_1: Type 1 command reply message</p> <p>MQFMT_COMMAND_2: Type 2 command reply message</p> <p>MQFMT_DEAD_LETTER_HEADER: Dead-letter header</p> <p>MQFMT_IMS: IMS information header</p> <p>MQFMT_IMS_VAR_STRING: IMS variable string</p> <p>MQFMT_MD_EXTENSION: Message-descriptor extension</p> <p>MQFMT_PCF: User-defined msg in programmable command format</p> <p>MQFMT_REF_MSG_HEADER: Reference message header</p> <p>MQFMT_STRING: Message consisting entirely of characters</p> <p>MQFMT_TRIGGER: Trigger message</p> <p>MQFMT_WORK_INFO_HEADER: Work information header</p> <p>MQFMT_XMIT_Q_HEADER: Transmission queue header</p> <p>MQFMT_RF_HEADER: Rules and formatting header</p> <p>MQFMT_RF_HEADER_2: Rules and formatting header version 2</p> <p>This field has no default.</p>

Table 10. Selection Criteria - MQS Panel's Field Names (Continued)

Field Name	Description
<b>Event Type</b>	<p>Use 1 to 28 characters to define a particular event type for which messages will be selected when the message format is MQFMT_EVENT. For a list of valid values, type a '?'.</p> <p>Use only if the Format field is set to MQFMT_EVENT.</p> <p>Other notes about this field:</p> <ul style="list-style-type: none"> <li>• An entry in this field is not required</li> <li>• This field is left blank by default to fire on all event types</li> </ul>
<b>Msgid</b>	<p>Use this field to select messages containing a specific Msgid in the Message Descriptor Structure. Also, you can use the following input fields to specify a substring, a length and a special operator to use in comparisons:</p> <p><b>Sub</b> (Substring): Specifies the starting point of the comparison. Optional. Valid values are 1 to 24.</p> <p><b>Len</b> (Length): Specifies the length of the substring comparison. Optional. If omitted, length is the remainder of the field following the substring. Valid values are 0 to 24.</p> <p><b>Op</b> (Operator): Specifies the type of data and the type of comparison. Valid operators are EQ, NE, GT, LT, GE, LE, IN, EX, HE, HN, HG, HL, GX, LX, FO and FN. The default is EQ. To view a description of the operators, see the Help panel for this field.</p> <p><b>Value</b> (Value): Specifies the data value against which the Msgid field of the Message Descriptor Block is compared. The data type must match that which is indicated by the Op field. Local and Shared variables are permitted.</p> <p>Other notes about this field:</p> <ul style="list-style-type: none"> <li>• You can use this field only for messages that are not of format, MQFMT_EVENT</li> <li>• An entry in this field is not required.</li> <li>• This field is left blank by default.</li> </ul>

Table 10. Selection Criteria - MQS Panel's Field Names (Continued)

Field Name	Description
<b>CorrelId</b>	<p>Use this field to select messages containing a specific CorrelId in the Message Descriptor Structure. Also, you can use the following input fields to specify a substring, a length and a special operator to use in comparisons:</p> <p><b>Sub</b> (Substring): Specifies the starting point of the comparison. Optional. Valid values are 1 to 24.</p> <p><b>Len</b> (Length): Specifies the length of the substring comparison. Optional. If omitted, length is the remainder of the field following the substring.</p> <p><b>Op</b> (Operator): Specifies the type of data and the type of comparison. Valid operators are EQ, NE, GT, LT, GE, LE, IN, EX, HE, HN, HG, HL, GX, LX, FO and FN. Default is EQ. To view a description of the operators, see the Help panel for this field.</p> <p><b>Value</b> (Value): Specifies the data value against which the Msgid field of the Message Descriptor Block is compared. The data type should match that indicated by the Op field. Local and Shared variables are permitted.</p> <p>Other notes about this field:</p> <ul style="list-style-type: none"> <li>• An entry in this field is not required.</li> <li>• Allowed only for messages that are not of format MQFMT_EVENT.</li> <li>• This field is left blank by default.</li> </ul>

Table 10. Selection Criteria - MQS Panel's Field Names (Continued)

Field Name	Description
<b>Msg Buffer</b>	<p>(Application Data) Use this field to select all messages that contain a specified text string. The first 255 bytes of the message participate in the Text String matching.</p> <p>You can use the following input fields to specify a substring, a length and a special operator to use in comparisons:</p> <p><b>Sub</b> (Substring): Specifies the starting point for a comparison to take place. Optional. Valid values are 1 to 255.</p> <p><b>Len</b> (Length): Specifies the length of the substring comparison. Optional. If omitted, length is the remainder of the field following the substring. Valid values are 0 to 255.</p> <p><b>Op</b> (Operator): Specifies the type of data and the type of comparison. Valid operators are EQ, NE, GT, LT, GE, LE, IN, EX, HE, HN, HG, HL, GX, LX, FO and FN. Default is EQ. To view a description of the operators, see the Help panel for this field.</p> <p><b>Value</b> (Value): Specifies the data value against which the data from the Msg Buffer is compared. The data type should match that indicated by the Op field. Local and Shared variables are permitted.</p> <p><b>Note:</b> Be aware that the variable, IMFTEXT, which is passed to an EXEC when invoked from a Rule, has the same limitations as the MSG Buffer of the Selection Criteria Panel, 255 bytes. For more information regarding IMFTEXT, see Chapter 4 in the <i>MAINVIEW AutoOPERATOR Advanced Automation Guide for REXX EXECs</i> and the <i>MAINVIEW AutoOPERATOR Advanced Automation Guide for CLIST EXECs</i>. To access data in the message beyond 255 bytes, you must use an EXEC to GET the message from the queue. Using the IMFEXEC MQI GET statement in an EXEC, you can access up to 32767 bytes of the message content.</p> <p>Other notes about this field:</p> <ul style="list-style-type: none"> <li>• An entry in this field is not required.</li> <li>• Allowed only for messages that are not of format MQFMT_EVENT.</li> <li>• This field is left blank by default.</li> </ul>

## Action Specification for MQS Panel

The Action Specification - MQS panel, shown in Figure 34, is one of the two panels used for specifying an action when a Rule fires; ALERT Action(s) is the other.

Additionally, the Action Specification - MQS panel can be used to remove the Dead Letter Header and as a Destination Queue Manager field for use with the Copy function.

```

BMC Software ----- Action Specification - MQS ----- AutoOPERATOR
COMMAND ==>                                         TGT --- SYSE

          Rule-set === AAORUL00                Rule-id === DLH
Automation Actions:
Journal          ==>
EXEC Name/Parms  ==>
Cmd (Type MQ )  ==>

Set Variable     ==>                               ==>
Notify          ==>                               Outboard Pager ID
Info           ==>

DOM Id          ==>                               Delete Operator Message
Issue WTO Msg   ==>

MQSeries Automation Actions:
Keep Message    ==> YES (Yes or NO)                Remove DLH ==> (Yes or No)
Destination Que ==>
Destination QMgr ==>

Press ENTER to continue, END return to Detail Control, CANCEL to cancel changes
  
```

Figure 34. Action Specification - MQS Field Description

Table 11 lists the Action Specification - MQS panel's field names and a description of each.

Table 11. Action Specification - MQS Panel's Field Names

Field name	Description
<b>EXEC Name/Parms</b>	<p>This field allows up to 56 characters and is used to schedule an EXEC to perform advanced automation that cannot be performed by a Rule. Local and SHARED variables are valid. The EXEC parameters may be specified, which are passed when the EXEC is scheduled for execution.</p> <ul style="list-style-type: none"> <li>An entry in this field is not required.</li> <li>This field is left blank by default.</li> </ul>

Table 11. Action Specification - MQS Panel's Field Names (Continued)

Field name	Description
<b>Cmd(Type)</b>	<p>This field allows up to 126 characters and is used to issue one or more console commands. Local and SHARED variables are valid. The following is a list of command types and origin:</p> <p><b>Blank</b> No command is issued.  <b>MVS</b> MVS command is issued from event originating address space.  <b>BBI</b> BBI command is issued from the BBI-SS PAS address space.  <b>IMS</b> IMS command is issued from the BBI-SS PAS address space.  <b>CICS</b> CICS command is issued from the BBI-SS PAS address space.  <b>MQ</b> MQ command is issued from the BBI-SS PAS address space.  <b>SS</b> MVS command is issued from the BBI-SS PAS address space.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>• Multiple commands are delimited by two colons.</li> <li>• CICS commands must contain a target. For example: <code>target: commandtext</code></li> <li>• If the Rule event type is not MQS or the targeted local-system OS/390 queue manager is not the queue manager firing the Rule, MQ commands must begin with the name of the queue manager processing the specified command. For example: <code>queuemgrname: command text</code></li> </ul> <p>Optional keywords for command type MQ:</p> <ul style="list-style-type: none"> <li>• <b>LM:</b> Name of local MQSeries queue manager. This keyword is necessary only when issuing a command to a remote queue manager connected to an OS/390 queue manager that is not the current queue manager firing the Rule. The keyword can be abbreviated to L.</li> <li>• <b>QM:</b> Name of the remote MQSeries queue manager to which the command should be sent. This is an optional keyword and can be abbreviated to Q.</li> <li>• <b>PCF:</b> Specifies command should be issued in Programmable Command Format. This is an optional keyword (default is N). Most non-OS/390 platforms require commands in PCF format (Y). Can be abbreviated to P.</li> <li>• <b>CQ:</b> Name of command queue. Optional keyword used in conjunction with QM parameter. Default is SYSTEM.ADMIN.COMMAND.QUEUE. Can be abbreviated to C. If QM is not specified (or is equal to LM), the CQ keyword will be ignored.</li> </ul> <p>Example 1: Issue ALTER QMGR on local MQSeries queue manager MQ03.</p> <p><b>CMD TYPE(MQ) ==&gt; MQ03: ALTER QMGR INHIBTEV(ENABLED)</b></p> <p>Example 2: Issue ALTER QMGR on remote queue manager through local queue manager MQ03, where MQ03 is the queue manager on which the PUT was issued that caused the Rule to fire. The local queue manager has a connection with the remote queue manager.</p> <p><b>CMD TYPE(MQ) ==&gt; QM(WINT. TE01) P(Y): ALT QMGR INHIBTEV(ENABLED)</b></p> <p>Example 3: Issue ALTER QMGR on remote queue manager through local queue manager MQ04, even though the queue manager on which the PUT was issued is MQ03.</p> <p><b>CMD TYPE(MQ) ==&gt; LM(MQ04) QM(WINT. TE01) P(Y): ALT QMGR INHIBTEV(ENABLED)</b></p>

Table 11. Action Specification - MQS Panel's Field Names (Continued)

Field name	Description
<b>Set Variable</b>	<p>In the first entry, up to 33 characters are allowed. In this field specify the name of the variable to be modified. Only SHARED variables will be created or updated. SHARED and local variables and constants may be used to create the variable name.</p> <p>In the second entry, up to 17 characters are allowed. SHARED and local variables and constants are valid. Use this field to assign a new value (using a variable or unsigned constant) or modify (using a signed constant).</p> <ul style="list-style-type: none"> <li>• An entry in this field is not required.</li> <li>• This field is left blank by default.</li> </ul>
<b>Notify</b>	<p>This field allows up to 33 characters and is used to store the telephone number or AutoOPERATOR Elan Operator ID to be notified using an outboard pager.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>• Specify the telephone number exactly as you would dial it on the telephone.</li> <li>• The Elan Operator ID must exist in the Elan Operator Profile.</li> <li>• This field is used in conjunction with the Info ===&gt; field.</li> </ul> <ul style="list-style-type: none"> <li>• Local and SHARED variables are valid.</li> <li>• An entry in this field is not required.</li> <li>• This field is left blank by default.</li> </ul>
<b>Info</b>	<p>This field allows up to 80 characters and is used to specify the information to appear in the display area of an outboard pager managed by AutoOPERATOR Elan.</p> <ul style="list-style-type: none"> <li>• Local and SHARED variables are valid.</li> <li>• An entry in this field is not required.</li> <li>• This field is left blank by default.</li> </ul>
<b>DOM ID</b>	<p>This field allows up to 33 characters and is used to allow a Rule to issue a Delete Operator Message (DOM) to delete a highlighted message from the operator console.</p> <ul style="list-style-type: none"> <li>• SHARED variables are valid.</li> <li>• An entry in this field is not required.</li> <li>• This field is left blank by default.</li> </ul>
<b>Issue WTO Msg</b>	<p>This field allows up to 125 characters and allows a Rule to issue a message to a console (Write To Operator).</p> <p>Specify the message you would like issued to the operator console. The message is issued without routing or descriptor codes. The message may be selected by another Rule.</p> <ul style="list-style-type: none"> <li>• An entry in this field is not required.</li> <li>• This field is left blank by default.</li> </ul>

Table 11. Action Specification - MQS Panel's Field Names (Continued)

Field name	Description
<b>Keep Message</b>	<p>You can enter Yes or No to specify whether or not to keep the message in the source MQSeries queue.</p> <p>Yes      Do not delete the message from the source MQSeries queue.            No      Delete the message from the source MQSeries queue.</p> <ul style="list-style-type: none"> <li>• An entry in this field is not required.</li> <li>• The default is Yes.</li> </ul>
<b>Destination Queue</b>	<p>You can enter up to 48 characters in this field for the name of the MQSeries queue to copy/move messages to.</p> <ul style="list-style-type: none"> <li>• Local and SHARED variables are valid.</li> <li>• An entry in this field is not required.</li> <li>• This field is left blank by default.</li> </ul> <p>Using “Keep Message=NO and Destination Queue” causes a message to be moved from one queue to another while “Keep Messages=YES and Destination Queue” causes a message to be copied from one queue to another.</p> <p>If Keep(YES) and destination queue are both specified, the Keep/Copy will be synchronized.</p>
<b>Remove DLH</b>	<p>During a Copy or Move operation, if YES is specified in this field, the original dead letter message is rebuilt by removing the Dead Letter Header and restoring the original Encoding, CodedCharSetId and Format fields back into the message descriptor.</p> <p><b>Note:</b> After removing the DLH, any Rules following this Rule will not see the DLH but will see the rebuilt message.</p> <p>Possible values are            Yes            No</p> <p>Using this function, you can strip a dead letter message of all information regarding dead letter processing and forward it to an alternate queue for normal processing. In addition, when using this option, a message passed to a Rule-invoked EXEC will not contain the Dead Letter Header.</p>
<b>Destination QMgr</b>	<p>A queue manager to which the message is targeted during a Copy or Move operation can be specified in this field. Without this specification, the message is PUT to a destination queue on the current queue manager for which the Rule fired. To use this field, the Destination Queue field must be set. The specified queue manager ID is placed into the object descriptor so that normal MQSeries queue and queue manager resolution take place.</p>

## Tracking Automation Statistics for MQS Events

The *MAINVIEW AutoOperator Basic Automation Guide* describes how to activate the Automation Reporter application, what it does, and how you can read the data collected by the application. For MAINVIEW AutoOPERATOR for MQSeries, the data collected by the Automation Reporter can be seen in the record types **EVNT** and **ACTN**.

The Automation Reporter record for event type **EVNT** tracks the number of times the event type MQS appears in the Event Statistics Table. An example of the record looks like Figure 35.

---

```
SYSC, EP01, 310, BBPLEX01, "19970210", "09: 47", 13, EVNT, ONLY, "19970210",  
"10: 00", "PUT_I NHI BI TED ", 15, 15, MQS, "", 4, EOR  
SYSC, EP01, 310, BBPLEX01, "19970210", "09: 47", 13, EVNT, ONLY, "19970210",  
"10: 00", "CHANNEL_STOPPED ", 1, 1, MQS, "", 4, EOR
```

---

Figure 35. Example of EVNT Automation Reporter Record

The Automation Reporter record for event type **ACTN** tracks the number of times automation handled events and what actions were taken. This record includes tracking of MQS events. An example of the record looks like Figure 36.

---

```
SYSC, EP01, 310, BBPLEX01, "19970210", "10: 00", 10, ACTN, ONLY, "19970210",  
"10: 00", 13, 7, MSG, 67, 0, 0, 0, 0, 0, CMD, 9, 0, 0, 0, 0, 0, JRNL, 3, 0, 0, 0, 0, 0,  
TIME, 0, 0, 0, 0, 0, 0, ALRT, 0, 0, 0, 0, 0, 0, EXT, 0, 0, 0, 0, 0, 0, IMS, 0, 0, 0, 0, 0, 0,  
CI CS, 0, 0, 0, 0, 0, 0, DB2, 0, 0, 0, 0, 0, 0, JES3, 0, 0, 0, 0, 0, 0,  
ALRM, 54, 0, 0, 0, 0, 0, VAR, 1, 0, 0, 0, 0, 0, MQS, 1, 0, 0, 0, 0, 0, EOR
```

---

Figure 36. Example of ACTN Automation Reporter Record

For a complete description of the data shown here, refer to the *MAINVIEW AutoOPERATOR Basic Automation Guide*.

## Chapter 5. Viewing AutoOPERATOR for MQSeries Automation Statistics

This chapter describes

- How to view AutoOPERATOR for MQSeries automation statistics from the AutoOPERATOR for MQSeries Workstation
- How to use BBI control commands for AutoOPERATOR for MQSeries

### Using the AutoOPERATOR for MQSeries Workstation

AutoOPERATOR for MQSeries provides the MQSeries Workstation panel (Figure 37), which allows you to view both performance and queue information from a single panel. As with all other BBI displays, the MQSeries Workstation can be pointed to from one BBI-SS PAS target to another.

Use the MQSeries Workstation panel, shown in Figure 37, to view statistics that show how much automation of MQSeries events has been accomplished and to see which queues are active for this BBI-SS PAS.

```
BMC Software ----- MQSeries Workstation ----- AutoOPERATOR
COMMAND ==>>
Interval ==> 3      INPUT                                TGT ==>> EP01
Commands: ALL, EQI xxx, NEQI xxx, EE xxx, NEE xxx      DATE --- 02/02/11
                                                    TIME --- 08:20:32
----- Performance Statistics -----
Total Queues Included                6
Total MQS Messages Arrived           1  Total MQS Messages Handled           0
Instrument. Events Arrived            1  Instrument. Events Handled           0
User Messages Arrived                 0  User Messages Handled                 0
Current Instr. Arrival Rate/Min       0  Peak Instr. Arrival Rate/Min         0
Rule Generated Alerts                  0  Rule Triggered EXECs                 0
----- Queue Management -----
Queue Manager                        Queues   Event Ques   Enabled
                                   Incl./Total  Included   Events
-----
CSQ1                                 6/24
***** Bottom of Data *****
```

Figure 37. MQSeries Workstation Panel

This panel is divided into two parts: Performance Statistics and Queue Management. With these two areas, you can see statistics on both the performance of MQSeries and the status of individual queues.

## Primary Commands

Table 12 lists and describes all the primary commands you can type on the MQSeries Workstation panel.

Table 12. MQSeries Workstation Panel Primary Commands

Primary command	Description
ALL	Shows all MQSeries queue managers for which AutoOPERATOR for MQSeries will attempt automation.
EQI xxx	<p>Shows only those queue managers where AutoOPERATOR for MQSeries is including instrumentation events for automation. Use xxx to specify which instrumentation events you want to view, where xxx may be any (or all) of the following:</p> <p><b>Q</b>      SYSTEM.ADMIN.QMGR.EVENT</p> <p><b>P</b>      SYSTEM.ADMIN.PERFM.EVENT</p> <p><b>C</b>      SYSTEM.ADMIN.CHANNEL.EVENT</p> <p>For example, to view only the queue managers that include SYSTEM.ADMIN.QMGR.EVENT and SYSTEM.ADMIN.CHANNEL.EVENT, type: <b>eqi qc</b></p> <p>This command cannot be used if AutoOPERATOR coexists with the PATROL for MQ products.</p>
NEQI xxx	<p>This command does the opposite of the EQI command. With this command you can specify which queues not to view. Use xxx to specify which instrumentation events you do not want to view, where xxx may be any (or all) of the following:</p> <p><b>Q</b>      SYSTEM.ADMIN.QMGR.EVENT</p> <p><b>P</b>      SYSTEM.ADMIN.PERFM.EVENT</p> <p><b>C</b>      SYSTEM.ADMIN.CHANNEL.EVENT</p> <p>For example, to view the queue managers that do not include SYSTEM.ADMIN.CHANNEL.EVENT, type</p> <p><b>neqi c</b></p> <p>This command cannot be used if AutoOPERATOR coexists with the PATROL for MQ products.</p>

Table 12. MQSeries Workstation Panel Primary Commands (Continued)

Primary command	Description
EE xxx	<p>Shows only queue managers with xxx events enabled, where xxx may be any (or all) of the following:</p> <p><b>S</b> STRSTPEV (Start/Stop event)</p> <p><b>L</b> LOCALEV (Local event)</p> <p><b>I</b> INHIBTEV (Inhibit event)</p> <p><b>P</b> PERFMEV (Performance event)</p> <p><b>R</b> REMOTEV (Remote event)</p> <p>For example, to view only those queue managers which include remote events, type <b>ee r</b></p>
NEE xxx	<p>This command works the opposite of the EE xxx. Use this command to show only the queue managers that do not have xxx events enabled, where xxx may be any one (or all) of the following:</p> <p><b>S</b> STRSTPEV (Start/Stop event)</p> <p><b>L</b> LOCALEV (Local event)</p> <p><b>I</b> INHIBTEV (Inhibit event)</p> <p><b>P</b> PERFMEV (Performance event)</p> <p><b>R</b> REMOTEV (Remote event)</p> <p>For example, to view the queue managers which do not include remote events, type: <b>nee r</b></p>

**Note:** The status of events is taken when AutoOPERATOR for MQSeries connects to the queue manager. If the status is changed, AutoOPERATOR for MQSeries must be disconnected and then reconnected to obtain the correct status. To request this status change, use the BBI control command .RESET:

. RESET QM xxxx

where xxxx is the four-character queue manager ID.

## MQSeries Workstation Fields

Figure 38 shows the fields of the MQSeries Workstation panel.

```

BMC Software ----- MQSeries Workstation ----- AutoOPERATOR
COMMAND ==>                                     TGT ==> EP01
Interval ==> 3      INPUT                          DATE --- 02/02/11
Commands: ALL, EQI xxx, NEQI xxx, EE xxx, NEE xxx      TIME --- 08:20:32
----- Performance Statistics -----
Total Queues Included                               6
Total MQS Messages Arrived                          1  Total MQS Messages Handled          0
Instrument. Events Arrived                          1  Instrument. Events Handled          0
User Messages Arrived                               0  User Messages Handled              0
Current Instr. Arrival Rate/Min                      0  Peak Instr. Arrival Rate/Min      0
Rule Generated Alerts                               0  Rule Triggered EXECs              0
----- Queue Management -----
Queue Manager                                     Queues   Event Ques   Enabled
                                                Incl./Total  Included     Events
-----
CSQ1                                             6/24                                     SLIPR
***** Bottom of Data *****

```

Figure 38. MQSeries Workstation Panel Fields

## Performance Statistics

Figure 39 shows the Performance Statistics portion of the MQSeries Workstation panel.

```

----- Performance Statistics -----
Total Queues                               174  Total Queues Included          9
Total MQS Messages Included                 1  Total MQS Messages Handled    1
Instrument. Events Included                 1  Instrument. Events Handled     1
User Messages Included                     0  User Messages Handled          0
Current Instr. Arrival Rate/Min            0  Peak Instr. Arrival Rate/Min  0
Rule Generated Alerts                      0  Rule Triggered EXECs          0

```

Figure 39. MQSeries Workstation - Performance Statistics

Use this portion of the panel when you are trying to evaluate how often the automation of MQSeries events is occurring on your system.

The Performance Statistics field names are described in Table 13.

Table 13. Performance Statistics Field Names

Field name	Description
Total Queues	Total number of queues found in all currently connected queue managers
Total Queues Included	Total number of queues eligible for automation by AutoOPERATOR for MQSeries
Total MQS Messages Included	Total number of MQS messages seen by AutoOPERATOR (eligible for automation)

Table 13. Performance Statistics Field Names (Continued)

<b>Field name</b>	<b>Description</b>
Total MQS Messages Handled	Total number of MQ messages where at least one Rule has fired
Instrument. Events Included	Total number of messages seen by AutoOPERATOR (eligible for automation) that were instrumentation events
Instrument. Events Handled	Total number of MQ messages that had at least one Rule fired and that were instrumentation events
User Messages Included	Total number of User MQ messages seen by AutoOPERATOR that were eligible for automation
User Messages Handled	Total number of User MQ messages where at least one Rule fired
Current Instr. Arrival Rate/Min	Arrival rate of instrumentation events per minute
Peak Instr. Arrival Rate/Min	Highest arrival rate of instrumentation events per minute
Rule Generated Alerts	Total number of event type (MQS) ALERTs that were generated by Rules
Rule Triggered EXECs	Total number of event type (MQS) EXECs that were triggered by Rules

In general, the figures shown in the Performance Statistics portion of the MQSeries Workstation panel are useful after you have determined the average amount of automation activity. Then, if there is a change in statistics, you can have a better sense of whether the change is due to an abnormal condition or if it is an expected and acceptable amount.

## Viewing Queue Management

Figure 40 shows the Queue Management portion of the MQSeries Workstation panel.

----- Queue Management -----			
Queue Manager	Queues Incl. /Total	Event Queues Included	Enabled Events
CSQ3	19/194		
CSQ4	9/174	QCP	SLIPR
***** Bottom of Data *****			

Figure 40. MQSeries Workstation - Queue Management

Use this portion of the panel to see which queue managers are active, the number of queues that are eligible for automation, and the total number of queues on the queue manager. This portion also shows you the type of Event Queues that are included and the type of Events that are enabled.

The Queue Management field names are described in Table 14.

Table 14. Queue Management Field Names

Field name	Description
Queue Manager	Name (SSID) of MQSeries queue manager. The list includes only queue managers to which AutoOPERATOR has attempted to connect. AutoOPERATOR will attempt to connect to queue managers if they are defined in a Rule.
Queues Incl./Total	Number of queues that AutoOPERATOR is listening to for automation, followed by the total number of queues on that queue manager.
Event Queues Included	Instrumentation queues that are included by AutoOPERATOR for automation.  <b>P:</b> Performance, <b>Q:</b> Queue Manager, <b>C:</b> Channel, <b>ALL:</b> All
Enabled Events	Events that were found to be enabled when the connection was made to this queue manager.  <b>I:</b> Inhibit, <b>L:</b> Local, <b>P:</b> Performance, <b>R:</b> Remote, <b>S:</b> Strstp,

---

## Chapter 6. Using AutoOPERATOR for MQSeries Solutions

This chapter describes how to use the following AutoOPERATOR for MQSeries solutions:

- Implementation Considerations
- Dead Letter Queue Management solution
- System Queue Archival solution
- Service Interval Performance solution
- Queue Depth Management solution
- Channel Availability solution
- Basic Intercommunication solution

---

### Implementation Considerations

The AutoOPERATOR solutions are primarily Rule-based and, therefore, easy to implement.

However, if your site does not have the IBM library for SAA REXX/370 installed and you plan to run the System Queue Archival solution, the Dead Letter Queue solution, or the Channel Availability solution, you must review the chapter titled “Activating the REXX/370 Alternate Library” in the *MAINVIEW AutoOPERATOR Customization Guide*. These solutions require that either the SAA REXX/370 library or the REXX370/ alternate library be installed.

All of the Rules for these solutions are in Rule Sets AAORULBQ, AAORULBH, and AAORULBR. You must edit and enable the appropriate Rules before the solutions are usable.

In addition, ensure that the automation strategy for Rules is set to INDIVIDUAL and the Rule Sets AAORULBQ, AAORULBH, and AAORULBR each have a strategy set to ALL. For more information about how to set automation strategy for AutoOPERATOR Rules and Rule Sets, refer to the *MAINVIEW AutoOPERATOR Basic Automation Guide*.

### System Queue Archival

The System Queue Archival solution uses an EXEC named QMQUNLDQ. This EXEC writes to the SYSIN file to pass control statements to IBM MQSeries utility programs. To provide serialization to the SYSIN file, QMQUNLDQ uses the IMFEXEC VENQ service. The minor name that QMQUNLDQ enqueues on is ARCSYSIN. Therefore, BMC Software strongly recommends that any user-written EXEC that uses the SYSIN file also uses the same technique. Any EXEC that uses the SYSIN file and does not use VENQ serialization may create unpredictable results.

---

## Using the Dead Letter Queue Solution

The Dead Letter Queue solution performs different functions that can be invoked separately or together. Use the Dead Letter Management portion of the solution when you want to

- Be informed automatically by an ALERT when a valid dead letter (a message with a dead letter header) arrives in the dead letter queue
- Have invalid dead letters (messages without a dead letter header) moved from the dead letter queue to an alternate queue and have an ALERT created after the messages are moved

Use the Dead Letter Aging Management portion of the solution when you want to fire an EXEC once a day that generates two ALERTs:

- A nonvolatile ALERT containing the total number of messages that are more than one day old
- A nonvolatile ALERT containing the total number of messages that are more than one week old from the dead letter queue.

The following table describes the different Rules that you can implement to perform various features of the Dead Letter Queue solution.

Rule ID	EXEC scheduled	Triggering event
MQDEDQ01	QMDEDQ1	Fires when an invalid dead letter arrives in the queue
MQDEDQ02	QMDEDQ1	Creates nonvolatile ALERTs which report on the total number of messages that are more than 1 day old and the total number of messages that are more than one week old in both the dead letter queue and the alternate queue for invalid dead letters
MQDEDQ03		Fires when a valid dead letter arrives in the queue

## Security Considerations

To use the Dead Letter Queue solution, AutoOPERATOR needs access authority to MQSeries resources as follows:

- Alter access to create the queue used to hold invalid dead letters and update access to put messages onto it. Specify the queue name when executing QMDEDQ1.
- Update authority for the dead letter queue on each queue manager that is automated by AutoOPERATOR.

## Implementing the Dead Letter Queue Solution

To invoke any of the Rules for the Dead Letter Queue solution:

1. Enter `INCL` to add the dead letter queue names to BBPARM member AAOMQLxx.

2. Issue the `RESET MQ xx` command where `xx` is the two-character suffix of the `AAOMQLxx` member.

The following sections describe in greater detail how to enable specific Rules that make up the Dead Letter Queue solution. To activate the complete solution, you must enable all three Rules as described in these sections.

## Enabling the Dead Letter Management Rules

The Dead Letter Management portion of the solution covers managing two different kinds of dead letters: dead letters, which are valid dead letters and have dead letter headers, and invalid dead letters, which do not have proper dead letter headers. The instructions in this section describe how to enable the Rules for both kinds of dead letters.

**Invalid Dead Letter Management:** To invoke the portion of the solution which manages invalid dead letters:

1. Invoke an EXEC called `QMDEDQ1`.

This EXEC creates a queue to which invalid dead letters (dead letters without proper headers) will be moved. You must pass the name of the queue manager ID as the first parameter and pass the name of the queue (to which invalid dead letters will be written). You can choose the queue name but the queue's attributes are modeled after the actual system dead letter queue.

To invoke the `QMDEDQ1` EXEC:

- a. Type the following command from the BBI terminal session log display:

```
%QMDEDQ1 queuemgrID queue. name
```

where `queuemgrID` is the queue manager ID and `queue. name` is the new queue name.

- b. You must invoke this EXEC for each queue manager that this solution will manage and you must use the same queue name so that the solution can manage all the queues.

2. Edit and enable the Rule `MQDEDQ01` in the `AAORULBQ` Rule Set.

When enabled, the `MQDEDQ01` Rule fires when an invalid dead letter arrives and notifies the operator with an `ALERT` that the letter has been moved to the queue created by EXEC `QMDEDQ1`.

To enable and edit the `MQDEDQ01` Rule:

1. Go to the Automation Control panel where the list of Rule Sets is displayed.
2. Ensure that the `AAORULBQ` Rule Set is enabled. If not, enable the Rule Set by typing the (E)nable line command next to the Rule Set name.
3. Select the `AAORULBQ` Rule Set.
4. Select the Rule `MQDEDQ01`.

5. Edit MQDEDQ01 using the Rules Processor application panels.

Press Enter to navigate through the panels and make the following changes:

- a. On the Selection Criteria panel under Queue Identification, change XXXX to the queue manager IDs of the queue managers that this Rule will manage. Remember that these queue manager names must be specified in BBPARM member AAOMQLxx and then reset with the BBI command `. RESET MQ xx`.
- b. On the Action Specification panel change the Destination Queue from XXX. XXX. XXX to the name of the queue created for invalid dead letters.
- c. On the Alert Action 1 panel change XXX. XXX. XXX in the ALERT text to the name of the queue created for invalid dead letters.

6. Enable MQDEDQ01 by performing one of the following actions:

- Enable the Rule on the Rule Processor Detail Control panel.
- Use the (E) nabl e line command on the Ruleset Overview panel.
- Issue the BBI command:  
**. T RULE, ENA, MQDEDQ01**

**Valid Dead Letter Management:** To invoke the portion of the solution which manages valid dead letters, you must enable the MQDEDQ03 Rule which fires every time a valid dead letter arrives in the queue and issues an ALERT to notify the operator that a valid dead letter has arrived.

To enable and edit the MQDEDQ03 Rule:

1. Follow Step 1 through Step 3 on page 123.
2. Select the Rule MQDEDQ03.
3. Edit MQDEDQ03 using the Rules Processor application panels.

Press Enter to navigate through the panels and make the following change:

- a. On the Selection Criteria panel under Queue Identification, change XXXX to the queue manager IDs of the queue managers that this Rule will manage. Remember that these queue manager names must be specified in BBPARM member AAOMQLxx and then reset with the BBI command `. RESET MQ xx`.

4. Enable MQDEDQ03 by performing one of the following actions:

- Enable the Rule on the Rule Processor Detail Control panel.
- Use the (E) nabl e line command on the Ruleset Overview panel.
- Issue the BBI command:  
**. T RULE, ENA, MQDEDQ03**

## Enabling the Dead Letter Aging Management Rule

To invoke the Dead Letter Aging Management portion of the solution, you must edit and enable the MQDEDQ02 Rule in the AAORULBQ Rule Set.

The MQDEDQ02 Rule creates nonvolatile ALERTs which report on the total number of messages that are more than one day old and the total number of messages that are more than one week old in both the dead letter queue and the alternate queue for invalid dead letters (dead letters without proper headers). Note that nonvolatile ALERTs are written to DASD and can therefore survive a BBI-SS PAS cold start or even an OS/390 IPL.

To enable and edit the MQDEDQ02 Rule:

1. Go to the Automation Control panel where the list of Rule Sets is displayed.
2. Ensure that the AAORULBQ Rule Set is enabled. If not, enable the Rule Set by typing the (E)nabl e line command next to the Rule Set name.
3. Select the AAORULBQ Rule Set.
4. Select the Rule MQDEDQ02.
5. Edit MQDEDQ02 using the Rules Processor application panels.

Press Enter to navigate through the panels and make the following change:

- a. On the Selection Criteria panel specify what time of day you want the Rule to fire and set the interval to 24:00:00.
6. Enable the MQDEDQ02 Rule by performing one of the following actions:
    - Enable the Rule on the Rule Processor Detail Control panel.
    - Use the (E) nabl e line command on the Ruleset Overview panel.
    - Issue the BBI command:  
**. T RULE, ENA, MQDEDQ02**

## Stopping the Dead Letter Queue Solution

To inactivate any part of the solution, disable the Rules by performing one of the following actions:

- Disable the Rule on the Rule Processor Detail Control panel.
- Use the (D) i sabl e line command on the Ruleset Overview panel.
- Issue the BBI command:  
**. T RULE, DIS, rul eID**

where rul eID is the Rule ID of the Rule you want to disable.

## Using the System Queue Archival Solution

Use the System Queue Archival solution to move messages from a system instrumentation event queue (or event queue) that has reached its capacity. The messages for that queue are off-loaded automatically to a sequential generation data group (GDG) data set.

The Archival solution uses a Rule that is fired when an event queue fills to a user-specified percentage. The messages within the queue are then off-loaded by an MQSeries offload utility and moved to an OS/390 generation data set—thereby freeing the queue to receive more messages.

Rule ID	Default state	EXEC scheduled	Triggering event	Default value
MQARC001	Disabled	QMQUMLDQ	Fires when an event queue fills to a user-specified percentage.	N/A
MQARC002 - MQARC005	Disabled	None	Fires when AutoOPERATOR connects to the queue manager. These Rules Set variables are used by the solution during archival of the messages.	
MQARC002	Disabled	None	Specifies the name of the GDG to be used as archive.	No default value
MQARC003	Disabled	None	Causes an MQSeries offload utility to move the messages to the specified SYSOUT class.	*
MQARC004	Disabled	None	Specifies the device unit to which a generation data set is allocated.	SYSDA
MQARC005	Disabled	None	Specifies the block size with which a generation data set is allocated.	0
MQARC006	Disabled	None	Specifies the number of tracks with which a generation data set is allocated.	(190,19)

## Security Considerations

To use the System Queue Archival solution, MAINVIEW AutoOPERATOR needs access authority to MQSeries resources as follows:

- Read access to the SYSTEM.ADMIN.QMGR.EVENT queue.
- Read access to the SYSTEM.ADMIN.PERFM.EVENT queue.
- Read access to the SYSTEM.ADMIN.CHANNEL.EVENT queue.
- Read access to the SYSBMC.ssid.EVENTS queue (if running Event Listener).
- Read access to the SYSBMC.DISTRIB.EVENTS queue (if running Event Listener).
- Alter access to create and delete the work queue BBOMVAO.SYSTEM.ADMIN.\*.EVENT.D\*.T\*, where the first \* is QMGR, PERFM or CHANNEL, the second \* is the julian date, and the third \* is the time (in the format

HHMM) used for copying the contents of the system administration queue before unloading to the GDG dataset.

## Implementing the System Queue Archival Solution

To invoke the System Queue Archival solution, do the following:

1. Add the names of the queue managers which this solution will manage to BBPARM member AAOMQLxx
2. Issue the `. RESET MQ xx` command where `xx` is the two-character suffix of the AAOMQLxx member

## Enabling the System Queue Archival Rules

To invoke the System Queue Archival solution, perform the following steps:

1. Edit and execute JCL in BBSAMP member QMARCDEF to create the Generation Data Group (GDG) Base.
2. Invoke the Rule Processor application by typing **RULES** from any BBI terminal session screen.
3. Ensure that the AAORULBQ Rule Set is enabled. If not, enable the Rule Set by typing the (E)nable line command next to the Rule Set name.
4. Select the AAORULBQ Rule Set.
5. Select the Rule MQARC002.
6. Edit the Rule using the Rule Processor application panels.
  - a. On the Action Specification panel and in the Set Variable field replace:  
`**GDGBASE**`  
with the name of the GDG base specified in QMARCDEF
7. By default, messages from the MQSeries offload utility are sent to the default SYSOUT class, or \*. To use another class, edit the Rule MQARC003.
  - a. On the Action Specification panel and in the Set Variable field, replace:  
`**OUTPUT**`  
with the SYSOUT class to be used.
8. By default, the archival data set is allocated on the device SYSDA. To use another device, edit the Rule MQARC004.
  - a. On the Action Specification panel and in the Set Variable field, replace:  
`**OUTUNIT**`  
with the device number to be used.
9. By default, the archival data set is allocated with the block size 0. To use another block size, edit the Rule MQARC005.
  - a. On the Action Specification panel and in the Set Variable field, replace:

**\*\*OUTBLK\*\***

with the block size to be used.

10. By default, the archival data set solution primary allocation is 190 tracks, and secondary allocation is 19 tracks. To change the number of tracks, edit the Rule MQARC006.

- a. On the Action Specification panel and in the Set Variable field, replace:

**\*\*OUTTRK\*\***

with the number of primary tracks to be allocated or (<primary>, <secondary>).

11. After editing the Rule, load the GDG base name into its variable by issuing the following BBI command:

**. E MQ**

12. Select the Rule MQARC001.

13. Edit the Rule using the Rule Processor application panels.

- a. On the Selection Criteria panel under Queue Identification, change XXXX to the queue manager IDs of the queue managers that this Rule will manage. Remember that these queue manager names must be specified in BBPARM member AAOMQLxx and then reset with the BBI command **. RESET MQ xx**.

14. Enable the MQARC001 and MQARC002 rules and all necessary MQARC003-MQARC006 Rules by performing one of the following actions:

- Enable the Rules on the Rule Processor Detail Control panel.
- Use the (E) enable line command on the Ruleset Overview panel.
- Issue the BBI commands:

**. T RULE, ENA, MQARC001**

**. T RULE, ENA, MQARC002**

**. T RULE, ENA, MQARC003**

**. T RULE, ENA, MQARC004**

**. T RULE, ENA, MQARC005**

**. T RULE, ENA, MQARC006**

**Note:** The three event queues are, by default, set to be archived when 50 percent full. To use a value other than 50 percent, change the value that the variable IMFQEPFL is compared to in the Rule's selection criteria. This value is changed on the Selection Variable Criteria panel for Rule MQARC001. There is a compare entry on the panel for each of the three event queues.

When the Rule is enabled and an event queue fills to a certain point, the Rule fires and the messages in the queue are moved to an OS/390 generation data set. The system queue is then enabled to continue receiving messages.

## Stopping the System Queue Archival Solution

To inactivate the solution, disable the Rules by performing one of the following actions:

- Disable the Rule on the Rule Processor Detail Control panel.
- Use the (D)isable line command on the Ruleset Overview panel.

- Issue the BBI commands:

```
. T RULE, DIS, MQARC001
```

```
. T RULE, DIS, MQARC002
```

```
. T RULE, DIS, MQARC003
```

```
. T RULE, DIS, MQARC004
```

```
. T RULE, DIS, MQARC005
```

```
. T RULE, DIS, MQARC006
```

---

## Using the Service Interval Performance Solution

Use the Service Interval Performance solution when you want to be informed automatically if a queue has not been receiving messages at its set rate. For example, a queue might be defined to receive (MQPUT) messages at a rate of once every five minutes. If a time interval of six minutes passes without the queue receiving any messages, a Q\_SERVICE\_INTERVAL\_HIGH event occurs when the next MQPUT is issued for the queue and a Rule fires, generating an ALERT to inform you of the situation.

Once the solution is enabled, the ALERT informs you that the time interval has been exceeded for the queue. When the time interval and interval rate return to normal, the solution automatically deletes the ALERT.

Rule ID	Triggering event
MQINT001	Fires when a Q_SERVICE_INTERVAL_HIGH event occurs.
MQINT002	Fires when a Q_SERVICE_INTERVAL_OK event occurs.

### Security Considerations

To use the Service Interval Performance solution, AutoOPERATOR needs access authority to MQSeries resources as follows:

- Read access to any queues specified in the AAOMQLxx member of BBPARM.

## Implementing the Service Interval Performance Solution

To invoke any part of the Service Interval Performance solution, first

1. Add the names of the queue managers which this solution will manage to BBPARM member AAOMQLxx
2. Issue the `.RESET MQ xx` command where `xx` is the two-character suffix of the AAOMQLxx member

### Enabling the Service Interval Performance Rules

To invoke the Service Interval Performance solution, edit and enable the Rules MQINT001 and MQINT002:

1. Invoke the Rule Processor application by typing `RULES` from any BBI terminal session screen.
2. Ensure that the AAORULBQ Rule Set is enabled. If not, enable the Rule Set by typing the `(E)nable` line command next to the Rule Set name.
3. Select the AAORULBQ Rule Set.
4. Edit MQINT001 using the Rules Processor application panels.

Press Enter to navigate through the panels and make the following changes:

- a. On the Selection Criteria panel under Queue Identification, change `XXXX` to the queue manager IDs of the queue managers that this Rule will manage. Remember that these

queue manager names must be specified in BBPARM member AAOMQLxx and then reset with the BBI command `. RESET MQ xx`.

- b. Make sure that the MQSeries queues that you want to monitor have their QSVICINT and QSVICIEV attributes set accordingly. Refer to the *IBM MQSeries Command Reference* for more information.

5. Repeat Step 4 for MQINT002.

6. Enable the MQINT001 and MQINT002 Rules by performing one of the following steps:

- Enable the Rule on the Rule Processor Detail Control panel.
- Use the (E)nable line command on the Ruleset Overview panel.
- Issue the BBI commands:

```
. T RULE, ENA, MQINT001
```

```
. T RULE, ENA, MQINT002
```

Once each Rule is enabled, when a Q\_SERVICE\_INTERVAL\_HIGH event occurs, the Rule fires and an ALERT is created. The ALERT states that the time interval has been exceeded for the queue.

When a Q\_SERVICE\_INTERVAL\_OK event occurs (signaling that the interval rate has been satisfied), another Rule fires and deletes the ALERT.

## Stopping the Service Interval Performance Solution

To inactivate the solution, disable Rules MQINT001 and MQINT002 by performing one of the following steps:

- Disable the Rule on the Rule Processor Detail Control panel.
- Use the (D)isable line command on the Ruleset Overview panel.
- Issue the BBI commands:

```
. T RULE, DIS, MQINT001
```

```
. T RULE, DIS, MQINT002
```

---

## Using the Queue Depth Management Solution

Use the Queue Depth Management solution when you want to be informed automatically of any user queue having a Q\_DEPTH\_HIGH event. A Q\_DEPTH\_HIGH event occurs when a queue has exceeded its depth threshold as defined by the QDEPTHHI attribute.

When a Q\_DEPTH\_HIGH event occurs, a Rule fires, generating an ALERT to inform you that the queue has reached its depth threshold. The ALERT text will indicate which queue manager the queue in question is running under.

The Queue Depth Management solution also uses a second Rule, which fires when a Q\_DEPTH\_LOW condition occurs. This second Rule will delete the ALERT generated by the first Rule.

Rule ID	Triggering event
MQQDP001	Fires when a Q_DEPTH_HIGH event occurs.
MQQDP002	Fires when a Q_DEPTH_LOW event occurs.

### Security Considerations

To use the Queue Depth Management solution, AutoOPERATOR needs access authority to MQSeries resources as follows:

- Alter and Read access to queues specified in the AAOMQLxx member of BBPARM that participate in this monitoring.

### Implementing the Queue Depth Management Solution

To invoke the Queue Depth Management solution, first

1. Add the names of the queue managers which this solution will manage to BBPARM member AAOMQLxx
2. Issue the `.RESET MQ xx` command where xx is the two-character suffix of the AAOMQLxx member

### Enabling the Queue Depth Management Rules

To invoke the Queue Depth Management solution, edit and enable the Rules MQQDP001 and MQQDP002:

1. Invoke the Rule Processor application by typing **RULES** from any BBI terminal session screen.
2. Ensure that the AAORULBQ Rule Set is enabled. If not, enable the Rule Set by typing the (E)nable line command next to the Rule Set name.
3. Select the AAORULBQ Rule Set.
4. Edit MQQDP001 using the Rules Processor application panels.

Press Enter to navigate through the panels and make the following changes:

- a. On the Selection Criteria panel under Queue Identification, change XXXX to the queue manager IDs of the queue managers that this Rule will manage. Remember that these queue manager names must be specified in BBPARM member AAOMQLxx and then reset with the BBI command `. RESET MQ xx`.
  - b. Make sure that the MQSeries queues that you want to monitor have their QDEPTHHI and QDEPTHLO attributes set accordingly. See the *IBM MQSeries Command Reference* for more details. For the solution to work properly, the percentage specified for QDEPTHLO should be one percent less than the percentage specified for QDEPTHHI.
5. Repeat Step 4 for MQQDP002.
  6. Enable the MQQDP001 and MQQDP002 Rules by performing one of the following:
    - Enable the Rule on the Rule Processor Detail Control panel.
    - Use the (E)nable line command on the Ruleset Overview panel.
    - Issue the BBI commands:
 

```
. T RULE, ENA, MQQDP001
. T RULE, ENA, MQQDP002
```

Once the Rule is enabled, when a Q\_SERVICE\_INTERVAL\_HIGH event occurs, the Rule fires and an ALERT is created stating that the service interval has been exceeded for the queue.

When a Q\_SERVICE\_INTERVAL\_OK event occurs (signaling that the interval rate has been satisfied), another Rule fires and deletes the ALERT.

## Stopping the Queue Depth Management Solution

To inactivate the solution, disable Rules MQQDP001 and MQQDP002 by performing one of the following steps:

- Disable the Rule on the Rule Processor Detail Control panel.
- Use the (D)isable line command on the Ruleset Overview panel.
- Issue the BBI commands:
 

```
. T RULE, DIS, MQQDP001
. T RULE, DIS, MQQDP002
```

---

## Using the Channel Availability Solution

The Channel Availability solution is a collection of Rules that fires when CHANNEL\_START and CHANNEL\_STOP instrumentation events are generated for LU6.2 and TCP/IP channels.

When a CHANNEL\_STOP event occurs, the Rules check if the event was due to one or more of the most common channel errors for MQSeries. In each case an ALERT is generated, indicating the type of stoppage, where it occurred and why. ALERT help panels accompany the ALERT text for all but the most obvious conditions. These panels describe the error in detail and prescribe steps to take to correct the channel outage.

When a CHANNEL\_START event occurs, Rules fire that ensure ALERTs previously issued for the channel just started are deleted from the ALERT queue. This helps to keep availability information timely and accurate.

Users can choose those outage conditions they wish to monitor. The table below describes the Rules included in the solution.

Rule ID	Default state	Remote command issued? Yes or No	Triggering Event
MQP2023D	Enabled	No	Connection ID or Remote Listener Unavailable - connection refused
MQTCK001	Disabled	No	Inspect TCP channel availability
MQP2023C	Enabled	No	Connection ID or Remote Listener Unavailable - connection timed out
MQP20832	Enabled	No	Error receiving data from connection ID - Network is down
MQP2083C	Enabled	No	Error receiving data from connection ID - Time out due to hardware failure
MQP20636	Enabled	No	Error sending data to connection ID - connection reset by peer
MQP54500	Disabled	Yes	Channel disconnect interval exceeded - AutoOPERATOR restarting channel
MQPRMT01	Enabled	No	Remote channel stopped by user command
MQP00001	Enabled	No	Host channel stopped by user command

**Note:** If a Rule issues remote commands, you must check the name of the command queue for queue managers on non-OS/390 platforms. If the command queue name is not SYSTEM.COMMAND.QUEUE, you must make a copy of the Rule. The command specified on the Rule Processor Automation Action panel must be modified to include a CQ (command queue) keyword naming the command input queue.

## Security Considerations

To use the Channel Availability solution, AutoOPERATOR needs access authority to MQSeries resources as follows:

- Control access to channels on which you want this solution to start.
- Authority to issue DISPLAY and START channel commands on distributed queue managers.

## Implementing the Channel Availability Solution

To invoke the Channel Availability solution, first

1. Add the names of the queue managers which this solution will manage to BBPARM member AAOMQLxx
2. Issue the . RESET MQ xx command where xx is the two-character suffix of the AAOMQLxx member

**Note:** The Channel Solution does not work for CICS channels.

## Enabling the Channel Availability Rule

To invoke the Channel Availability solution, edit and enable the Rules in Rule Set AAORULBH:

1. If you want to use Rule MQTCK001 to inspect TCP channel availability:
  - a. Copy member AAOMQMxx from BBPARM to the first data set in the subsystem BBPARM concatenation.
  - b. Rename it to have the same suffix as the AAOPRMxx member for this subsystem (for example, if the AAOPRMxx for this subsystem has a suffix of 12, the name of the AAOMQMxx member should be AAOMQM12).
  - c. Edit the AAOMQMxx member. Add a line for the name of each remote queue manager that uses TCP to communicate with the local queue manager. Type the name in column 1 of each line.
  - d. Alter the attributes of the TCP sender channels you want to monitor by typing
    - **SHORTRTY(1)**
    - **SHORTTMR(60)**
    - **LONGRTY(0)**
    - **LONGTMR(0)**

Refer to the *IBM MQSeries Command Reference* for the exact command syntax.

2. Invoke the Rule Processor application by typing **RULES** from any BBI terminal session screen.
3. Ensure that the AAORULBH Rule Set is enabled. If not, enable the Rule Set by typing the (E)nable line command next to the Rule Set name.
4. Select the AAORULBH Rule Set.
5. Edit each Rule using the Rules Processor application panels.

Press Enter to navigate through the panels and make the following changes:

- a. On the Selection Criteria panel under Queue Identification, change XXXX to the queue manager IDs of the queue managers that this Rule will manage. Remember that these

queue manager names must be specified in BBPARM member AAOMQLxx and then reset with the BBI command `. RESET MQ xx`.

- b. If you are using the Rule MQTCK001, go to the Action Specification panel. In the EXEC Name/Parms field replace:

`XXXX`

with the four-character name of the local queue manager

6. Enable the MQTCK001 Rule to monitor specific outage conditions, or any of the other Rules by performing one of the following actions:
  - Enable the Rule on the Rule Processor Detail Control panel
  - Use the `(E)nable` line command on the Ruleset Overview panel
  - Issue the BBI command:  
**`. T RULE, ENA, ruleID`**  
where `ruleID` is the Rule ID of the Rule you want to enable.
7. When you enable the Rule MQTCK001, the BBI-SS PAS must be restarted (either WARM or COLD), which allows the Rule to become automatically active at BBI-SS PAS initialization.

## Stopping the Channel Availability Solution

To inactivate the solution, disable the Rules by performing one of the following actions:

- Disable the Rule on the Rule Processor Detail Control panel.
- Use the `(D)isable` line command on the Ruleset Overview panel.
- Issue the BBI command:

**`. T RULE, DIS, ruleID`**

where `ruleID` is the Rule ID of the Rule you want to disable.

---

## Using the Basic Intercommunication Solution

The Basic Intercommunication solution maximizes the availability of connections between OS/390 and non-OS/390 queue managers by using event-driven automation.

**Note:** The Basic Intercommunication EXECs use REXX socket calls; therefore, they will not work on systems, which do not have IBM TCP/IP active. Interlink TCPAccess does not support REXX socket calls.

## Security Considerations

To use the Basic Intercommunication solution, AutoOPERATOR needs access authority to MQSeries resources as follows:

- Control access to any channels that you want this solution to start.
- Read and Alter authority for transmit queues that this solution will monitor.
- Authority to issue commands on distributed queue managers.

## Before You Begin

For this solution to work, the PATROL for MQ - Administrator product must be installed on each non-OS/390 target platform to which you are connecting. For more information about this product, refer to the BMC Software documents *PATROL for MQ- Administrator Installation Guide* and *PATROL for MQ- Administrator User Guide*.

The Basic Intercommunication solution is made up of AutoOPERATOR Rules driven by MQSeries events that are related to distributed queueing and EXECs that take action to configure and repair MQSeries connections.

Users have the ability to change the mode in which this solution operates from Diagnose (Diag) mode to Repair mode by issuing a BBI Journal message, which causes a shared variable to be set that tells the solution what mode it is operating in. This operating mode specifies whether EXECs scheduled by the Basic Intercommunication solution should try to repair discovered errors or just diagnose and report the errors for manual resolution. If the EXECs cannot repair queue manager connection errors, an ALERT is issued requesting that an operator manually repair the connection.

The table below describes the Rules from the Rule Set AAORULBR that are included in the Basic Intercommunication solution.

Table 15. Rules from the Rule Set AAORULBR

Rule ID	Default state	EXEC scheduled	Triggering event
MQCOR000	Enabled	QMQCORCF	<p>Time-initiated Rule used to define whether the Basic Intercommunication solution runs in Diag or Repair mode. This Rule executes at BBI-SS PAS start up. The default value is Diag mode.</p> <p>To change the mode to Repair:</p> <ul style="list-style-type: none"> <li>• Copy the Rule into another Rule and change Diag to Repair on the Action Specification panel. Disable the MQCOR000 Rule.</li> <li>• Issue the following message from the BBI Journal to cause a Rule to fire and change the mode to Repair for the duration of the SS or until you change it back:</li> </ul> <p><b>*Rule Set Var Mqmode Repair</b></p>
MQCORMOD	Disabled	QMQCORCF	Journal Rule that responds to the <b>*Rule Set Var Mqmode Diag/Repair</b> journal message and changes the mode in which the solution works.
MQCOR001	Enabled	QMQCORCF	Fires when a Channel_Stopped condition with a non-zero reason code is detected by AutoOPERATOR. QMQCORCF is invoked and attempts to verify that the channel is using the MQSeries triggering process. If the EXEC is running in Repair mode, QMQCORCF starts the channel manually.
MQCOR006	Enabled	QMQCORCF	Fires when an Unknown_Object_Name event is detected by AutoOPERATOR and the object queue manager is the same as the local queue manager. QMQCORCF notifies the operator of the condition through an ALERT.
MQCOR007	Enabled	QMQCORCF	Fires when a Put_Inhibited event is detected by AutoOPERATOR for the System.Command.Input queue. QMQCORCF is invoked and attempts to correct the situation if the EXEC is running in Repair mode.
MQCOR008	Enabled	QMQCORCF	Fires when a Get_Inhibited event is detected by AutoOPERATOR for the System.Command.Input queue. QMQCORCF is invoked and attempts to correct the situation if the EXEC is running in Repair mode.
MQCOR009	Enabled	QMQCORCF	Fires when an Xmit_Q_Usage_Error event is detected by AutoOPERATOR. QMQCORCF is invoked and attempts to repair the condition if the EXEC is running in Repair mode.

Table 15. Rules from the Rule Set AAORULBR (Continued)

<b>Rule ID</b>	<b>Default state</b>	<b>EXEC scheduled</b>	<b>Triggering event</b>
MQCOR010	Enabled	QMQCORCF	Fires when a Get_Inhibited event is detected by AutoOPERATOR and the queue for which the event occurred is a transmit queue that follows the naming convention expected by Command MQ On Ramp. QMQCORCF is invoked and attempts to repair the condition if the EXEC is running in Repair mode.
MQCOR011	Enabled	QMQCORCF	Fires when a Put_Inhibited event is detected by AutoOPERATOR and the queue for which the event occurred is a transmit queue that follows the naming convention expected by Command MQ On Ramp. QMQCORCF is invoked and attempts to repair the condition if the EXEC is running in Repair mode.
MQCOR012	Enabled	QMQCORCF	Fires when an Unknown_Remote_Q_Mgr event is detected by AutoOPERATOR. The Rule notifies the operator of the condition through an ALERT.
MQCOR013	Enabled	QMQCORCF	Fires when an Xmit_Q_Type_Error event is detected by AutoOPERATOR. QMQCORCF is invoked and attempts to repair the condition if the EXEC is running in Repair mode.
MQCOR014	Enabled	QMQCORCF	Fires when a Def_Xmit_Q_Usage_Error event is detected by AutoOPERATOR. QMQCORCF notifies the operator of the condition through an ALERT.
MQCOR015	Enabled	QMQCORCF	Fires when an Unknown_Def_Xmit_Q event is detected by AutoOPERATOR. QMQCORCF notifies the operator of the condition through an ALERT.
MQCOR016	Enabled	QMQCORCF	Fires when a Not_Authorized event is detected by AutoOPERATOR. QMQCORCF notifies the operator of the condition through an ALERT.
MQCOR017	Enabled	QMQCORCF	Fires when a Put_Inhibited event is detected by AutoOPERATOR and the queue for which the error occurred is a remote command queue. QMQCORCF is invoked and attempts to repair the condition if the EXEC is running in Repair mode.
MQCOR018	Enabled	QMQCORCF	Fires when a Q_Mgr_Not_Active event is detected by AutoOPERATOR. QMQCORCF is invoked and attempts to repair the condition if the EXEC is running in Repair mode or notifies the operator of the condition through an ALERT.
MQCOR019	Enabled	QMQCORCF	Fires when a Q_Full event is detected by AutoOPERATOR and the queue for which the error occurred is a transmit queue. QMQCORCF is invoked and attempts to repair the condition if the EXEC is running in Repair mode.

## Invoking the Basic Intercommunication Solution

To invoke the Basic Intercommunication solution:

1. Add the names of the OS/390 queue managers that this solution will manage to BBPARM member AAOMQLxx.
2. Issue the .RESET MQ xx command where xx is the two-character suffix of the AAOMQLxx member

**Note:** The Rule that is triggered by CHANNEL\_STOPPED events does not work for CICS channels.

## Enabling the Basic Intercommunication Solution

To enable the Basic Intercommunication solution, verify that the Rules in the Rule Set AAORULBR and the Rule Set itself are enabled. By default, each Rule in the Rule Set is already enabled except MQCORMOD.

To verify the Rule Set is enabled:

1. Invoke the Rule Processor application by typing RULES from any BBI terminal session screen.
2. Ensure that the AAORULBR Rule Set is enabled. If not, enable the Rule Set by entering the (E)nable line command next to the Rule Set name.

To verify the Rules are enabled:

1. Select the AAORULBR Rule Set.
2. Verify the Rules are enabled.
3. Press Enter to navigate through the panels and make the following changes:  
On the Selection Criteria panel under Manager(s), change XXXX to the queue manager IDs of the OS/390 queue managers that each Rule will manage.

To enable MQCORMOD:

1. Select the AAORULBR Rule Set.
2. Edit the Rule using the Rules Processor application panels.

To change the operating mode of the solution:

- Issue the **\*Rule Set Var Mmode Diag** journal message if you want the solution to try to diagnose the connection errors.
- Issue the **\*Rule Set Var Mmode Repair** journal message if you want the solution to try to diagnose and repair the errors.

## Stopping the Basic Intercommunication Solution

To stop the Basic Intercommunication solution, disable the Rules by performing the following action:

Disable the Rule Set in the Automation Control panel.

Or do one of the following actions to disable each of the Rules individually:

- Disable the Rule on the Rule Processor Detail Control panel.
- Enter the (D)isable line command on the Ruleset Overview panel.
- Issue the BBI command:

```
. T RULE, DIS, rul e ID
```

where `rul eID` is the name of the Rule that you want to disable.



---

## Chapter 7. Command MQ Automation Power Line (APL)

Command MQ Automation Power Line (APL) provides the capability to issue commands from an EXEC to manage non-OS/390 MQSeries queue managers and receive responses to those commands without using MQSeries Channels. APL can only be invoked from an EXEC.

The APL EXEC requires

- PATROL Node Manager running on a non-OS/390 node
- At least one BBI-SS PAS containing AutoOPERATOR 6.2
- TCP/IP level 3.1 or higher running on OS/390

APL must have authority for MQSeries on the computer with which it is communicating. For information about PATROL Node Manager, refer to *PATROL for MQ - Administrator User Guide*.

The following statement shows the format of the APL EXEC:

```
IMFEXEC SELECT E(QMQPOWER CMD(Commands) NODE(IPname)
PORT(port) RM(qmgr) WAIT(sec) DEBUG(debug) HELP(help) RESP(response))
WAIT(YES)
```

**Note:** The APL EXEC uses REXX socket calls; therefore, it will not work on systems, which do not have IBM TCP/IP active. Interlink TCPAccess does not support REXX socket calls.

Table 16 describes the EXEC's parameters.

Table 16. APL parameters

Parameter	Required?	Function	Notes
CMD	YES	Valid PATROL Node Manager command	<p>This parameter has no default value.</p> <p>The PATROL Node Manager command can be up to 256 characters in length.</p> <p>You must replace all blanks with plus signs.</p> <p>If MQSeries commands use lowercase or mixed case MQSeries object name, the name should be enclosed in single quotation marks. To pass a single-quote enclosed character string, it must also be enclosed in single quotation marks. That is, the quotation marks must be typed twice, for example:</p> <pre>'CMD(DIS+CHL("CSQ1.TO.remote")+ALL)'</pre> <p>PATROL Node Manager accepts any valid MQSeries command and any of the following PATROL Node Manager "EXEC" commands:</p> <pre>EXCMD_START_Q_MGR EXCMD_STOP_Q_MGR EXCMD_INQUIRE_Q_MGR_NAMES</pre> <p>Multiple MQSeries or Node Manager "EXEC" commands can be passed in one request. The commands must be separated by two consecutive colons (::).</p>
NODE	YES	PATROL Node Manager network node name or IP address	This parameter has no default value.
PORT	NO	PATROL Node Manager port number on its node	<p>If omitted, this parameter uses the 5000 default value.</p> <p>Valid values are 0-9999.</p>
RM	NO	Non-MVS queue manager with which APL communicates	If omitted, this parameter uses the default value equal to Userid.

Table 16. APL parameters (Continued)

<b>Parameter</b>	<b>Required?</b>	<b>Function</b>	<b>Notes</b>
WAIT	NO	Number of seconds to wait for the PATROL Node Manager response. This time applies to every TCP/IP send or response communication.	If omitted, this parameter uses a 60-second default value.
DEBUG	NO	To write debug information to the journal (YES, NO)	If omitted, this parameter uses NO as the default.
RESPonse	NO	Response format	Can have one of three values:  NO: Responses for commands will not be saved.  YES: (default) All responses will be saved in APLNxx variables, contiguously without separation lines between them.  DELIM: Each response begins and ends with a separation line.
HELP	NO	To write APL's call format to the journal (YES, NO)	If omitted, this parameter uses NO as the default.  If HELP(YES) is provided, all other parameters are ignored. Calling QMQPOWER without parameters is the same as calling it with HELP(YES).
For more information about MQSeries, refer to the <i>IBM MQSeries Application Programming Reference</i> .			

## Variables Returned from APL

The IMFCC TSO variable describes the success or failure of the EXEC that calls APL. The IMFRC TSO variable contains information about the success of APL. LOCAL variables APLCC and APLRC are also returned (and they contain additional information). For more information, refer to Table 17.

Table 17. APL Variables IMFRC, APLCC, and APLRC

Message in APLLNxx	IMFRC	APLCC	APLRC
PL0000I SUCCESS or MQSeries response	0	0	0
PATROL Node Manager Return code	8	4	PATROL Node Manager reason code
SOCKET(SOCKET) rc=nn SOCKET(CONNECT) rc=nn SOCKET(WRITE) rc=nn SOCKET(SELECT) rc=nn	8	8	TCP/IP return code
Error in EXCMD command	8	12	1
Unable to initialize SOCKET	8	12	3
Connection reset by peer	8	12	4
TIMED OUT	8	12	5
Unrecognized response from MQ agent	8	12	6
CMD parameter is not provided	8	16	51
NODE parameter is not provided	8	16	52
Debug parameter can be only YES or NO	8	16	53
Required A0 for MQSeries is not active	8	16	54
A0 for MQSeries services required not available on this level. Must be 5.1 or higher	8	16	55
Error in parameter structure	8	16	56
Unknown parameter	8	16	57
Error in RESPONSE parameter	8	16	58

**Note:** If more than one command is passed, IMFRC, APLCC and APLRC correspond to the highest value of APLCC in the set of returns or the highest level of APLRC if APLCC=0.

For more information on TCP/IP return codes, refer to the *IBM TCP/IP for MVS Application Programming Interface Reference*, SC31-7187-03.

For more information on MQSeries return codes, refer to *MQSeries for MVS/ESA Message and Codes*, GC33-0819-03.

Additional response information from the queue manager is returned in the LOCAL variables APLNOL and APLL1 through APLLxx, where xx is the last line of returned information. APLNOL contains the number of returned lines (which is equal to the value of xx). These variables must be retrieved using the IMFEXEC VGET command before they can be used.

Variable Name	Definition
APLNOL	Contains the number of returned lines in variables APLL1 through APLLxx
APLL1	Contains the first line of response information from the queue manager
APLLxx	Contains the last line of response information from the queue manager

**Note:** If more than one command is passed:

- APLNOL contains the total number of the response lines from all responses together.
- APLLxx contains the response information from all commands. Each command response begins from a new line.

Figure 41 shows that the variable APLNOL has a value of 13, which equals 13 lines of returned information from the queue manager.

---

```

APLNOL = 13
AMQ8408: Display queue manager details
DESCR( ) DEADQ(DEAD. LETTER.)
DEFXMTQ CHADEXIT( )
COMMANDQ(SYSTEM ADMIN. COMMAND. QUEUE) QMNAME(EPESIN)
TRIGINT(999999999) MAXHANDS(256)
MAXUMSGS(10000) AUTHOREV(DISABLED)
INHIBTEV(ENABLED) LOCALEV(ENABLED)
REMOTEEV(ENABLED) PERFMEV(ENABLED)
STRSTPEV(ENABLED) CHAD(ENABLED)
CHADEV(DISABLED) MAXMSGL(4194304)
MAXPRTY(9) CCSID(437)
CMDLEVEL(500) DI STL(YES)
SYNCPT
  
```

Figure 41. Example of Information Returned in Variables APLNOL and APLL1 through APLL13

---

## Examples

The following examples show the majority of the functions of the APL EXEC.

### Example 1– Displaying Attributes of Queue Manager

---

```
IMFEXEC SELECT EXEC(QMQPOWER NODE(137.72.4.17) RM(EPESIN) ,  
CMD(DI S+QMGR+ALL) PORT(5006) DEBUG(YES) ) WAIT(YES)
```

---

This request displays all the attributes of queue manager EPESIN.

**Note:** This example uses an MQSeries command; when MQSeries commands are used, a + sign is used to separate the command and its parameters.

### Example 2 – Requesting Names of Queue Managers

---

```
IMFEXEC SELECT EXEC(QMQPOWER NODE(137.72.4.17) DEBUG(YES) ,  
CMD(EXCMD_INQUIRE_Q_MGR_NAMES) ) WAIT(YES)
```

---

This command requests the names of the queue managers defined in the Windows NT system by the address of 137.72.4.17. The EXEC returns their names in one of the local variables APLLN1 - APLLNnn, depending on the number of queue managers.

### Example 3 – Start Queue Manager

---

```
IMFEXEC SELECT EXEC(QMQPOWER NODE(137.72.4.17) RM(EPESIN) ,  
CMD(EXCMD_START_Q_MGR) PORT(5005) ) WAIT(YES)
```

---

This request starts queue manager EPESIN.

### Example 4 – Stop Queue Manager

---

```
IMFEXEC SELECT EXEC(QMQPOWER  
CMD(EXCMD_STOP_Q_MGR) PORT(5006) NODE(137.72.4.17) RM(EPESIN) ) WAIT(YES)
```

---

This request stops queue manager EPESIN.

## Example 5 – Passing Multiple Commands

### Example 5a

```
"IMFEXEC SELECT EXEC(QMQPOWER CMD(EXCMD_INQUIRE_Q_MGR_NAMES: : DIS" || ,
"+QMGR+ALL: : DIS+Q(BMC. LISTENER. SUB)+CURDEPTH)" || ,
"NODE(172. 20. 40. 243) RM(EPESIN) PORT(5000)" || ,
"DEBUG(YES) WAIT(180)) WAIT(YES)"
```

#### Response:

```
APLCC = 00000
APLRC = 00000
IMFCC = 0000
IMFRC = 0000
APLNOL = 22
cortest
EPESIN
AMQ8408: Display Queue Manager details.
DESCR( ) DEADQ(SYSTEM DEAD. LETTER. QUEUE)
DEFXMITQ( ) CHADEXIT( )
CLWLEXIT( ) CLWLDATA( )
REPOS( ) REPOSNL( )
COMMANDQ(SYSTEM ADMIN. COMMAND. QUEUE) QMNAME(EPESIN)
CRDATE(1999-07-08) CRTIME(16.00.52)
ALTDATE(1999-11-11) ALTTIME(13.26.35)
QMID(EPESIN_1999-07-08_16.00.52) TRIGINT(999999999)
MAXHANDS(256) MAXUMSGS(10000)
AUTHOREV(ENABLED) INHIBTEV(ENABLED)
LOCALEV(ENABLED) REMOTEEV(ENABLED)
PERFMEV(ENABLED) STRSTPEV(ENABLED)
CHAD(DISABLED) CHADEV(ENABLED)
CLWLEN(100) MAXMSGL(4194304)
CCSID(437) MAXPRTY(9)
CMDLEVEL(520) PLATFORM(WINDOWSNT)
SYNCPT DI STL(YES)
AMQ8409: Display Queue details.
QUEUE(BMC. LISTENER. SUB) CURDEPTH(1)
```

### Example 5b

```
"IMFEXEC SELECT EXEC(QMQPOWERCMD("EXCMD_INQUIRE_Q_MGR_NAMES: : " || ,
"DIS+Q(SYSTEM ADMIN. COMMAND. QUEUE)+ALL: : " || ,
"EXCMD_START_Q_MGR: : DIS+QMGR+ALL: : " || ,
"DIS+Q(BBOMVAO. YXP. NT)+CURDEPTH)" || ,
"NODE(172. 20. 40. 243) RM(EPESIN) PORT(5000)" || ,
"RESP(DELM) DEBUG(YES) WAIT(180)) WAIT(YES)"
```

#### Response:

```
APLCC = 0
APLRC = 0
IMFCC = 0000
IMFRC = IMFRC
APLNOL = 57
PL9000I CMD 1 2 EXCMD_INQUIRE_Q_MGR_NAMES
cortest
EPESIN
PL9001I CMD 1 2 EXCMD_INQUIRE_Q_MGR_NAMES
PL9000I CMD 2 24 DIS Q(SYSTEM ADMIN. COMMAND. QUEUE) ALL
AMQ8409: Display Queue details.
DESCR(MQSeries administration command queue)
PROCESS( ) BOQNAME( )
INITQ( ) TRIGDATA( )
```

```

CLUSTER( )
QUEUE(SYSTEM ADMIN. COMMAND. QUEUE)
CRTIME(16. 01. 00)
ALTTIME(16. 01. 00)
PUT(ENABLED)
DEFPSIST(NO)
MAXMSGL(9000)
NOSHARE
HARDENBO
RETINTVL(999999999)
NOTRIGGER
TRIGPTH(1)
QDEPTHHI(80)
QDPMAXEV(ENABLED)
QDPLOEV(DISABLED)
QSVCEV(NONE)
DEFTYPE(PREDEFINED)
SCOPE(QMGR)
IPPROCS(1)
CURDEPTH(0)
PL9001I CMD 2 24 DIS Q(SYSTEM ADMIN. COMMAND. QUEUE) ALL
PL9000I CMD 3 1 EXCMD_START_Q_MGR
SUCCESS
PL9001I CMD 3 1 EXCMD_START_Q_MGR
PL9000I CMD 4 18 DIS QMGR ALL
AMQ8408: Display Queue Manager details.
DESCR( ) DEADQ(DEAD. LETTER. QUEUE)
DEFXMITQ( ) CHAEXIT( )
CLWLEXIT( ) CLWLDATA( )
REPOS( ) REPOSNL( )
COMMANDQ(SYSTEM ADMIN. COMMAND. QUEUE) QMNAME(EPESIN)
CRDATE(1999-07-08) CRTIME(16. 00. 52)
ALTDATE(1999-11-11) ALTTIME(13. 26. 35)
QMID(EPESIN_1999-07-08_16. 00. 52) TRIGINT(999999999)
MAXHANDS(256) MAXUMSGS(10000)
AUTHOREV(ENABLED) INHIBTEV(ENABLED)
LOCALEV(ENABLED) REMOTEEV(ENABLED)
PERFMEV(ENABLED) STRSTPEV(ENABLED)
CHAD(DISABLED) CHADEV(ENABLED)
CLWLEN(100) MAXMSGL(4194304)
CCSID(437) MAXPRTY(9)
CMDLEVEL(520) PLATFORM(WINDOWSNT)
SYNCPT DI STL(YES)
PL9001I CMD 4 18 DIS QMGR ALL
PL9000I CMD 5 2 DIS Q(BBOMVAO. YXP. NT) CURDEPTH
AMQ8409: Display Queue details.
QUEUE(BBOMVAO. YXP. NT) CURDEPTH(17)
PL9001I CMD 5 2 DIS Q(BBOMVAO. YXP. NT) CURDEPTH

```

---

## Chapter 8. Securing AutoOPERATOR for MQSeries

The following resources for MQSeries for MVS/ESA can be secured:

- Connections to MQSeries
- Access to MQSeries objects (such as queues)
- Use of MQSeries system commands

This security is implemented from within the MQSeries product. Therefore, when you want to use the AutoOPERATOR for MQSeries component for automating MQSeries events, you must ensure AutoOPERATOR for MQSeries is granted the proper authority to access queue managers, system queues, and user queues as required.

AutoOPERATOR for MQSeries does allow you to specify who has access to updating and reading Rules and scheduling EXECs. If you need to secure who has access to updating and reading Rules and EXECs for MQSeries events, you must secure who has access to these resources. Refer to the BMC Software manual *Implementing Security for MAINVIEW Products* for more information about these security issues.

AutoOPERATOR for MQSeries provides a new resource name:

prefix.ssid.AAO.target.APPL.MQSERIES

Use this resource name to secure who has access to the AutoOPERATOR for MQSeries Workstation (described in Chapter 5, “Viewing AutoOPERATOR for MQSeries Automation Statistics” on page 115).

For information about how to secure access to this resource, refer to the *Implementing Security for MAINVIEW Products* book and read about “Securing Resources for AutoOPERATOR: Advanced Security” and “Securing AutoOPERATOR Applications: Feature=APPL”.



---

## Chapter 9. Using MAINVIEW AutoOPERATOR for MQSeries IMFEXEC Statements

This chapter contains the following discussions:

A description of various IMFEXEC MQI command statements that you can use in EXECs to interface with MQSeries objects.

A description of IMFEXEC MQI variables.

A description of the IMFEXEC CMD TYPE(MQS) statement that you can use to issue commands to the MQI command server.

A description of the IMFEXEC COPY MQI and IMFEXEC DISPLAY MQI command statements that you can use to copy or display MQI variable values.

For additional information, refer to the publication, *IBM MQSeries Application Programming Reference Guide*.

For information about other IMFEXEC command statements, refer to the *MAINVIEW AutoOPERATOR Advanced Automation Guide for CLIST EXECs* or *MAINVIEW AutoOPERATOR Advanced Automation Guide for REXX EXECs*.

---

## MQI Commands

Using EXECs, you can use the MAINVIEW AutoOPERATOR for MQSeries EXEC interface to communicate with MQSeries objects. Table 18 briefly describes the IMFEXEC command statements and where you can find additional information.

Table 18. IMFEXEC Command Statements

Command	Function	Refer to
IMFEXEC MQI BACK	Back out changes since the last syncpoint	“MQI BACK” on page 163
IMFEXEC MQI CLOSE	Release access to an object	“MQI CLOSE” on page 164
IMFEXEC MQI CMIT	Commit all changes since last syncpoint	“MQI CMIT” on page 167
IMFEXEC MQI CONN	Connect to a queue manager	“MQI CONN” on page 169
IMFEXEC MQI DISC	Disconnect from a queue manager	“MQI DISC” on page 171
IMFEXEC MQI GET	Get a message from a local queue	“MQI GET” on page 173
IMFEXEC MQI OPEN	Establish access to an object	“MQI OPEN” on page 183
IMFEXEC MQI PUT	Put one or more messages into a queue	“MQI PUT” on page 187
IMFEXEC MQI PUT1	Put one message in a queue	“MQI PUT1” on page 197

In addition to the MQI commands, other function are available. They are COPY MQI and DISPLAY MQI. For detailed information about these commands, see “COPY MQI” on page 214 and “DISPLAY MQI” on page 217.

## IMFEXEC MQI Variables

The following list describes general properties for the IMFEXEC MQI variables:

- Variable names consist of a
  - Prefix name (for example, IMFMQI)
  - Structure name (for example, MD, DLH)
  - Field name (for example, MsgId or CorrelId in the message descriptor)
- Each node of the variable name is delimited by an underscore (\_)
- Field-name variables whose corresponding possible values are constants, defined for them by MQSeries, are populated with the constant name rather than the actual value. For example, in the case of a Dead Letter message, the variable IMFMQI\_MD\_FORMAT, which contains the format field from the message descriptor, is set to MQFMT\_DEAD\_LETTER\_HEADER. Likewise, the variable IMFMQI\_MD\_VERSION, which contains the version of the message descriptor, is set to MQMD\_VERSION\_1 instead of x'1'. The following rules apply for variables set by AutoOPERATOR for MQSeries:
  - If the value for a character field does not have a defined constant value, the actual value is used.

- If the value for a numeric field that has a defined range does not have a defined constant, the actual numeric value converted to zoned decimal is used.
- Variables for fields that are defined as hexadecimal fields (MQBYTExx in MQSeries) are not converted and are set 'as is.'
- Variables that are created for flag fields in an MQSeries structure may contain multiple constant values, delimited by blanks. For example, the variable for the message descriptor Report field, IMFMQI\_MD\_REPORT, can have several values. The resulting value may look like this: 'MQRO\_COA MQRO\_COD'. Any other valid combination could be seen as well.
- Variables that are created for MQSeries fields where the value has more than one defined constant will contain both constants. For example, the variable IMFMQI\_MD\_PUTAPPLTYPE, which contains the PutApplType value from the message descriptor, would contain the value, 'MQAT\_OS390 MQAT\_MVS', if the message originated on an OS/390 system. This result is because both constants contain the value '2'.

## What the AutoOPERATOR MQI EXEC Interface Is

The AutoOPERATOR MQI EXEC interface consists of the following features:

- Commands through which applications can access the queue manager and its facilities
- Commands and variable structures that applications use to pass data to and get data from the queue manager
- Elementary data types for passing data to, and getting data from, the queue manager

The MQI commands in the AutoOPERATOR MQI EXEC interface correspond to MQSeries MQI functions.

## How Applications Communicate with MQSeries Objects

For an EXEC to communicate with a queue manager, the application must have a unique identifier by which it knows that queue manager. This identifier is called a *connection handle*. The connection handle is returned by the IMFEXEC MQI CONN statement when an application connects to the queue manager. This connection handle is used as an input, explicitly or implicitly, when an application executes subsequent MQI statements.

## How Applications Interact with MQSeries Objects

For an application to work with an MQSeries object (for example: a queue, a name list, a queue manager, or a process definition), both the application and the queue manager must have a unique identifier by which it knows that object. This identifier is called an *object handle*. The handle is returned by the IMFEXEC MQI OPEN statement when an application opens the object to work with it. Applications pass the object handle as input, explicitly or implicitly, when they use subsequent IMFEXEC MQI PUT, GET, or CLOSE statements.

## How Completion and Reason Codes Are Returned

A completion code and reason code are returned as output by each statement. The completion code is stored in the IMFCC and IMFMQCC variables. The variables usually contain a 0 or 2 in IMFMQCC and a 0 or 8 in IMFCC, showing success and failure, respectively. Some IMFEXEC MQI statements can return an intermediate state, a 1 in IMFMQCC and a 4 in IMFCC, indicating partial success.

The reason code is stored in the IMFRC and IMFMQRC variables. The contents of both variables are the same. The reason code shows the reason for the failure, or partial success. There are many reason codes covering circumstances such as a full queue or GET operations not allowed for a queue. Applications can analyze the reason code and based on the returned value, take additional actions.

The variable IMFMQI\_REASON contains the constant (character string) reason code that corresponds with the numeric reason code returned by MQSeries when an error is encountered. For example: when MQSeries returns the reason code 2195, the variable IMFMQI\_REASON contains 'MQRC\_UNEXPECTED\_ERROR'. The numeric reason code is displayed on the EM920II message, which is sent to the BBI journal when there is an error returned by MQSeries.

The completion codes for each statement with the description of the statement are listed in Table 19.

Table 19. MQI EXEC Completion Codes

IMFMQCC	IMFCC	Reason	System Action	User Action
0	0	Successful completion	None	None
2	8	MQSeries returned error	Command fails	You have 2 options:  1. Obtain the numeric reason code from variable IMFMQRC and refer to the <i>IBM MQSeries Application Programming Reference Guide</i> for reason information.  2. Obtain the character constant reason code from variable IMFMQI_REASON and take appropriate action. The value of this variable corresponds to the numeric value of IMFMQRC. For example, if IMFMQRC = 2033, then IMFMQI_REASON = 'MQRC_NO_MSG_AVAILABLE'.
	12	Buffer too small	Message is not PUT to the queue and command fails.	Specify a buffer large enough to hold the entire message and all included structures. See the topic "Creating Messages from Variables" for more information.
4	16	Syntax error	Command fails	Refer to EM0022E message in BBI journal for error description.
	20	Variable Processing routine not found	Command fails; for COPY MQI and DISPLAY MQI commands only	Look for errors during BBI PAS startup, related to program QAVARS, correct if possible and restart the BBI PAS. Otherwise contact BMC Software customer support.

The reason code is returned in the IMFMQRC and IMFRC variables; contents of both variables are the same. For the reason code descriptions, refer to the "Message Queuing Constants" chapter of the *IBM MQSeries Application Programming Reference Guide*.

## Creating Messages from Variables

You can use AutoOPERATOR for MQSeries variables and IMFEXEC MQI statements to modify existing MQSeries messages. You can also use AutoOPERATOR for MQSeries to create new MQSeries messages. This section describes how you can accomplish these tasks.

The IBM MQSeries product offers a programming interface for many languages such as assembler, COBOL and C. BMC Software has created the IMFEXEC MQI interface for AutoOPERATOR EXECs to provide access to these facilities using the REXX and CLIST

programming languages. The structures associated with the IBM MQSeries API (application programming interface) are available to you through AutoOPERATOR for MQSeries variables in the format, `IMFMQI_structure_fieldname`, where `structure` is the structure identifier and `fieldname` is the name of the field in the structure. See the section “IMFEXEC MQI Variables” on page 154 for more information about the format and content of these variables.

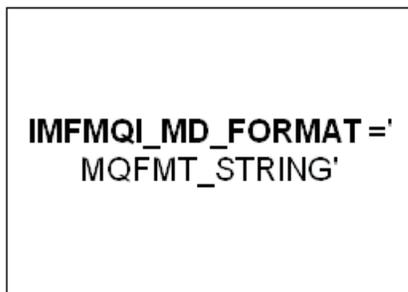
AutoOPERATOR can update messages retrieved from a queue, based upon values of variables that you set in your EXEC and then place the updated message on the same or a different queue when the IMFEXEC MQI PUT or PUT1 statement is executed. Some variables correspond to each field of each structure in MQSeries (CIH, DLH, IIH, MD, MDE, RFH, RFH2, RMH, TM, TMC2, WIH, XQH, CFH, OD). Additionally, other variables correspond to each data item name for an instrumentation event. Details about the most commonly used variables for the IMFEXEC MQI statements are covered in this chapter; all other variables are listed in Appendix D, “EXECs Variables” in this book.

When creating new messages, AutoOPERATOR builds the Message Descriptor, the Object Descriptor, and the message buffer from variables that you create, or from default values where possible (there are no default values for the message buffer). Note that when creating new messages, the author of the EXEC is responsible for entering all necessary variables for structures except for the Message Descriptor and Object Descriptor (MD and OD).

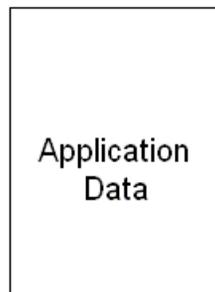
## Creating a Simple Message

AutoOPERATOR for MQSeries creates variables from messages it reads from queues when executing the IMFEXEC MQI GET statement. Likewise, new messages created by an EXEC, for the purpose of being PUT or PUT1 to a queue, are built from the contents of variables you create in the EXEC. An example of a message with a Message Descriptor and no additional structures is pictured below.

### Message Descriptor



### Message Buffer



## Creating a Complex Message

When creating new messages, you can add MQSeries structures to the messages by changing the value of the message descriptor format variable, `IMFMQI_MD_FORMAT`, and creating the necessary structure variables. A message with a message descriptor and an added structure is shown below:

### Message Descriptor

```
IMFMQI_MD_FORMAT =  
'MQFMT_CICS'
```

### Message Buffer

```
CIH (CICS  
Information Header)  
IMFMQI_CIH_FORMAT  
='MQFMT_STRING'
```

## Updating an Existing Message

You can update existing messages (ones which have been retrieved from a queue by the `IMFEXEC MQI GET` statement) by updating the variables created during the `GET`, and `PUT`ting the message to the same or a different queue. Additionally, a new structure can be added to an existing message. If you decide to add a structure to a message, the message buffer provided must be large enough to hold the new structure and a position in the message buffer must be available to write the data. When you add a structure (`DLH`, `CIH`, `WIH`, etc.), the following steps must be taken:

1. Update the variable `IMFMQI_MD_FORMAT` to specify the format of the structure you are adding, for example: `MQFMT_DEAD_LETTER_HEADER` or `MQFMT_WORK_INFO_HEADER`.
2. Create the appropriate structure variables. You must create all the variables in the structure because `AutoOPERATOR` will only update the message buffer for each field's position if the corresponding variable is set.
3. Create a new message buffer that is large enough to hold the new structure you are adding and the remaining current message buffer. For example, suppose you are adding the `WIH` (Work Information Header) structure that is 120 bytes and the current message buffer is called `IMFMQI_BUFFER`. You could create a variable that consists of the `WIH` structure ID ('`WIH` ') followed by 116 blanks (call it `WIH`, for example). You would issue the following REXX instruction in your `EXEC` to create the new message buffer:  
`My_New_Buffer = WIH||IMFMQI_BUFFER.`
4. Issue the `IMFEXEC MQI PUT` or `PUT1` statement (assuming you have connected to the queue manager and opened the destination queue).

**Note:** If the buffer specified is not large enough, a error code of 12 will be returned to the `EXEC` in the variable `IMFCC` and the `PUT` or `PUT1` statement will not be completed.

## Adding a Structure to an Existing Message

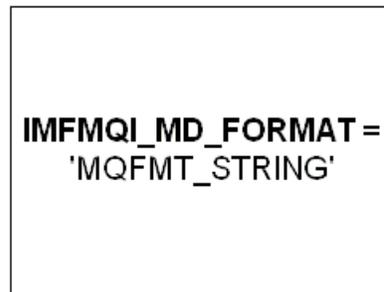
Adding a new structure to an existing message is more difficult than updating an existing message since the design of AutoOPERATOR for MQSeries, in this area, is more suited for updating existing structures. With existing messages, AutoOPERATOR is dealing with a message buffer that already contains the structures in place. However, to add new structures, a new message buffer must be created.

In this example, you want to GET a message from a queue, add a Dead Letter Header to it and PUT it onto the dead letter queue. After issuing the IMFEXEC MQI GET statement, the message buffer (IMFMQI\_BUFFER) by default, contains the application data, as shown below:

### Message Buffer



### Message Descriptor



Then, your EXEC sets Message Descriptor (MD), Dead Letter Header (DLH) and buffer variables as follows:

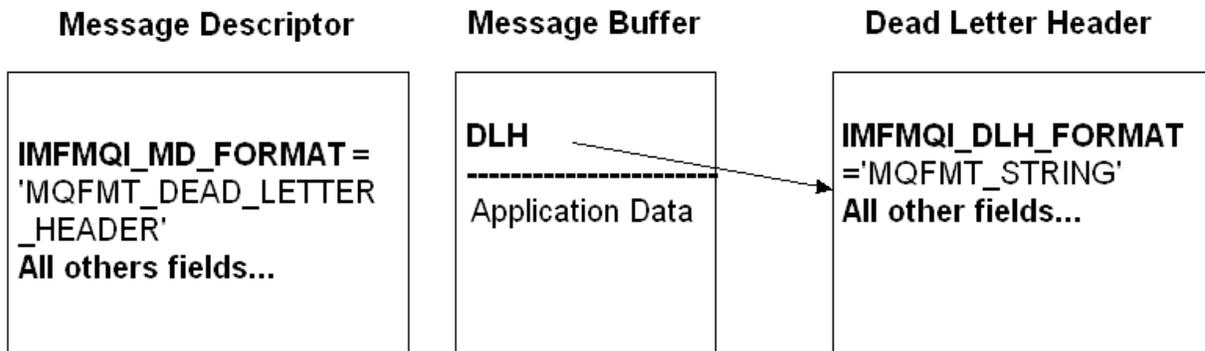
---

```
IMFMQI_MD_FORMAT = 'MQFMT_DEAD_LETTER_HEADER'  
IMFMQI_DLH_STRUCID = 'MQDLH_STRUC_ID'  
IMFMQI_DLH_VERSION = 'MQDLH_VERSION_1'  
IMFMQI_DLH_FORMAT = 'MQFMT_STRING'  
IMFMQI_DLH_REASON = '915'  
IMFMQI_DLH_DESTQNAME = 'MY.QUEUE1'  
IMFMQI_DLH_DESTQMGRNAME = 'MYREMOTE'  
IMFMQI_DLH_ENCODING = 'MQENC_NATIVE'  
IMFMQI_DLH_CODEDCHARSETID = '500'  
IMFMQI_DLH_PUTAPPLTYPE = 'MQAT_OS390'  
IMFMQI_DLH_PUTAPPLNAME = 'MYAPPL'  
IMFMQI_MD_PUTDATE = date('s')  
t = substr(translate(time('L'),'',':.'),1,11)  
IMFMQI_MD_PUTTIME = substr(t,1,2)||substr(t,4,2)||substr(t,7,2)||substr(t,10,2)  
My_New_Buffer = DLH||IMFMQI_BUFFER./* DLH = 'DLH..blanks...'*/
```

---

To use the new message buffer, issue IMFEXEC MQI PUT or PUT1 using the parameters, BUFFER(My\_New\_Buffer) and BUFLLEN(length(My\_New\_Buffer)).

During processing of the IMFEXEC MQI PUT or PUT1 statement, the Message Descriptor, Message Buffer and Dead Letter Header will be updated as follows:



**Note:** When processing messages, be aware that

- To add structures to your message, the message buffer must be large enough to hold the structure and the contents of the buffer. If it is not large enough, an error code 12 is returned to the EXEC in the variable IMFCC.
- You can chain multiple structures together by updating the Format field of the previous structure, as long as the message buffer is the correct size and the space that will contain the structure is empty.
- You cannot chain multiple structures together when a CFH structure is used.
- You cannot create instrumentation events with an EXEC.

## Adding an IMS bridge message

In this example, an IMS bridge message is built. The necessary object descriptor variables and all the IIH structure variables are set as follows:

---

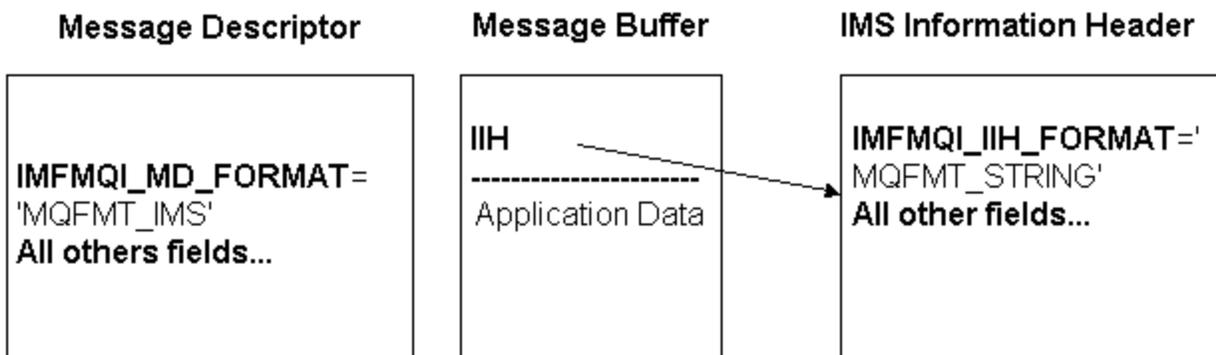
```

IMFMQI_OD_OBJECTNAME = QUEUE
IMFMQI_OO_OPTIONS = 'MQOO_OUTPUT'
IMFMQI_MD_REPLYTOQ = "BBOMVAO.YXP.QUEUE1"
IMFMQI_MD_REPLYTOQMGR = 1m
IMFMQI_MD_PERSISTENCE = "MQPER_PERSISTENT"
IMFMQI_MD_FORMAT = "MQFMT_IMS"
IMFMQI_PMO_TIMEOUT = -1
/* Variable Name value namevalue length*/
IMFMQI_IIH_STRUCID= "MQIIH_STRUC_ID"/* "IIH" 4*/
IMFMQI_IIH_VERSION= "MQIIH_VERSION_1"/* "0000001"X 4*/
IMFMQI_IIH_STRUCLength= "MQIIH_LENGTH_1"/* "00000054"X 4*/
IMFMQI_IIH_ENCODING= "MQENC_NATIVE"/* 785 4*/
IMFMQI_IIH_CODECHARSETID = "MQCCSI_Q_MGR"/* "00000000"X 4*/
IMFMQI_IIH_FORMAT= "MQFMT_IMS_VAR_STRING"/* "MQIMSVS " 8*/
IMFMQI_IIH_FLAGS= "MQIIH_NONE"/* "00000000"X 4*/
IMFMQI_IIH_LTERM_OVERRIDE = " "/* 8*/
IMFMQI_IIH_MFSMAPNAME= " "/* 8*/
IMFMQI_IIH_REPLYTOFORMAT = "MQFMT_NONE"/* " " 8*/
IMFMQI_IIH_AUTHENTICATOR = "MQIAUT_NONE"/* " " 8*/
IMFMQI_IIH_TRANSACTIONID = copies("00"X, 16)/* 16*/
IMFMQI_IIH_TRANSTATE= "MQITS_NOT_IN_CONVERSATION"/* " " 1*/
IMFMQI_IIH_COMMITMODE= "MQICM_SEND_THEN_COMMIT"/* "1" 1*/
IMFMQI_IIH_SECURITYSCOPE = "MQISS_CHECK"/* "C" 1*/
IMFMQI_IIH_RESERVED= " "/* " " 1*/
LL = "0050"X
ZZ = "0000"X
TRANCODE = LEFT("GBGTRNMI .", 76)
PUTDATA = left(' ', 84) || 11 || zz || TRANCODE
buf1 = length(putdata)
"IMFEXEC MQI PUT1 BUFFER(PUTDATA) " ||,
"BUFFLEN("BUFL") POPTS('MQPMO_NO_SYNCPOINT') "

```

---

During the IMFEXEC MQI PUT or PUT1 statement processing, the Message Descriptor, Message Buffer and IMS Information Header are created as follows:



## MQI BACK

This statement invokes the MQSeries BACK command which backs out MQ message changes since last syncpoint for all queries opened with SYNCPOINT.

Command	Parameters
IMFEXEC MQI BACK	HCONN(hconn)

The following table describes the IMFEXEC MQI BACK parameters.

Parameter	Function	Notes
HCONN	Provides the connection handle that represents the connection to the queue manager.	<p>This parameter</p> <ul style="list-style-type: none"> <li>• Specifies a variable created in an earlier CONN command</li> <li>• Is not required</li> </ul> <p>If omitted, the connection handle value is the default variable IMFHCONN. The connection handle is returned by a previous IMFEXEC MQI CONN request and stored in the IMFHCONN variable or a user-specified variable.</p>

### Example

```

/* REXX */
"IMFEXEC MQI CONN NAME(CSQ1)"           /* Connect to the queue manager */
IMFMQI_OD_OBJECTNAME = "TEST.QUEUE1"   /* Set queue name */
"IMFEXEC MQI OPEN OOPTS('MQOO_OUTPUT') " /* Open the queue */
IMFMQI_PMO_OPTIONS = 'MQPMO_SYNCPOINT'  /* Set Put options*/
PUTDATA = 'MQPUT001 - TEST DATA ECB = MQPUTECB1 POST CODE = PUTECB1'
"IMFEXEC MQI PUT BUFFER(PUTDATA)"       /* Put message on queue*/
/* Wait for another EXEC to post this EXEC and if it fails, backout changes*/
"IMFEXEC WAIT 30 NAME(MQPUTECB1)"       /* Wait on Post*/
IF IMFPOST /= 'PUTECB1' THEN
    "IMFEXEC MQI BACK"                   /* Back out the work */
"IMFEXEC MQI CLOSE COPTS(MQCO_NONE)"     /* Close the queue */
"IMFEXEC MQI DISC"                       /* Disconnect from queue manager */
EXIT

```

This command backs out all the changes that occurred since the last syncpoint (previous BACK or COMMIT). The following list describes the variable values that are used or set by this statement:

- Variable IMFHCONN contains the connection handle name.
- Variable IMFMQCC and IMFCC contain the completion code.
- Variable IMFMQRC and IMFRC contain the reason code.
- Variable IMFHOBJ contains the object handle for the queue.
- Variable IMFMQI\_REASON contains the constant (character) reason code.

## MQI CLOSE

This statement closes a previously opened queue.

Command	Parameters
IMFEXEC MQI CLOSE	HCONN(hconn) HOBJ(hobj) COPTS(options)

The following table describes the IMFEXEC MQI CLOSE parameters.

Parameter	Function	Notes
HCONN	Provides the connection handle that represents the connection to the queue manager.	This parameter <ul style="list-style-type: none"><li>• Specifies a variable created in an earlier CONN command</li><li>• Is not required</li></ul> If omitted, the connection handle value is the default variable, IMFHCONN. The connection handle is returned by a previous IMFEXEC MQI CONN request and stored in the IMFHCONN variable or a user-specified variable.

Parameter	Function	Notes
HOBJ	Provides the object handle that represents the object being closed.	<p>This parameter</p> <ul style="list-style-type: none"> <li>• Specifies a variable created in an earlier OPEN command</li> <li>• Is not required</li> </ul> <p>If omitted, the object handle value is taken from the default variable IMFHOBJ. The object handle is returned by a previous IMFEXEC MQI OPEN request and stored in the IMFHOBJ variable or a user-specified variable.</p>
COPTS	Specifies the options that control the action of CLOSE.	<p>This parameter is not required.</p> <p>If omitted, the close options value is taken from the default variable IMFMQI_CO_OPTIONS.</p> <p>If IMFMQI_CO_OPTIONS is not set, the default option of MQCO_NONE is used.</p> <p>Valid close options:  MQCO_NONE  MQCO_DELETE  MQCO_DELETE_PURGE</p>

The completion code is returned in the IMFMQCC and IMFCC variables. For additional information, see “How Completion and Reason Codes Are Returned” on page 156.

The reason code is returned in the IMFMQRC and IMFRC variables; contents of both variables are the same.

IMFMQI\_REASON contains the constant (character) reason code. Refer to the *IBM MQSeries Application Programming Reference Guide* for the reason code description.

## Example

---

```
/* REXX */
"IMFEXEC MQI CONN NAME(CSQ1)"           /* Connect to the queue manager */
IMFMQI_OD_OBJECTNAME = TEST.QUEUE1     /* Set queue name */
"IMFEXEC MQI OPEN OOPTS(MQOO_OUTPUT)"   /* Open the queue */
IMFMQI_PMO_OPTIONS = 'MQPMO_SYNCPOINT' /* Set Put options */
PUTDATA = 'MQPUT001 - TEST DATA ECB = MQPUTECB1 POST CODE = PUTECEB1'
IMFMQI_BUFFERLEN = LENGTH(PUTDATA)     /* Specify buffer length */
IMFMQI_BUFFER = PUTDATA                /* Place data in buffer */
"IMFEXEC MQI PUT"                       /* Put message on queue */
"IMFEXEC MQI CMIT"                       /* Commit the work */
"IMFEXEC MQI CLOSE COPTS(MQCO_NONE)"    /* Close the queue */
"IMFEXEC MQI DISC"                       /* Disconnect from queue manager */
EXIT
```

---

This example highlights closing an object defined by IMFHOBJ for the queue manager whose connection handle is stored in variable IMFHCONN. The following list describes the variable values that are used or set by this command statement:

- Variable IMFHCONN contains the connection handle name.
- Variable IMFHOBJ contains the object handle for the queue.
- Variable IMFMQCC and IMFCC contain the completion code.
- Variable IMFMQRC and IMFRC contain the reason code.
- Variable IMFMQI\_REASON contains the constant (character) reason code.

## MQI CMIT

This statement invokes the MQSeries CMIT command which commits all new messages and changes to queues opened with SYNCPoint.

Command	Parameters
IMFEXEC MQI CMIT	HCONN(hconn)

The following table describes the IMFEXEC MQI CMIT parameters.

Parameter	Function	Notes
HCONN	Provides the connection handle that represents the connection to the queue manager.	This parameter <ul style="list-style-type: none"><li>• Specifies a variable created in an earlier CONN command</li><li>• Is not required</li></ul> If omitted, the connection handle value is the default variable IMFHCONN. The connection handle is returned by a previous IMFEXEC MQI CONN request and stored in the IMFHCONN variable or a user-specified variable.

The completion code is returned in the IMFMQCC and IMFCC variables. For additional information, see “How Completion and Reason Codes Are Returned” on page 156.

The reason code is returned in the IMFMQRC and IMFRC variables; contents of both variables are the same.

IMFMQI\_REASON contains the constant (character) reason code. Refer to the *IBM MQSeries Application Programming Reference Guide* for the reason code description.

## Example

---

```
/* REXX */
"IMFEXEC MQI CONN NAME(CSQ1)"          /* Connect to the queue manager */
IMFMQI_OD_OBJECTNAME = TEST.QUEUE1    /* Set queue name */
"IMFEXEC MQI OPEN OOPTS(MQOO_OUTPUT)"  /* Open the queue */
IMFMQI_PMO_OPTIONS = 'MQPMO_SYNCPOINT' /* Set Put options */
PUTDATA = 'MQPUT001 - TEST DATA ECB = MQPUTECB1 POST CODE = PUTECB1'
IMFMQI_BUFFER = PUTDATA                /* Place data in buffer */
"IMFEXEC MQI PUT BUFFER(PUTDATA)"      /* Put message on queue */
/* Wait for another EXEC to post this EXEC and if it fails, backout out changes. */
"IMFEXEC WAIT 30 NAME(MQPUTECB1)"      /* Wait on Post */
IF IMFPOST /= 'PUTECB1' THEN
  "IMFEXEC MQI BACK"                   /* Back out the work */
ELSE
  "IMFEXEC MQI COMMIT"                 /* Commit the work */
"IMFEXEC MQI CLOSE COPTS(MQCO_NONE)"   /* Close the queue */
"IMFEXEC MQI DISC"                     /* Disconnect from queue manager */
EXIT
```

---

This example highlights committing all changes that have occurred since the last syncpoint. The following list describes the variable values that are used or set by this command statement:

- Variable IMFHCONN contains the connection handle name.
- Variable IMFMQCC and IMFCC contain the completion code.
- Variable IMFMQRC and IMFRC contain the reason code.
- Variable IMFMQI\_REASON contains the constant (character) reason code.
- Variable IMFHOBJ contains the object handle for the queue.

## MQI CONN

With the MQI CONN statement, an application can connect to a queue manager. It returns a queue manager connection handle to the EXEC, which is used by the application for all other IMFEXEC MQI statements.

Command	Parameters
IMFEXEC MQI CONN	NAME(qmgr) HCONN(hconn)

The following table describes the IMFEXEC MQI CONN parameters.

Parameter	Function	Notes
NAME	The name of a connectable queue manager.	
HCONN	Provides a variable to contain the connection handle.	<p>This parameter</p> <ul style="list-style-type: none"> <li>Specifies a variable where the connection handle will be placed</li> <li>Is not required</li> </ul> <p>If omitted, the connection handle value is the default variable IMFHCONN.</p>

The completion code is returned in the IMFMQCC and IMFCC variables. For additional information, see “How Completion and Reason Codes Are Returned” on page 156.

The reason code is returned in the IMFMQRC and IMFRC variables; contents of both variables are the same.

IMFMQI\_REASON contains the constant (character) reason code. Refer to the *IBM MQSeries Application Programming Reference Guide* for the reason code description.

### Example

---

```

/* REXX */
"IMFEXEC MQI CONN NAME(CSQ1)"           /* Connect to the queue manager */
IMFMQI_OD_OBJECTNAME = TEST.QUEUE1     /* Set queue name */
"IMFEXEC MQI OPEN OOPTS(MQOO_OUTPUT)"   /* Open the queue */
IMFMQI_PMO_OPTIONS = 'MQPMO_SYNCPOINT' /* Set Put options */
PUTDATA = 'MQPUT001 - TEST DATA ECB = MQPUTECB1 POST CODE = PUTECEB1'
"IMFEXEC MQI PUT BUFFER(PUTDATA)"       /* Put message on queue */
"IMFEXEC MQI COMMIT"                   /* Commit the work */
"IMFEXEC MQI CLOSE COPTS(MQCO_NONE)"   /* Close the queue */
"IMFEXEC MQI DISC"                      /* Disconnect from queue manager */
EXIT

```

---

This example highlights connecting the EXEC to the queue manager CSQ1. The following list describes the variable values that are used or set by this command:

- Variable IMFHCONN contains the connection handle name.
- Variable IMFMQCC and IMFCC contain the completion code.
- Variable IMFMQRC and IMFRC contain the reason code.
- Variable IMFHOBJ contains the object handle for the queue.
- Variable IMFMQI\_REASON contains the constant (character) reason code.

# MQI DISC

With the MQI DISC statement, an application can disconnect from a queue manager.

Command	Parameters
IMFEXEC MQI DISC	HCONN(hconn)

The following table describes the IMFEXEC MQI DISC parameters.

Parameter	Function	Notes
HCONN	Provides the connection handle that represents the connection to the queue manager.	<p>This parameter</p> <ul style="list-style-type: none"> <li>Specifies a variable created in an earlier CONN command</li> <li>Is not required</li> </ul> <p>If omitted, the connection handle value is the default variable IMFHCONN. The connection handle is returned by a previous IMFEXEC MQI CONN request and stored in the IMFHCONN variable or a user-specified variable.</p>

The completion code is returned in the IMFMQCC and IMFCC variables. For additional information, see “How Completion and Reason Codes Are Returned” on page 156.

The reason code is returned in the IMFMQRC and IMFRC variables; contents of both variables are the same.

IMFMQI\_REASON contains the constant (character) reason code. Refer to the *IBM MQSeries Application Programming Reference Guide* for the reason code description.

## Example

---

```

/* REXX */
"IMFEXEC MQI CONN NAME(CSQ1)"           /* Connect to the queue manager */
IMFMQI_OD_OBJECTNAME = TEST.QUEUE1     /* Set queue name */
"IMFEXEC MQI OPEN OOPTS(MQOO_OUTPUT)"   /* Open the queue */
IMFMQI_PMO_OPTIONS = 'MQPMO_SYNCPOINT' /* Set Put options */
PUTDATA = 'MQPUT001 - TEST DATA ECB = MQPUTECB1 POST CODE = PUTECEB1'
"IMFEXEC MQI PUT BUFFER(PUTDATA)"       /* Put message on queue */
"IMFEXEC MSG 'MQEXAMPL - TEST DATA ECB = MQPUTECB1 POST CODE = PUTECEB1'"
"IMFEXEC MQI COMMIT"                   /* Commit the work */
"IMFEXEC MQI CLOSE COPTS(MQCO_NONE)"    /* Close the queue */
"IMFEXEC MQI DISC"                      /* Disconnect from queue manager */
EXIT

```

---

This example highlights disconnecting from the queue manager whose connection handle is stored in variable IMFHCONN. The following list describes the variable values that are used or set by this command:

- Variable IMFHCONN contains the connection handle name.
- Variable IMFMQCC and IMFCC contain the completion code.
- Variable IMFMQRC and IMFRC contain the reason code.
- Variable IMFHOBJ contains the object handle for the queue.
- Variable IMFMQI\_REASON contains the constant (character) reason code.

## MQI GET

This statement is used to GET a message from a previously opened queue.

Command	Parameters
IMFEXEC MQI GET	HCONN(hconn) HOBJ(hobj) GOPTS(options) BUFFLEN(bufferlength) BUFFER(buffer) DATALEN(datalength)

The following table describes the IMFEXEC MQI GET parameters.

Parameter	Function	Notes
<p><b>Note:</b> The following list of constants may not be complete because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i>.</p>		
HCONN	Provides the connection handle that represents the connection to the queue manager.	<p>This parameter</p> <ul style="list-style-type: none"> <li>• Specifies a variable created in an earlier CONN command</li> <li>• Is not required</li> </ul> <p>If omitted, the connection handle value is the default variable IMFHCONN. The connection handle is returned by a previous IMFEXEC MQI CONN request and stored in the IMFHCONN variable or a user-specified variable. Either this parameter, specifying an existing variable that contains the connection handle, or a previously created IMFHCONN variable containing the connection handle is required.</p>
HOBJ	Provides the object handle that represents the queue from which the message is returned.	<p>This parameter</p> <ul style="list-style-type: none"> <li>• Specifies a variable created in an earlier OPEN command</li> <li>• Is not required</li> </ul> <p>If omitted, the object handle value is taken from the default variable IMFHOBJ. The object handle is returned by a previous IMFEXEC MQI OPEN request and stored in the IMFHOBJ variable or a user-specified variable. Either this parameter, specifying an existing variable that contains the object handle, or a previously created IMFHOBJ variable containing the object handle is required.</p>

Parameter	Function	Notes
GOPTS	Specifies the options that control the action of GET.	<p>This parameter is not required.</p> <p>If omitted, the GET options value is taken from the default variable IMFMQI_GMO_OPTIONS.</p> <p>The variable IMFMQI_GMO_OPTIONS is not required. If GET options are not specified, MQSeries will use the default, MQGMO_NO_WAIT.</p> <p>More than one of the following options can be specified in any valid combination.</p> <p>Valid GET options:</p> <ul style="list-style-type: none"> <li>MQGMO_WAIT</li> <li>MQGMO_NO_WAIT</li> <li>MQGMO_SYNCPOINT</li> <li>MQGMO_SYNCPOINT_IF_PERSISTENT</li> <li>MQGMO_NO_SYNCPOINT</li> <li>MQGMO_MARK_SKIP_BACKOUT</li> <li>MQGMO_BROWSE_FIRST</li> <li>MQGMO_BROWSE_NEXT</li> <li>MQGMO_MSG_UNDER_CURSOR</li> <li>MQGMO_ACCEPT_TRUNCATED_MSG</li> <li>MQGMO_SET_SIGNAL</li> <li>MQGMO_FAIL_IF QUIESCING</li> <li>MQGMO_CONVERT</li> <li>MQGMO_NONE</li> </ul> <p><i>See note at the beginning of this table.</i></p>
GOPTS continued	Specifies the options that control the action of GET.	<p>In addition to the above options, another AutoOPERATOR-provided option is available to the user. This option is</p> <p><b>REMOVE_DLH</b></p> <p>Using the REMOVE_DLH option, you can retrieve messages from a Dead Letter queue, process them and forward them to an application queue without the Dead Letter Header. The REMOVE_DLH option is added to the GOPTS() keyword parameter of the IMFEXEC MQI GET statement. This AutoOPERATOR option causes the <b>IMFMQI_MD_ENCODING</b>, <b>IMFMQI_MD_CODEDCHARSETID</b> and <b>IMFMQI_MD_FORMAT</b> variables to be set from the Dead Letter Header instead of the message descriptor. The returned buffer is without the DLH. This action occurs before control is returned to the EXEC following the IMFEXEC MQI GET command. The message can optionally be placed directly onto another queue.</p>

Parameter	Function	Notes
BUFFER	Specifies the variable that contains the message buffer data following a successful GET.	This parameter is not required. If omitted, the variable IMFMQI_BUFFER will contain the Message Buffer data upon successful completion of the GET.
BUFFLEN	Specifies the length of the data used to create the variable containing the message buffer data.	<p>This parameter</p> <ul style="list-style-type: none"> <li>• Is not required</li> <li>• Can specify a numeric value or the name of a variable that contains a numeric value</li> <li>• Can specify a value in the range of 0 to 32767, either specifically, or using a variable</li> </ul> <p>If this parameter is omitted, the variable IMFMQI_BUFFLEN is used to obtain the BUFFER length. If the variable is not set, the message length is used. The recommendation is that you do not set the variable so that the message length is used.</p> <p><b>Note:</b> The BUFFER variable will be left justified and blank-padded to the right up to the value of the BUFFLEN variable. If you want to change the BUFFER variable size to be exactly the message size, use the REXX SUBSTR or LEFT built-in function. Examples follow:</p> <pre>IMFMQI_BUFFER = substr(IMFMQI_BUFFER,1,IMFMQI_DATALEN)</pre> <p>or</p> <pre>IMFMQI_BUFFER = LEFT(IMFMQI_BUFFER,IMFMQI_DATALEN)</pre>
DATALEN	Specifies the variable to be set to the message length following a successful GET.	<p>This parameter is not required.</p> <p>If omitted, the variable IMFMQI_DATALEN is set with the length of the message data returned from the GET. If the value is greater than the buffer length value, only buffer length bytes are returned in the buffer variable. If the value is zero, it means that the message contains no application data.</p>

The following table describes the most common input and output variables for the IMFEXEC MQI GET command. These variables, whether used as input or output, are always accessible after a successful GET. For structures other than the MD (message descriptor), refer to Appendix D. Variables created from structures begin with “IMFMQI\_”, followed by the MQSeries structure, followed by the field name within the MQSeries structure, with each node delimited by an underscore. For definitions of field names and constants used in this table that

are derived from MQSeries structure field names, see the *IBM MQSeries Application Programming Reference Guide*.

Variable name	Source	Notes
<p><b>Note:</b> The following list of constants may not be complete because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i>.</p>		
IMFMQI_STRUCTURES	Output	<p>Contains a blank-delimited list of the MQSeries structures contained in the message. For example, in the case of a Dead Letter message, the contents would contain at least:</p> <p>'MD DLH'</p>
IMFMQI_OFFSETS	Output	<p>Contains a blank-delimited list of the offsets of the structures contained in the message (as well as the offset of the application data) from the beginning of the message buffer. Using this variable, you can access application data beyond the structures in the message buffer. The number of entries in the list is always one more than the number of structures participating in the operation (IMFMQI_STRUCTURES) because an entry for the offset of the application data is included in the list. A 'NONE' is always used for a structure that does not exist in the buffer, such as 'MD'. For example, if the message was a dead letter message and contained some application data, the variable might look like this:</p> <p>'NONE 0 172'</p> <p>In the above example, NONE is specified for the MD, 0 for the offset of the Dead Letter Header and 172 is the offset of the application data.</p>
IMFMQI_MD_STRUCID	Input	<p>Optional. The EXEC may or may not set this variable. If the EXEC does not set this value, the MQSeries default will be used, unmodified by the EXEC Manager.</p>
IMFMQI_MD_VERSION	Input	<p>Optional. The EXEC may or may not set this variable. If the EXEC does not set this value, the MQSeries default will be used unmodified by the EXEC Manager.</p>

Variable name	Source	Notes
IMFMQI_MD_REPORT	Output	<p>Contains one or more of the following delimited by blanks:</p> <p>MQRO_EXCEPTION  MQRO_EXCEPTION_WITH_DATA  MQRO_EXCEPTION_WITH_FULL_DATA  MQRO_EXPIRATION  MQRO_EXPIRATION_WITH_DATA  MQRO_EXPIRATION_WITH_FULL_DATA  MQRO_COA  MQRO_COA_WITH_DATA  MQRO_COA_WITH_FULL_DATA  MQRO_COD  MQRO_COD_WITH_DATA  MQRO_COD_WITH_FULL_DATA  MQRO_PAN  MQRO_NAN  MQRO_NEW_MSG_ID  MQRO_PASS_MSG_ID  MQRO_COPY_MSG_ID_TO_CORREL_ID  MQRO_PASS_CORREL_ID  MQRO_DEAD_LETTER_Q  MQRO_DISCARD_MSG  MQRO_NONE</p> <p><i>See note at the beginning of this table.</i></p>
IMFMQI_MD_MSGTYPE	Output	<p>Contains one of the following:</p> <p>MQMT_DATAGRAM  MQMT_REQUEST  MQMT_REPLY  MQMT_REPORT</p> <p><i>See note at the beginning of this table.</i></p>
IMFMQI_MD_EXPIRY	Output	<p>Contains a decimal number, in the range 1 to 999999999, or MQEI_UNLIMITED.</p>

Variable name	Source	Notes
IMFMQI_MD_FEEDBACK	Output	<p>Contains one of the following values for report messages:</p> <ul style="list-style-type: none"> <li>MQFB_NONE</li> <li>MQFB_COA</li> <li>MQFB_COD</li> <li>MQFB_EXPIRATION</li> <li>MQFB_PAN</li> <li>MQFB_NAN</li> <li>MQFB_QUIT</li> </ul> <p>In addition, this field can contain any reason code from the following sources:</p> <ul style="list-style-type: none"> <li>• IMS-bridge feedback codes</li> <li>• CICS-bridge feedback codes</li> <li>• MQSeries reason codes</li> </ul> <p><i>See note at the beginning of this table.</i></p>
IMFMQI_MD_ENCODING	Output	<p>Contains</p> <ul style="list-style-type: none"> <li>MQENC_NATIVE</li> </ul> <p>Or any decimal number up to 999999999.</p>
IMFMQI_MD_CODEDCHARSETID	Input/Output	<p>Contains</p> <ul style="list-style-type: none"> <li>MQCCSI_Q_MGR</li> <li>MQCCSI_EMBEDDED</li> </ul> <p>Or any decimal number up to 999999999. See <i>IBM MQSeries Application Programming Reference Guide</i> for details on how to use the CodedCharSetId field as input.</p>

Variable name	Source	Notes
IMFMQI_MD_FORMAT	Output	<p>Contains one of the following formats:</p> <p>MQFMT_NONE  MQFMT_ADMIN  MQFMT_CHANNEL_COMPLETED  MQFMT_CICS  MQFMT_COMMAND_1  MQFMT_COMMAND_2  MQFMT_DEAD_LETTER_HEADER  MQFMT_EVENT  MQFMT_IMS  MQFMT_IMS_VAR_STRING  MQFMT_MD_EXTENSION  MQFMT_PCF  MQFMT_REF_MSG_HEADER  MQFMT_STRING  MQFMT_TRIGGER  MQFMT_WORK_INFO_HEADER  MQFMT_XMIT_Q_HEADER  MQFMT_RF_HEADER  MQFMT_RF_HEADER_2</p> <p><i>See note at the beginning of this table.</i></p>
IMFMQI_MD_PRIORITY	Output	Contains a decimal number in the range 0 to 999999999.
IMFMQI_MD_PERSISTENCE	Output	<p>Contains one of the following:</p> <p>MQPER_PERSISTENT  MQPER_NOT_PERSISTENT</p>
IMFMQI_MD_MSGID	Input/Output	Processes or receives up to a 32-byte MsgId. Processed exactly as received for input to, and as output from, the GET. No conversion of hexadecimal data is done in either case. For input, the value MQMI_NONE is valid.
IMFMQI_MD_CORRELID	Input/Output	Processes or receives up to a 32-byte CORRELID. Processed exactly as received for input to and as output from the GET. No conversion of hexadecimal data is done in either case. For input, the following constant values is recognized and converted to their corresponding hexadecimal equivalent: MQCI_NONE MQCI_NEW_SESSION
IMFMQI_MD_BACKOUTCOUNT	Output	Contains a decimal value in the range 0 to 255.
IMFMQI_MD_REPLYTOQ	Output	Contains up to a 48-character name of the ReplyToQ.

Variable name	Source	Notes
IMFMQI_MD_REPLYTOQMGR	Output	Contains up to a 48-character name of the ReplyToQMgr.
IMFMQI_MD_USERIDENTIFIER	Output	Contains up to a 12-character UserIdentifier.
IMFMQI_MD_ACCOUNTINGTOKEN	Output	Contains MQACT_NONE or up to a 32-byte AccountingToken. Processed exactly as received from the GET. No conversion of hexadecimal data is done.
IMFMQI_MD_APPLIDENTITYDATA	Output	Contains up to a 32-character value for ApplIdentityData.
IMFMQI_MD_PUTAPPLTYPE	Output	<p>Contains a user-defined type or one of the following standard types:</p> <ul style="list-style-type: none"> <li>MQAT_AIX</li> <li>MQAT_BROKER</li> <li>MQAT_CICS</li> <li>MQAT_CICS_BRIDGE</li> <li>MQAT_CICS_VSE</li> <li>MQAT_DEFAULT</li> <li>MQAT_DOS</li> <li>MQAT_DQM</li> <li>MQAT_GUARDIAN</li> <li>MQAT_IMS</li> <li>MQAT_IMS_BRIDGE</li> <li>MQAT_JAVA</li> <li>MQAT_MVS</li> <li>MQAT_NOTES_AGENT</li> <li>MQAT_NSK</li> <li>MQAT_OS2</li> <li>MQAT_OS390</li> <li>MQAT_OS400</li> <li>MQAT_QMGR</li> <li>MQAT_UNKNOWN</li> <li>MQAT_UNIX</li> <li>MQAT_VMS</li> <li>MQAT_VOS</li> <li>MQAT_WINDOWS</li> <li>MQAT_WINDOWS_NT</li> <li>MQAT_XCF</li> </ul> <p><i>See note at the beginning of this table.</i></p>
IMFMQI_MD_PUTAPPLNAME	Output	Contains up to a 28-character value for PutApplName.
IMFMQI_MD_PUTDATE	Output	Contains an 8-character date stamp.
IMFMQI_MD_PUTTIME	Output	Contains an 8-character time stamp.
IMFMQI_MD_APPLORIGINDATA	Output	Contains a 4-character value for ApplOriginData.

Variable name	Source	Notes
IMFMQI_MD_GROUPID	Output	If an MQMD_VERSION_2 MD is present, this variable may contain a 24-byte GROUPID. Processed exactly as received from the GET. No conversion of hexadecimal data is done.
IMFMQI_MD_MSGSEQNUMBER	Output	If an MQMD_VERSION_2 MD is present, this variable may contain a decimal number in the range 1 to 999999999.
IMFMQI_MD_OFFSET	Output	If an MQMD_VERSION_2 MD is present, this variable may contain a decimal number in the range 0 to 999999999.
IMFMQI_MD_MSGFLAGS	Output	If an MQMD_VERSION_2 MD is present, this variable may contain one or more of the following character strings delimited by blanks: MQMF_SEGMENTATION_INHIBITED MQMF_SEGMENTATION_ALLOWED MQMF_MSG_IN_GROUP MQMF_LAST_MSG_IN_GROUP MQMF_SEGMENT MQMF_LAST_SEGMENT MQMF_NONE  <i>See note at the beginning of this table.</i>
IMFMQI_MD_ORIGINALLENGTH	Output	Contains a decimal number, in the range 1 to 999999999, or MQOL_UNDEFINED.  <i>See note at the beginning of this table.</i>

The completion code is returned in the IMFMQCC and IMFCC variables. For additional information, see “How Completion and Reason Codes Are Returned” on page 156.

The reason code is returned in the IMFMQRC and IMFRC variables; contents of both variables are the same. Refer to the *IBM MQSeries Application Programming Reference Guide* for the reason code description.

## Example

---

```

/* REXX */
"IMFEXEC MQI CONN NAME(CSBD)"                               /* Connect to the queue manager */
IMFMQI_OD_OBJECTNAME = JOHNB.QUEUE4                         /* Set queue name */
"IMFEXEC MQI OPEN OOPTS(MQOO_BROWSE)"                       /* Open the queue */
IMFMQI_GMO_OPTIONS = 'MQGMO_BROWSE_NEXT'                  /* Browse msg on queue */
"IMFEXEC MQI GET BUFFER(GETDATA)"
"IMFEXEC MQI CLOSE COPTS(MQCO_NONE)"                       /* Close the queue */
"IMFEXEC MQI DISC"                                         /* Disconnect from queue manager */
EXIT

```

---

This example highlights GETting a message from a local queue. The following describes the variable values that are in effect during the GET processing:

- Variable IMFHCONN contains the connection handle name.

- Variable IMFHOBJ contains the object handle for the queue.
- Variable IMFMQCC and IMFCC contain the completion code.
- Variable IMFMQRC and IMFRC contain the reason code.
- Variable IMFMQI\_REASON contains the constant (character) reason code.

## MQI OPEN

This statement opens a queue.

Command	Parameters
IMFEXEC MQI OPEN	HCONN(hconn) HOBJ(hobj) OOPTS(options)

The following table describes the IMFEXEC MQI OPEN parameters.

Parameter	Function	Notes
<p><b>Note:</b> The following list of constants may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i>. Also note that when specifying multiple options, they must be enclosed within single quotation marks. For example, OOPTS('MQOO_OUTPUT MQOO_SET_ALL_CONTEXT')</p>		
HCONN	Provides the connection handle that represents the connection to the queue manager.	<p>This parameter:</p> <ul style="list-style-type: none"> <li>• Specifies a variable created in an earlier CONN command</li> <li>• Is not required</li> </ul> <p>If omitted, the connection handle value is the default variable IMFHCONN. The connection handle is returned by a previous IMFEXEC MQI CONN request and stored in the IMFHCONN variable or a user-specified variable. Either this parameter, specifying an existing variable that contains the connection handle, or a previously created IMFHCONN variable containing the connection handle is required.</p>

Parameter	Function	Notes
HOBJ	Provides the object handle that represents the object being opened.	<p>This parameter:</p> <ul style="list-style-type: none"> <li>• Specifies a variable to be created to contain the object handle.</li> <li>• Is not required.</li> </ul> <p>If omitted, the object handle value is placed into the default variable IMFHOBJ. The object handle is returned by the IMFEXEC MQI OPEN request and stored in the IMFHOBJ variable or a user-specified variable.</p>
OOPTS	Specifies the options that control the OPEN action.	<p>This parameter is not required.</p> <p>If omitted, the OPEN options value is taken from the default variable IMFMQI_OO_OPTIONS. Either the parameter or the variable is required.</p> <p>Options can be specified as any valid combination of the following:</p> <p style="margin-left: 40px;"> MQOO_BIND_NOT_FIXED  MQOO_BIND_ON_OPEN  MQOO_FAIL_IF QUIESCING  MQOO_ALTERNATE_USER_AUTHORITY  MQOO_SET_ALL_CONTEXT  MQOO_SET_IDENTITY_CONTEXT  MQOO_PASS_ALL_CONTEXT  MQOO_PASS_IDENTITY_CONTEXT  MQOO_SAVE_ALL_CONTEXT  MQOO_INQUIRE  MQOO_OUTPUT  MQOO_BROWSE  MQOO_INPUT_EXCLUSIVE  MQOO_INPUT_SHARED  MQOO_INPUT_AS_Q_DEF  MQOO_BIND_AS_Q_DEF </p> <p><i>See note at the beginning of this table.</i></p>

This table describes the variables used to build the object descriptor prior to issuing the OPEN statement.

Variable name	Source	Notes
<p><b>Note:</b> The following list of constants may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i>. Also note that when specifying multiple options, they must be enclosed within single quotation marks. For example, OOPTS('MQOO_OUTPUT MQOO_SET_ALL_CONTEXT')</p>		
IMFMQI_OD_STRUCID	Input	Optional. The EXEC may or may not set this variable. If the EXEC does not set this variable, the MQSeries default value will be used unmodified by the EXEC Manager.
IMFMQI_OD_VERSION	Input	Optional. The EXEC may or may not set this variable. If the EXEC does not set this variable, the MQSeries default value will be used, unmodified by the EXEC Manager.
IMFMQI_OD_OBJECTTYPE	Input	Optional. The EXEC may or may not set this variable. If the EXEC does not set this variable, the MQSeries default value will be used, unmodified by the EXEC Manager.
IMFMQI_OD_OBJECTNAME	Input	Required. Contains the queue name to be opened. Up to 48 characters can be specified.
IMFMQI_OD_OBJECTQMGRNAME	Input	Optional. Contains the 48-character object queue manager name. The EXEC Manager does not specify any default value when this variable is not set.
IMFMQI_OD_DYNAMICQNAME	Input	Optional. Contains the 48-character dynamic queue name when using a model queue. The EXEC Manager does not specify any default value when this variable is not set.
IMFMQI_OD_ALTERNATEUSERID	Input	Optional. Contains the 12-character alternate user identifier. The EXEC Manager does not specify any default value when this variable is not set.
IMFMQI_OD_RECSPRESENT	Input	Optional. The EXEC may or may not set this variable. If the EXEC does not set this variable, the MQSeries default value will be used, unmodified by the EXEC Manager.
IMFMQI_OD_KNOWNDESTCOUNT	Output	This variable contains 0.
IMFMQI_OD_UNKNOWNDDESTCOUNT	Output	This variable contains 0.
IMFMQI_OD_INVALIDDESTCOUNT	Output	This variable contains 0.
IMFMQI_OD_OBJECTRECOFFSET	Input	This variable contains 0.
IMFMQI_OD_RESPONSERECOFFSET	Input	This variable contains 0.
IMFMQI_OD_OBJECTRECPtr	Input	This variable contains 0.
IMFMQI_OD_RESPONSERECPtr	Input	This variable contains 0.

Variable name	Source	Notes
IMFMQI_OD_ALTERNATESECURITYID	Input	This variable exists but is null.
IMFMQI_OD_RESOLVEDQNAME	Output	This variable contains blanks.
IMFMQI_OD_RESOLVEDQMGRNAME	Output	This variable contains blanks.
IMFMQI_STRUCTURES	Output	Contains a blank-delimited list of the MQSeries structures used for this statement. For this statement, the variable contains 'OD'.
IMFMQI_OFFSETS	Output	Contains the value 'NONE NONE'. The variable is provided for consistency with other IMFEXEC MQI statements that use this variable and is not of significant use for the IMFEXEC MQI OPEN statement.

The completion code is returned in the IMFMQCC and IMFCC variables. For additional information, see "How Completion and Reason Codes Are Returned" on page 156.

The reason code is returned in the IMFMQRC and IMFRC variables; contents of both variables are the same. Refer to the *IBM MQSeries Application Programming Reference Guide* for the reason code description.

## Example

---

```

/* REXX */
"IMFEXEC MQI CONN NAME(CSQ1)"           /* Connect to the queue manager */
IMFMQI_OD_OBJECTNAME = TEST.QUEUE1     /* Set queue name */
"IMFEXEC MQI OPEN OOPTS(MQOO_OUTPUT)"   /* Open the queue */
IMFMQI_PMO_OPTIONS = 'MQPMO_SYNCPOINT' /* Set Put options */
PUTDATA = 'MQPUT001 - TEST DATA ECB = MQPUTECB1 POST CODE = PUTECEB1'
"IMFEXEC MQI PUT BUFFER(PUTDAT)"        /* Put message on queue */
"IMFEXEC MQI COMMIT"                    /* Commit the work */
"IMFEXEC MQI CLOSE COPTS(MQCO_NONE)"    /* Close the queue */
"IMFEXEC MQI DISC"                       /* Disconnect from queue manager */
EXIT

```

---

This example highlights connecting to the queue manager whose connection handle will be stored in variable IMFHCONN.

- Variable IMFHCONN contains the connection handle name.
- Variable IMFMQCC and IMFCC contain the completion code.
- Variable IMFMQRC and IMFRC contain the reason code.
- Variable IMFHOBJ contains the object handle for the queue.
- Variable IMFMQI\_REASON contains the constant (character) reason code.

## MQI PUT

This statement puts a message to a previously opened queue.

Command	Parameters
IMFEXEC MQI PUT	HCONN(hconn) HOBJ(hobj) POPTS(options) BUFFLEN(bufferlength) BUFFER(buffer)

The following table describes the IMFEXEC MQI PUT parameters.

Parameter	Function	Notes
<p><b>Note:</b> The following list of constants may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i>. Also note that when specifying multiple options, they must be enclosed within single quotation marks. For example, POPTS('MQPMO_NO_SYNCPOINT MQPMO_SET_ALL_CONTEXT')</p>		
HCONN	Provides the connection handle that represents the connection to the queue manager.	<p>This parameter:</p> <ul style="list-style-type: none"> <li>• Specifies a variable created in an earlier CONN command</li> <li>• Is not required</li> </ul> <p>If omitted, the connection handle value is the default variable IMFHCONN. The connection handle is returned by a previous IMFEXEC MQI CONN request and stored in the IMFHCONN variable or a user-specified variable. Either this parameter, specifying an existing variable that contains the connection handle, or a previously created IMFHCONN variable containing the connection handle is required.</p>
HOBJ	Provides the object handle that represents the queue to which the message is being PUT.	<p>This parameter:</p> <ul style="list-style-type: none"> <li>• Specifies a variable created in an earlier OPEN command</li> <li>• Is not required</li> </ul> <p>If omitted, the object handle value is taken from the default variable IMFHOBJ. The object handle is returned by a previous IMFEXEC MQI OPEN request and stored in the IMFHOBJ variable or a user-specified variable. Either this parameter, specifying an existing variable that contains the object handle, or a previously created IMFHOBJ variable containing the object handle is required.</p>

Parameter	Function	Notes
POPTS	Specifies the options that control the PUT action.	<p>This parameter is not required.</p> <p>If omitted, the PUT options value is taken from the default variable IMFMQI_PMO_OPTIONS.</p> <p>The variable IMFMQI_PMO_OPTIONS is not required. Options can be specified as any valid combination of the following choices:</p> <ul style="list-style-type: none"> <li>MQPMO_SYNCPOINT</li> <li>MQPMO_NO_SYNCPOINT</li> <li>MQPMO_NO_CONTEXT</li> <li>MQPMO_DEFAULT_CONTEXT</li> <li>MQPMO_PASS_IDENTITY_CONTEXT</li> <li>MQPMO_PASS_ALL_CONTEXT</li> <li>MQPMO_SET_IDENTITY_CONTEXT</li> <li>MQPMO_SET_ALL_CONTEXT</li> <li>MQPMO_ALTERNATE_USER_AUTHORITY</li> <li>MQPMO_FAIL_IF QUIESCING</li> <li>MQPMO_NONE (IBM default)</li> </ul> <p><i>See note at the beginning of this table.</i></p> <p>AutoOPERATOR provides you with another option, REMOVE_DLH, in addition to the above options.</p> <p>With the REMOVE_DLH option, you can remove the Dead Letter Header from a message before putting it to another queue. The REMOVE_DLH is added to the POPTS() keyword parameter of the IMFEXEC MQI PUT statement. This is an AutoOPERATOR option that causes the PUT message to be rebuilt by reloading the <b>Encoding</b>, <b>CodedCharSetId</b> and <b>Format</b> fields from the Dead Letter Header (DLH) into the Message Descriptor. This occurs before the PUT statement executes. The DLH, if present, is removed from the buffer containing the data.</p>

Parameter	Function	Notes
BUFFER	Specifies the variable that contains the message buffer data before issuing the PUT.	<p>This parameter is not required. If omitted, the variable IMFMQI_BUFFER must contain the message buffer data in order to complete the PUT. Be aware that upon successful completion of the PUT statement, the original buffer variable may be different if you have specified structure variables prior to the PUT. For example, you may want to build a Dead Letter message by changing the format variable, IMFMQI_MD_FORMAT, to MQFMT_DEAD_LETTER_HEADER and by setting variables starting with IMFMQI_DLH_ and ending with Dead Letter (DLH) field names (see Appendix D for lists of structure variables). In this example, AutoOPERATOR updates the buffer from the variables set before issuing the MQSeries PUT (see the section entitled “Creating Messages from Variables” on page 157 to learn how AutoOPERATOR creates messages out of variables). The Dead Letter header is prefixed in front of the data that is in the buffer. AutoOPERATOR updates the variable specified for the buffer so that it contains the DLH structure before control is returned to the EXEC.</p>
BUFFLEN	Specifies the length of the data used to create the variable containing the message buffer data.	<p>This parameter:</p> <ul style="list-style-type: none"> <li>• Is not required.</li> <li>• Can specify either a numeric value or the name of a variable that contains a numeric value.</li> <li>• Can specify a value in the range of 0 to 32767, either specifically or using variable.</li> </ul> <p>If this parameter is omitted, the variable IMFMQI_BUFFLEN will be used to obtain the BUFFER length. If neither is set, the length of the data in the message buffer variable is used.</p>

The following table describes the most common input and output variables for the IMFEXEC MQI PUT statement. Any required variables must be set prior to the PUT. For structures other than the MD (message descriptor), refer to Appendix D, “EXECs Variables” on page 271. Variables used begin with “IMFMQI\_”, followed by the MQSeries structure, followed by the field name within the MQSeries structure, with each node delimited by an underscore. For

definitions of field names and constants used in this table that are derived from MQSeries structure field names, see the *IBM MQSeries Application Programming Reference Guide*.

Variable name	Source	Notes
<b>Note:</b>		The following list of constants may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i> . Also note that when specifying multiple options, they must be enclosed within single quotation marks. For example, POPTS('MQPMO_NO_SYNCPOINT MQPMO_SET_ALL_CONTEXT')
IMFMQI_MD_STRUCID	Input	Optional. The EXEC may or may not set this variable. If the EXEC does not set this variable, the MQSeries default value will be used, unmodified by the EXEC Manager.
IMFMQI_MD_VERSION	Input	Optional. The EXEC may or may not set this variable. If the EXEC does not set this variable, the MQSeries default value will be used, unmodified by the EXEC Manager.
IMFMQI_MD_REPORT	Input	<p>Optional. Specify one or more of the following delimited by blanks:</p> <p>MQRO_EXCEPTION  MQRO_EXCEPTION_WITH_DATA  MQRO_EXCEPTION_WITH_FULL_DATA  MQRO_EXPIRATION  MQRO_EXPIRATION_WITH_DATA  MQRO_EXPIRATION_WITH_FULL_DATA  MQRO_COA  MQRO_COA_WITH_DATA  MQRO_COA_WITH_FULL_DATA  MQRO_COD  MQRO_COD_WITH_DATA  MQRO_COD_WITH_FULL_DATA  MQRO_PAN  MQRO_NAN  MQRO_NEW_MSG_ID  MQRO_PASS_MSG_ID  MQRO_COPY_MSG_ID_TO_CORREL_ID  MQRO_PASS_CORREL_ID  MQRO_DEAD_LETTER_Q  MQRO_DISCARD_MSG  MQRO_NONE (IBM default)</p> <p><i>See note at the beginning of this table.</i></p>
IMFMQI_MD_MSGTYPE	Input	<p>Optional. Specify one of the following:</p> <p>MQMT_DATAGRAM (IBM default)  MQMT_REQUEST  MQMT_REPLY  MQMT_REPORT</p> <p><i>See note at the beginning of this table.</i></p>

Variable name	Source	Notes
IMFMQI_MD_EXPIRY	Input	Optional. Specify a decimal number, in the range 1 to 999999999, or MQEI_UNLIMITED (IBM default).
IMFMQI_MD_FEEDBACK	Input	<p>Optional. Specify one of the following values for report messages:</p> <ul style="list-style-type: none"> <li>MQFB_NONE (IBM default)</li> <li>MQFB_COA</li> <li>MQFB_COD</li> <li>MQFB_EXPIRATION</li> <li>MQFB_PAN</li> <li>MQFB_NAN</li> <li>MQFB_QUIT</li> </ul> <p><i>See note at the beginning of this table.</i></p> <p>In addition, this variable can contain any reason code from the following sources:</p> <ul style="list-style-type: none"> <li>IMS-bridge feedback codes</li> <li>CICS-bridge feedback codes</li> <li>MQSeries reason codes</li> <li>User-specified numeric codes</li> </ul>
IMFMQI_MD_ENCODING	Input	Optional. Specify MQENC_NATIVE (IBM default) or any decimal number up to 999999999.
IMFMQI_MD_CODEDCHARSETID	Input	<p>Optional. Contains one of the following:</p> <ul style="list-style-type: none"> <li>MQCCSI_Q_MGR</li> <li>MQCCSI_EMBEDDED</li> </ul> <p>Or any decimal number up to 999999999.</p> <p><i>See note at the beginning of this table.</i></p>

Variable name	Source	Notes
IMFMQI_MD_FORMAT	Input	<p>Optional. Specify one of the following:</p> <p>MQFMT_NONE (IBM default)  MQFMT_ADMIN  MQFMT_CHANNEL_COMPLETED  MQFMT_CICS  MQFMT_COMMAND_1  MQFMT_COMMAND_2  MQFMT_DEAD_LETTER_HEADER  MQFMT_EVENT  MQFMT_IMS  MQFMT_IMS_VAR_STRING  MQFMT_MD_EXTENSION  MQFMT_PCF  MQFMT_REF_MSG_HEADER  MQFMT_STRING  MQFMT_TRIGGER  MQFMT_WORK_INFO_HEADER  MQFMT_XMIT_Q_HEADER  MQFMT_RF_HEADER  MQFMT_RF_HEADER_2</p> <p><i>See note at the beginning of this table.</i></p>
IMFMQI_MD_PRIORITY	Input	<p>Optional. Specify a decimal number in the range 0 to 999999999 or MQPRI_PRIORITY_AS_Q_DEF (IBM default).</p>
IMFMQI_MD_PERSISTENCE	Input	<p>Optional. Specify one of the following:</p> <p>MQPER_PERSISTENT  MQPER_NOT_PERSISTENT  MQPER_PERSISTENCE_AS_Q_DEF (IBM default)</p> <p><i>See note at the beginning of this table.</i></p>
IMFMQI_MD_MSGID	Input/Output	<p>Optional. Specify up to a 32-byte MsgId. Processed exactly as received. No conversion of hexadecimal data is done. For input, the value MQMI_NONE is valid. After a successful PUT, the variable is set from the field in the message descriptor.</p>
IMFMQI_MD_CORRELID	Input/Output	<p>Optional. Specify up to a 32-byte CORRELID. Processed exactly as received. No conversion of hexadecimal data is done. After a successful PUT, the variable is set from the field in the message descriptor. For input, the following values are valid:</p> <p>MQCI_NONE (IBM default)  MQCI_NEW_SESSION</p>

Variable name	Source	Notes
IMFMQI_MD_BACKOUTCOUNT	N/A	This variable is not used for PUT.
IMFMQI_MD_REPLYTOQ	Input	Optional. Specify up to a 48-character name of the ReplyToQ. The IBM default value is 48 blanks.
IMFMQI_MD_REPLYTOQMGR	Input	Optional. Specify up to a 48-character name of the ReplyToQMgr. The IBM default value is 48 blanks.
IMFMQI_MD_USERIDENTIFIER	Input/Output	Optional. Specify up to a 12-character UserIdentifier. See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the UserIdentifier field is used as input and output.
IMFMQI_MD_ACCOUNTINGTOKE N	Output	Optional. Contains MQACT_NONE (IBM default) or up to a 32-byte AccountingToken. Processed exactly as received. No conversion of hexadecimal data is done.
IMFMQI_MD_APPLIDENTITYDATA	Input/Output	Optional. Contains up to a 32-character value for ApplIdentityData. See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the ApplIdentityData field is used as input and output.
IMFMQI_STRUCTURES	Output	Contains a blank-delimited list of the MQSeries structures used in the creation of the message. For example, in the case of a Dead Letter message, the contents would contain at least, 'MD DLH'.
IMFMQI_OFFSETS	Output	<p>Contains a blank-delimited list of the offsets of the structures used in the creation of the message (as well as the offset of the application data) from the beginning of the message buffer. Using this variable you can access application data beyond the structures in the message buffer after issuing the PUT. The number of entries in the list is always one more than the number of structures participating in the operation, including MD, which participates in the operation but is not part of the message buffer. This is because an entry for the offset of the application data is included in the list. The value 'NONE' is always used for a structure that does not exist in the buffer but is used in creating the message for PUT, such as 'MD'. For example, if the message was a dead letter message and contained application data, the variable may look like this:</p> <p>'NONE 0 172'</p> <p>In the above example, NONE is specified for the MD, 0 for the offset of the Dead Letter Header and 172 is the offset of the application data.</p>

Variable name	Source	Notes
IMFMQI_MD_PUTAPPLTYPE	Input/Output	<p>Optional. Specify one of the following standard types or a numeric user-defined type:</p> <ul style="list-style-type: none"> <li>MQAT_AIX</li> <li>MQAT_BROKER</li> <li>MQAT_CICS</li> <li>MQAT_CICS_BRIDGE</li> <li>MQAT_CICS_VSE</li> <li>MQAT_DEFAULT</li> <li>MQAT_DOS</li> <li>MQAT_DQM</li> <li>MQAT_GUARDIAN</li> <li>MQAT_IMS</li> <li>MQAT_IMS_BRIDGE</li> <li>MQAT_JAVA</li> <li>MQAT_MVS</li> <li>MQAT_NO_CONTEXT</li> <li>MQAT_NOTES_AGENT</li> <li>MQAT_NSK</li> <li>MQAT_OS2</li> <li>MQAT_OS390</li> <li>MQAT_OS400</li> <li>MQAT_QMGR</li> <li>MQAT_UNIX</li> <li>MQAT_UNKNOWN</li> <li>MQAT_VMS</li> <li>MQAT_VOS</li> <li>MQAT_WINDOWS</li> <li>MQAT_WINDOWS_NT</li> <li>MQAT_XCF</li> </ul> <p><i>See note at the beginning of this table.</i></p> <p>To determine the circumstances under which the PutApplType field is used as input and output see the <i>IBM MQSeries Application Programming Reference Guide</i>.</p>
IMFMQI_MD_PUTAPPLNAME	Input/Output	<p>Optional. Specify up to a 28-character value for PutApplName. See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the PutApplName field is used as input and output.</p>
IMFMQI_MD_PUTDATE	Input/Output	<p>Optional. Specify an 8-character date stamp. See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the PutDate field is used as input and output.</p>
IMFMQI_MD_PUTTIME	Input/Output	<p>Optional. Specify an 8-character time stamp. See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the PutTime field is used as input and output.</p>

Variable name	Source	Notes
IMFMQI_MD_APPLORIGINDATA	Input/Output	Optional. Specify a 4-character value for ApplOriginData. See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the ApplOriginData field is used as input and output.
IMFMQI_MD_GROUPID	Input/Output	Optional. If the current MD is an MQMD_VERSION_2 MD, this variable may be set to a 24-byte GROUPID. Processed exactly as received from the PUT. No conversion of hexadecimal data is done. See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the GROUPID field is used as input and output.
IMFMQI_MD_MSGSEQNUMBER	Input/Output	Optional. If an MQMD_VERSION_2 MD is present, this variable may contain a decimal number in the range 1 to 999999999 (IBM default 1). See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the MsgSeqNumber field is used as input and output.
IMFMQI_MD_OFFSET	Input/Output	Optional. If an MQMD_VERSION_2 MD is present, this variable may contain a decimal number in the range 0 to 999999999 (IBM default 0). See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the Offset field is used as input and output.
IMFMQI_MD_MSGFLAGS	Input	Optional. If an MQMD_VERSION_2 MD is present, this variable may contain one or more of the following character strings delimited by blanks: MQMF_SEGMENTATION_INHIBITED MQMF_SEGMENTATION_ALLOWED MQMF_MSG_IN_GROUP MQMF_LAST_MSG_IN_GROUP MQMF_SEGMENT MQMF_LAST_SEGMENT MQMF_NONE (IBM default)  <i>See note at the beginning of this table.</i>
IMFMQI_MD_ORIGINALLENGTH	Input	Optional. Contains a decimal number, in the range 1 to 999999999, or MQOL_UNDEFINED.

The completion code is returned in the IMFMQCC and IMFCC variables. For additional information, see “How Completion and Reason Codes Are Returned” on page 156.

The reason code is returned in the IMFMQRC and IMFRC variables; contents of both variables are the same. Refer to the *IBM MQSeries Application Programming Reference Guide* for the reason code description.

## Example

---

```
/* REXX */
"IMFEXEC MQI CONN NAME(CSQ1)"           /* Connect to the queue manager */
IMFMQI_OD_OBJECTNAME = "TEST.QUEUE1"   /* Set queue name */
"IMFEXEC MQI OPEN OOPTS(MQOO_OUTPUT)"   /* Open the queue */
IMFMQI_PMO_OPTIONS = 'MQPMO_SYNCPOINT' /* Set Put options */
PUTDATA = 'MQPUT001 - TEST DATA ECB = MQPUTECB1 POST CODE = PUTECEB1'
"IMFEXEC MQI PUT BUFFER(PUTDATA)"       /* Put message on queue */
"IMFEXEC MQI COMMIT"                    /* Commit the work */
"IMFEXEC MQI CLOSE COPTS(MQCO_NONE)"    /* Close the queue */
"IMFEXEC MQI DISC"                       /* Disconnect from queue manager */
EXIT
```

---

This example highlights PUTting a message to a local queue. The following describes the variable values that are in effect during the PUT processing:

- Variable IMFHCONN contains the connection handle name.
- Variable IMFHOBJ contains the object handle for the queue.
- Variable IMFMQCC and IMFCC contain the completion code.
- Variable IMFMQRC and IMFRC contain the reason code.
- Variable IMFMQI\_REASON contains the constant (character) reason code.

## MQI PUT1

This statement puts a message to a queue without the need for an OPEN and CLOSE.

Command	Parameters
IMFEXEC MQI PUT1	HCONN(hconn) POPTS(options) BUFFLEN(bufferlength) BUFFER(buffer)

The following table describes the IMFEXEC MQI PUT1 parameters.

Parameter	Function	Notes
<p><b>Note:</b> The following list of constants may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i>.</p>		
HCONN	Provides the connection handle that represents the connection to the queue manager.	<p>This parameter:</p> <ul style="list-style-type: none"> <li>• Specifies a variable created in an earlier CONN command</li> <li>• Is not required</li> </ul> <p>If omitted, the connection handle value is the default variable IMFHCONN. The connection handle is returned by a previous IMFEXEC MQI CONN request and stored in the IMFHCONN variable or a user-specified variable. Either this parameter, specifying an existing variable that contains the connection handle, or a previously created IMFHCONN variable containing the connection handle is required.</p>

Parameter	Function	Notes
POPTS	Specifies the options that control the PUT1 action.	<p>This parameter is not required. If omitted, the PUT1 options are taken from the default variable IMFMQI_PMO_OPTIONS.</p> <p>The variable IMFMQI_PMO_OPTIONS is not required. Valid PUT1 options:</p> <ul style="list-style-type: none"> <li>MQPMO_SYNCPOINT</li> <li>MQPMO_NO_SYNCPOINT</li> <li>MQPMO_NO_CONTEXT</li> <li>MQPMO_DEFAULT_CONTEXT</li> <li>MQPMO_PASS_IDENTITY_CONTEXT</li> <li>MQPMO_PASS_ALL_CONTEXT</li> <li>MQPMO_SET_IDENTITY_CONTEXT</li> <li>MQPMO_SET_ALL_CONTEXT</li> <li>MQPMO_ALTERNATE_USER_AUTHORITY</li> <li>MQPMO_FAIL_IF QUIESCING</li> <li>MQPMO_NONE (IBM default)</li> </ul> <p><i>See note at the beginning of this table.</i></p> <p>When specifying multiple options, they must be enclosed within single quotation marks, for example:  POPTS('MQPMO_NO_SYNCPOINT  MQPMO_SET_ALL_CONTEXT')</p> <p>In addition to the above options, another AutoOPERATOR-provided option (REMOVE_DLH) is available.</p> <p>Using the REMOVE_DLH option, you can remove the Dead Letter Header from a message before PUTting it to another queue. REMOVE_DLH is added to the POPTS() keyword parameter of the IMFEXEC MQI PUT1 statement. This AutoOPERATOR option causes the message being PUT1 to be rebuilt by reloading the Encoding, CodedCharSetId and Format fields from the Dead Letter Header (DLH) into the Message Descriptor. This occurs before the PUT1 statement executes. The DLH, if present, is removed from the buffer containing the data.</p>

Parameter	Function	Notes
BUFFER	Specifies the variable that contains the message buffer data prior to issuing a PUT1.	This parameter is not required. If omitted, the variable IMFMQI_BUFFER must contain the message buffer data in order to complete the PUT1. Be aware that upon successful completion of the PUT1 statement, the original buffer variable may be different if you have specified structure variables prior to the PUT1. For example, you may want to build a Dead Letter message by changing the format variable, IMFMQI_MD_FORMAT, to MQFMT_DEAD_LETTER_HEADER and by setting variables starting with IMFMQI_DLH_ and ending with Dead Letter (DLH) field names (see Appendix D for lists of structure variables). In this example, AutoOPERATOR updates the buffer from the variables set before issuing the MQSeries PUT1 (see the section entitled “Creating Messages from Variables” on page 157 to learn how AutoOPERATOR creates message out of variables). The Dead Letter header is prefixed in front of the data that is currently in the buffer. AutoOPERATOR updates the variable specified for the buffer so that it contains the DLH structure before control is returned to the EXEC.
BUFFLEN	Specifies the length of the data used to create the message buffer data.	<p>This parameter</p> <ul style="list-style-type: none"> <li>• Is not required</li> <li>• Can specify a numeric value or the name of a variable that contains a numeric value</li> <li>• Can specify a value in the range of 0 to 32767, either specifically, or using a variable</li> </ul> <p>If this parameter is omitted, the variable IMFMQI_BUFFLEN is used to obtain the BUFFER length. If neither is set, the length of the data in the message buffer variable is used.</p>

The following table describes the most common input and output variables for the IMFEXEC MQI PUT1 statement. Any required variables must be set prior to the PUT. For structures other than the MD (message descriptor) and the OD (object descriptor), please refer to Appendix D, “EXECs Variables” on page 271. Variables used begin with “IMFMQI\_”, followed by the MQSeries structure, followed by the field name within the MQSeries structure, with each node delimited by an underscore. For definitions of field names and constants derived from

MQSeries structure field names and used in this table, see the *IBM MQSeries Application Programming Reference Guide*.

Variable name	Source	Notes
<p><b>Note:</b> The following list of constants may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i>. Also note that when specifying multiple options, they must be enclosed within single quotation marks. For example, POPTS('MQPMO_NO_SYNCPOINT MQPMO_SET_ALL_CONTEXT')</p>		
IMFMQI_OD_STRUCID	Input	Optional. The EXEC may or may not set this variable. If the EXEC does not set this variable, the MQSeries default value will be used unmodified by the EXEC Manager.
IMFMQI_OD_VERSION	Input	Optional. The EXEC may or may not set this variable. If the EXEC does not set this value, the MQSeries default will be used, unmodified by the EXEC Manager.
IMFMQI_OD_OBJECTTYPE	Input	Optional. The EXEC may or may not set this variable. If the EXEC does not set this value, the MQSeries default will be used, unmodified by the EXEC Manager.
IMFMQI_OD_OBJECTNAME	Input	Required. Contains the queue name to be opened; up to 48 characters can be specified.
IMFMQI_OD_OBJECTQMGRNAME	Input	Optional. Contains the 48-character object queue manager name. The EXEC Manager does not specify any default value when this variable is not set.
IMFMQI_OD_DYNAMICQNAME	Input	Optional. Contains the 48-character dynamic queue name when using a model queue. The EXEC Manager does not specify any default value when this variable is not set.
IMFMQI_OD_ALTERNATEUSERID	Input	Optional. Contains the 12-character alternate user identifier. The EXEC Manager does not specify any default value when this variable is not set.
IMFMQI_OD_RECSPRESENT	Input	Optional. The EXEC may or may not set this variable. If the EXEC does not set this value, the MQSeries default will be used, unmodified by the EXEC Manager.
IMFMQI_OD_KNOWNDESTCOUNT	Output	This variable contains 0.
IMFMQI_OD_UNKNOWNDESTCOUNT	Output	This variable contains 0.
IMFMQI_OD_INVALIDDESTCOUNT	Output	This variable contains 0.
IMFMQI_OD_OBJECTRECOFFSET	Input	This variable contains 0.
IMFMQI_OD_RESPONSERECOFFSET	Input	This variable contains 0.
IMFMQI_OD_OBJECTRECPTR	Input	This variable contains 0.

Variable name	Source	Notes
IMFMQI_OD_RESPONSERECPTR	Input	This variable contains 0.
IMFMQI_OD_ALTERNATESECURITYI D	Input	This variable exists but is null.
IMFMQI_OD_RESOLVEDQNAME	Output	This variable contains blanks.
IMFMQI_OD_RESOLVEDQMGRNAME	Output	This variable contains blanks.
IMFMQI_MD_STRUCID	Input	Optional. The EXEC may or may not set this variable. If the EXEC does not set this value, the MQSeries default will be used, unmodified by the EXEC Manager.
IMFMQI_MD_VERSION	Input	Optional. The EXEC may or may not set this variable. If the EXEC does not set this value, the MQSeries default will be used, unmodified by the EXEC Manager.
IMFMQI_MD_REPORT	Input	Optional. Specify one or more of the following delimited by blanks: MQRO_EXCEPTION MQRO_EXCEPTION_WITH_DATA MQRO_EXCEPTION_WITH_FULL_DATA MQRO_EXPIRATION MQRO_EXPIRATION_WITH_DATA MQRO_EXPIRATION_WITH_FULL_DATA MQRO_COA MQRO_COA_WITH_DATA MQRO_COA_WITH_FULL_DATA MQRO_COD MQRO_COD_WITH_DATA MQRO_COD_WITH_FULL_DATA MQRO_PAN MQRO_NAN MQRO_NEW_MSG_ID MQRO_PASS_MSG_ID MQRO_COPY_MSG_ID_TO_CORREL_ID MQRO_PASS_CORREL_ID MQRO_DEAD_LETTER_Q MQRO_DISCARD_MSG MQRO_NONE (IBM default)  <i>See note at the beginning of this table.</i>
IMFMQI_MD_MSGTYPE	Input	Optional. Specify one of the following: MQMT_DATAGRAM (IBM default) MQMT_REQUEST MQMT_REPLY MQMT_REPORT  <i>See note at the beginning of this table.</i>

Variable name	Source	Notes
IMFMQI_MD_EXPIRY	Input	Optional. Specify a decimal number, in the range 1 to 999999999, or MQEI_UNLIMITED (IBM default).
IMFMQI_MD_FEEDBACK	Input	Optional. Specify one of the following values for report messages: MQFB_NONE (IBM default) MQFB_COA MQFB_COD MQFB_EXPIRATION MQFB_PAN MQFB_NAN MQFB_QUIT  <i>See note at the beginning of this table.</i>  In addition, this variable can contain any reason code from the following sources: IMS-bridge feedback codes CICS-bridge feedback codes MQSeries reason codes User-specified numeric codes
IMFMQI_MD_ENCODING	Input	Optional. Specify one of the following:  MQENC_NATIVE (IBM default) or any decimal number up to 999999999.
IMFMQI_MD_CODEDCHARSETID	Input	Optional. Contains one of the following: MQCCSI_Q_MGR MQCCSI_EMBEDDED Or any decimal number up to 999999999.  <i>See note at the beginning of this table.</i>

Variable name	Source	Notes
IMFMQI_MD_FORMAT	Input	Optional. Specify one of the following: MQFMT_NONE (IBM default) MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PCF MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2  <i>See note at the beginning of this table.</i>
IMFMQI_MD_PRIORITY	Input	Optional. Specify a decimal number in the range 0 to 999999999 or MQPRI_PRIORITY_AS_Q_DEF (IBM default).
IMFMQI_MD_PERSISTENCE	Input	Optional. Specify one of the following: MQPER_PERSISTENT MQPER_NOT_PERSISTENT MQPER_PERSISTENCE_AS_Q_DEF (IBM default)  <i>See note at the beginning of this table.</i>
IMFMQI_MD_MSGID	Input/ Output	Optional. Specify up to a 32-byte MsgId. Processed exactly as received. No conversion of hexadecimal data is done. For input, the value MQMI_NONE is valid. After a successful PUT, the variable is set from the field in the message descriptor.
IMFMQI_MD_CORRELID	Input/ Output	Optional. Specify up to a 32-byte CORRELID. Processed exactly as received. No conversion of hexadecimal data is done. After a successful PUT, the variable is set from the field in the message descriptor. For input, the following values are valid: MQCI_NONE (IBM default) MQCI_NEW_SESSION
IMFMQI_MD_BACKOUTCOUNT	N/A	This variable is not used for PUT.
IMFMQI_MD_REPLYTOQ	Input	Optional. Specify up to a 48-character name of the ReplyToQ. The IBM default value is 48 blanks.

Variable name	Source	Notes
IMFMQI_MD_REPLYTOQMGR	Input	Optional. Specify up to a 48-character name of the ReplyToQMgr. The IBM default value is 48 blanks.
IMFMQI_MD_USERIDENTIFIER	Input/ Output	Optional. Specify up to a 12-character UserIdentifier. See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the UserIdentifier field is used as input and output.
IMFMQI_MD_ACCOUNTINGTOKEN	Output	Optional. Contains MQACT_NONE (IBM default) or up to a 32-byte AccountingToken. Processed exactly as received. No conversion of hexadecimal data is done.
IMFMQI_MD_APPLIDENTITYDATA	Input/ Output	Optional. Contains up to a 32-character value for ApplIdentityData. See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the ApplIdentityData field is used as input and output.

Variable name	Source	Notes
IMFMQI_MD_PUTAPPLTYPE	Input/ Output	<p>Optional. Specify one of the following standard types or a numeric user-defined type:</p> <ul style="list-style-type: none"> <li>MQAT_AIX</li> <li>MQAT_BROKER</li> <li>MQAT_CICS</li> <li>MQAT_CICS_BRIDGE</li> <li>MQAT_CICS_VSE</li> <li>MQAT_DEFAULT</li> <li>MQAT_DOS</li> <li>MQAT_DQM</li> <li>MQAT_GUARDIAN</li> <li>MQAT_IMS</li> <li>MQAT_IMS_BRIDGE</li> <li>MQAT_JAVA</li> <li>MQAT_MVS</li> <li>MQAT_NO_CONTEXT</li> <li>MQAT_NOTES_AGENT</li> <li>MQAT_NSK</li> <li>MQAT_OS2</li> <li>MQAT_OS390</li> <li>MQAT_OS400</li> <li>MQAT_QMGR</li> <li>MQAT_UNKNOWN</li> <li>MQAT_UNIX</li> <li>MQAT_VMS</li> <li>MQAT_VOS</li> <li>MQAT_WINDOWS</li> <li>MQAT_WINDOWS_NT</li> <li>MQAT_XCF</li> </ul> <p><i>See note at the beginning of this table.</i></p> <p>See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the PutApplType field is used as input and output.</p>
IMFMQI_MD_PUTAPPLNAME	Input/ Output	<p>Optional. Specify up to a 28-character value for PutApplName. See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the PutApplName field is used as input and output.</p>
IMFMQI_MD_PUTDATE	Input/ Output	<p>Optional. Specify an 8-character date stamp. See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the PutDate field is used as input and output.</p>
IMFMQI_MD_PUTTIME	Input/ Output	<p>Optional. Specify an 8-character time stamp. See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the PutTime field is used as input and output.</p>

Variable name	Source	Notes
IMFMQI_MD_APPLORIGINDATA	Input/ Output	Optional. Specify a 4-character value for ApplOriginData. See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the ApplOriginData field is used as input and output.
IMFMQI_MD_GROUPID	Input/ Output	Optional. If the current MD is an MQMD_VERSION_2 MD, this variable may be set to a 24-byte GROUPID. Processed exactly as received from the PUT1. No conversion of hexadecimal data is done. See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the Grouped field is used as input and output.
IMFMQI_MD_MSGSEQNUMBER	Input/ Output	Optional. If an MQMD_VERSION_2 MD is present, this variable may contain a decimal number in the range 1 to 999999999 (IBM default 1). See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the MsgSeqNumber field is used as input and output.
IMFMQI_MD_OFFSET	Input/ Output	Optional. If an MQMD_VERSION_2 MD is present, this variable may contain a decimal number in the range 0 to 999999999 (IBM default 0). See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the Offset field is used as input and output.
IMFMQI_MD_MSGFLAGS	Input	Optional. If an MQMD_VERSION_2 MD is present, this variable may contain one or more of the following character strings delimited by blanks:  MQMF_SEGMENTATION_INHIBITED MQMF_SEGMENTATION_ALLOWED MQMF_MSG_IN_GROUP MQMF_LAST_MSG_IN_GROUP MQMF_SEGMENT MQMF_LAST_SEGMENT MQMF_NONE (IBM default)  <i>See note at the beginning of this table.</i>
IMFMQI_MD_ORIGINALLENGTH	Input	Optional. Contains a decimal number, in the range 1 to 999999999, or MQOL_UNDEFINED.

Variable name	Source	Notes
IMFMQI_STRUCTURES	Output	Contains a blank-delimited list of the MQSeries structures used in the creation of the message. For example, in the case of a Dead Letter message, the contents would contain at least, 'MD OD DLH'.
IMFMQI_OFFSETS	Output	<p>Contains a blank-delimited list of the offsets of the structures used in the creation of the message (as well as the offset of the application data) from the beginning of the message buffer. Using this variable you can access application data beyond the structures in the message buffer after issuing the PUT1. The number of entries in the list is always one more than the number of structures participating in the operation, including MD, which participates in the operation but is not part of the message buffer. This is because an entry for the offset of the application data is included in the list. The value 'NONE' is always used for a structure that does not exist in the buffer but is used in creating the message for PUT1, such as 'MD'. For example, if the message was a dead letter message and contained application data, the variable may look like this:</p> <p>'NONE NONE 0 172'</p> <p>In the above example, NONE is specified for the OD, 0 for the offset of the Dead Letter Header and 172 is the offset of the application data.</p>

The completion code is returned in the IMFMQCC and IMFCC variables. For additional information, see "How Completion and Reason Codes Are Returned" on page 156.

The reason code is returned in the IMFMQRC and IMFRC variables; contents of both variables are the same. Refer to the *IBM MQSeries Application Programming Reference Guide* for the reason code description.

## Example

---

```

/* REXX */
"IMFEXEC MQI CONN NAME(CSBD)"                /* Connect to the queue manager */
IMFMQI_OD_OBJECTNAME = "JOHNB.QUEUE1"       /* Set queue name */
IMFMQI_PMO_OPTIONS = 'MQPMO_NO_SYNCPOINT'   /* Set Put options */
PUTDATA = 'MQEXAMPL - TEST DATA ECB = MQPUTECB1 POST CODE = PUTECH1'
"IMFEXEC MQI PUT1 BUFFER(PUT DATA)"        /* Put message on queue */
"IMFEXEC MQI DISC"                          /* Disconnect from queue manager */
EXIT

```

---

This example highlights the PUTting of a message to a local queue. The following describes the variable values that are in effect during the PUT processing:

- Variable IMFHCONN contains the connection handle name.
- Variable IMFHOBJ contains the object handle for the queue.

- Variable IMFMQCC and IMFCC contain the completion code.
- Variable IMFMQRC and IMFRC contain the reason code.
- Variable IMFMQI\_REASON contains the constant (character) reason code.

## CMD (Issue IMFEXEC Command to MQSeries with Response)

This format issues commands to the MQSeries command server. A response is returned to the issuing EXEC.

Command	Parameters
IMFEXEC CMD	'MQ command' TYPE(mqs) LM(queue manager name) RESPONSE(msgid1,msgid2,...) WAIT(30 nnnn) PCF(yes no) QM(queue manager name) CQ(command queue name)

The following table describes the parameters:

Parameter	Function	Notes
MQ command	A valid MQ command	This parameter <ul style="list-style-type: none"> <li>• Is required</li> <li>• Has no default value</li> </ul> <p>The MQ command can be up to 256 characters long.</p>
TYPE	Type of IMFEXEC CMD	This parameter <ul style="list-style-type: none"> <li>• Is required</li> <li>• Uses a default of MVS</li> </ul> <p>You must code TYPE(mqs) in order to issue the command to an MQ command server.</p>
LM	Local MQ queue manager name	This parameter <ul style="list-style-type: none"> <li>• Is not required</li> <li>• Uses the default MQ manager name</li> </ul> <p>You can issue MQ commands to any MQ queue manager that is on the same OS/390 system as AutoOPERATOR. If this parameter is not specified, the default queue manager will be used. (Refer to the IBM MQ documentation for information about how the default MQ queue manager is determined.)</p>

Parameter	Function	Notes
RESPONSE	Allows you to specify from 1 to 8 message IDs	<p>This parameter</p> <ul style="list-style-type: none"> <li>• Is not required</li> <li>• Has no default value</li> </ul> <p>Each message ID may contain up to 16 characters including wildcard characters. An example message ID is 'CSQN205I'. If this parameter is coded, only command response messages that match at least one of the message IDs will be returned to the EXEC. Do not code this parameter if you want to receive all the command response messages.</p>
WAIT	A number from 5 to 9999 (in seconds)	<p>This parameter</p> <ul style="list-style-type: none"> <li>• Is not required</li> <li>• Uses a default value of 30</li> </ul> <p>Number of seconds that the IMFEXEC CMD waits for MQ command response messages. MAINVIEW AutoOPERATOR for MQSeries need not always wait this number of seconds; in fact, the EXEC will continue processing as soon as all MQ command response messages have been received. Unless you are experiencing problems with a slow MQ command server, do not code this parameter.</p>
PCF	With this parameter, you can specify whether to issue commands in PCF format. Valid values are YES or NO.	<p>This parameter</p> <ul style="list-style-type: none"> <li>• Is not required</li> <li>• Uses a default value of no</li> </ul> <p>When PCF=YES, AutoOPERATOR will issue commands in PCF (Programmable Command Format) format. You need to code this parameter only when issuing commands to a non-OS/390 MQ queue manager.</p>
QM	With this parameter, you can specify the name of a MQ queue manager to which you want to send a command.	<p>This parameter</p> <ul style="list-style-type: none"> <li>• Is not required</li> <li>• Has no default value</li> </ul> <p>Supports 1 to 48 characters. This queue manager can be OS/390 or non-OS/390.</p> <p><b>Note:</b> This field is case sensitive (others are automatically upper cased).</p> <p>AutoOPERATOR will issue the command through the local MQ queue manager specified (or defaulted) by the NAME( ) parameter.</p>

Parameter	Function	Notes
CQ	The name of the system command queue at the remote MQ queue manager	<p>This parameter:</p> <ul style="list-style-type: none"> <li>• Is not required</li> <li>• Has no default value</li> </ul> <p>Specify the name of the system command queue (1 to 48 characters) of the remote MQ queue manager. For example, the queue SYSTEM.ADMIN.COMMAND.QUEUE is used for many non-OS/390 MQ queue managers.</p>
<p>For more information about the parameters described in this table, refer to the <i>IBM MQSeries Application Programming Reference</i>.</p>		

In addition, note the following:

- The number of MQ response messages is returned in the TSO variable IMFNOL.
- The TSO variables MQLNxxxx contain the data received (for example, MQLN1 contains the first response message, and MQLN2 contains the second, and so on). Up to 9999 response messages are supported. The maximum size of each MQLNxxxx variable (response message) is 13000 characters.
- The CSQ9022I message (for example, CSQ9022I @MQS1CPF CSQMDRTS 'DISPLAY QUEUE' NORMAL COMPLETION) contains embedded single quotation marks and non-displayable characters, which can cause problems when using IMFEXEC MSG and some other commands. One solution to this problem is to translate the single quotation marks and invalid characters to other characters using the REXX TRANSLATE function. Refer to “Examples – Retrieving Responses to an MQ Command” on page 213 for an illustration.
- AutoOPERATOR creates a permanent dynamic queue, 'BBOMVAO.EXEC.REPLY.aosid.qmgrssid'. This queue is modeled after the IBM-supplied SYSTEM.COMMAND.REPLY.MODEL, which is defined by CSQ4INP2 with the SHARE attribute. AutoOPERATOR requires this Share attribute to allow multiple EXECs to concurrently retrieve response messages from MQ.
- The first OS/390 MQ command response message (CSQN205I) carries the count of the MQ command response messages as well as the return and reason codes for the command. Since the number of response messages is known, there is no need to specify it.
- AutoOPERATOR destructively reads all MQ command responses (for example, the number of messages specified in CSQN205I) for this IMFEXEC CMD before returning control to the EXEC.
- If MQ detects an error with the command, IMFEXEC CMD fails with IMFCC=12. If you get this failure, retrieve all the MQ response messages and inspect the CSQN205I message carrying the MQ return and reason codes. In addition, inspect the MQ messages that follow the CSQN205I message. These messages provide more descriptions about the error.
- AutoOPERATOR inserts the name of the user into the mqmd\_UserIdentifier field of the message containing the command. It inserts the TS (Terminal Session) user for user initiated EXECs, and the SSID (PAS-ID) for automatically invoked (rule, timer, and so on) EXECs. You must ensure that this user ID has appropriate authority to issue the command

## CMD TYPE(MQS)

at the destination MQ queue manager, particularly when it is a non-OS/390 MQ queue manager.

- AutoOPERATOR issues remote commands through the local MQ queue manager specified (or defaulted) in the NAME( ) parameter. The remote MQ queue manager must be connected to this local MQ queue manager. In addition, one of the following requirements must be met at each MQ queue manager:
  - There is a queue manager alias to the other MQ queue manager.
  - There is a transmission queue with the name of the other MQ queue manager.

One of the following completion code is returned in the IMFCC variable:

<b>Value</b>	<b>Description</b>
0	Successful completion
4	Time out occurred while waiting for MQ command response messages. IMFNOL contains the number of response message received so far.
8	MQSeries returned error
12	Buffer too small
16	Syntax error
20	Variable Processing routine not found

## Examples – Retrieving Responses to an MQ Command

### Example 1– Retrieving All Responses to an MQ Command

---

```

/* REXX */
/* Example to retrieve all responses to an MQ command*/
PARSE ARG EXNAME.
"IMFEXEC MSG '."EXNAME "EID="IMFEID "STARTED' "
"IMFEXEC CMD ' DISPLAY QUEUE(SYSTEM CHANNEL. *)' TYPE(MQS) "
"IMFEXEC MSG '.IMFNOL="IMFNOL "CC="IMFCC "IMFEXEC CMD MQ' "
DO i=1 TO IMFNOL
bad_chars = xrange('00' x, '41' x);
bad_chars = bad_chars""""";
v1 = value('MQLN'i);
clean_data = translate(v1, ' ', bad_chars)
  "IMFEXEC MSG '.MQLN"i "LENGTH=length(v1)"" "
  "IMFEXEC MSG '.MQLN"i "="clean_data"" "
END

"IMFEXEC MSG '."EXNAME "EID="IMFEID "CC="IMFCC "RC="IMFRC "ENDED' "

```

---

### Example 2 – Retrieving Selected Responses to an MQ Command

---

```

/* REXX */
/* Example to retrieve selected responses to an MQ command.*/
PARSE ARG EXNAME.
"IMFEXEC MSG '."EXNAME "EID="IMFEID "STARTED' "
/* example of an IMFEXEC command that continues on the next line */
"IMFEXEC CMD ' DISPLAY QUEUE(SYSTEM CHANNEL. *)' TYPE(MQS) ,
LM(MQS1) RESP(CSQN205I, CSQ9022I) "
"IMFEXEC MSG '.IMFNOL="IMFNOL "CC="IMFCC "IMFEXEC CMD MQ' "
DO i=1 TO IMFNOL
bad_chars = xrange('00' x, '41' x);
bad_chars = bad_chars""""";
v1 = value('MQLN'i);
clean_data = translate(v1, ' ', bad_chars)
  "IMFEXEC MSG '.MQLN"i "LENGTH=length(v1)"" "
  "IMFEXEC MSG '.MQLN"i "="clean_data"" "
END

"IMFEXEC MSG '."EXNAME "EID="IMFEID "CC="IMFCC "RC="IMFRC "ENDED' "

```

---

**COPY MQI**

Using this statement, you can copy the contents of a set (or sets) of variables created from a structure (or multiple structures), found in an MQSeries message, to another set of variables for the purpose of temporarily saving. For example, suppose you want to GET a message from one queue, save the message descriptor values for later, GET another message and inspect some portion of that message and, based on values discovered from the second GET, forward the first message to another queue. Using the COPY MQI command, you can do this by saving the first set of structure values. Later, using another COPY command, you can copy them back to the IMFMQI\_ variables and, optionally, PUT the message to another queue.

The IMFEXEC COPY MQI statement will copy entire structures based on input parameters. You can copy a single structure, for example, the MD, or multiple structures, for example, the MD and the MDE or ALL structures that exist in the message. You can also use the COPY MQI statement to nullify all variables in a structure or more than one structure.

Command	Parameters
IMFEXEC COPY MQI	STRTYPE(structure Ids) STRFROM(input prefix) STRTO(output prefix)

This table describes the IMFEXEC COPY MQI parameters.

Parameters	Function	Notes
STRTYPE	Identifies the structure IDs to copy.	<p>This parameter is required.</p> <p>Possible input values are as follows:</p> <ul style="list-style-type: none"> <li>MD (Message Descriptor)</li> <li>MDE (Message Descriptor Extension)</li> <li>OD (Object Descriptor)</li> <li>DLH (Dead Letter Header)</li> <li>CIH (CICS Information Header)</li> <li>IIH (IMS Information Header)</li> <li>XQH (Transmission Queue Header)</li> <li>TM (Trigger Monitor structure)</li> <li>TMC (Trigger Monitor structure Character format)</li> <li>WIH (Work Information Header)</li> <li>RMH (Reference Message Header)</li> <li>CFH (Command Format Header)</li> <li>RFH (Rules and Formatting Header)</li> <li>RFH2 (Rules and Formatting Header – version 2)</li> <li>EVEN (Event variables)</li> <li>ALL (all known existing structures)</li> </ul> <p>Multiple structures can be specified, delimited by blanks or commas. The variables IMFMQI_STRUCTURES or IMFQ_STRUCTURES can be used in place of individual entries. The IMFMQI_STRUCTURES variable is returned from an IMFEXEC MQI GET statement and contains a blank-delimited list of structures that exist within the message. The IMFQ_STRUCTURES variable is passed to an EXEC from an MQSeries rule and contains a blank-delimited list of structures that exist within the message.</p> <p><b>Note:</b> The variable IMFMQI_BUFFER (by default) contains the contents of the message buffer upon a successful GET. If this variable or another user-specified variable containing the message buffer exists during a COPY MQI operation, the EXEC may need to preserve the contents by issuing an assignment statement to set another variable to the value of the buffer variable prior to issuing another IMFEXEC MQI GET statement. For example, in REXX the statement might look like this: MYNEW_BUFFER = IMFMQI_BUFFER</p>

Parameters	Function	Notes
STRFROM	Specifies the prefix of the variables to copy from.	This parameter is not required.  If omitted, the default variable prefix is IMFMQI_. The value specified must contain characters acceptable to CLIST and REXX. To set a structure of variables to nulls, specify STRFROM(NULLS). The allowable input prefix length is from 1 to 24 characters.
STRTO	Specifies the prefix of the variables to copy to	This parameter is required. The length of the value specified must be from 1 to 24 characters and must consist of characters acceptable to CLIST and REXX.  <b>Note:</b> Do not use STRTO value of BBIJRNL_ because AutoOPERATOR reserves this for internal processing.

The completion code is returned in the IMFMQCC and IMFCC variables. For additional information, see “How Completion and Reason Codes Are Returned” on page 156.

The reason code is returned in the IMFMQRC and IMFRC variables; contents of both variables are the same. Refer to the *IBM MQSeries Application Programming Reference Guide* for the reason code description.

### Example

---

```

/* REXX */
"IMFEXEC MQI CONN NAME(CSBD)" /* Connect to the queue manager*/
IMFMQI_OD_OBJECTNAME = 'JOHNB.QUEUE4' /* Set queue name*/
"IMFEXEC MQI OPEN OOPTS(MQOO_BROWSE)" /* Open the queue*/
IMFMQI_GMO_OPTIONS = 'MQGMO_BROWSE_NEXT' /* Set Get options*/
/* Browse msg on queue*/
"IMFEXEC MQI GET BUFFER(GETDATA)"
"IMFEXEC COPY MQI STRTYPE("IMFMQI_STRUCTURES")
                STRFROM(IMFMQI_) STRTO(IMFMQX_)") /* Make a copy */
"IMFEXEC MQI CLOSE COPTS(MQCO_NONE)" /* Close the queue*/
"IMFEXEC MQI DISC"/* Disconnect from queue manager*/
EXIT

```

---

This example highlights the copying of a message to a local queue. The following describes the variable values that are in effect during the COPY processing:

- Variable IMFHCONN contains the connection handle name.
- Variable IMFHOBJ contains the object handle for the queue.
- Variable IMFMQCC and IMFCC contain the completion code.
- Variable IMFMQRC and IMFRC contain the reason code.
- Variable IMFMQI\_REASON contains the constant (character) reason code.

## DISPLAY MQI

Using this statement you can display the contents of a set (or sets) of variables created from a structure (or multiple structures) found in an MQSeries message, in the BBI journal. The variable contents are displayed in character format and in hexadecimal format. This information could be useful for debugging when variable comparisons are not matching and you are not able to determine why.

Command	Parameters
IMFEXEC DISPLAY MQI	STRTYPE(structure Ids) PREFIX(input prefix)

This table describes the IMFEXEC DISPLAY MQI parameters.

Parameters	Function	Notes
STRTYPE	Identifies the structure variables to display.	<p>This parameter is required.</p> <p>Possible input values are:</p> <ul style="list-style-type: none"> <li>MD (Message Descriptor)</li> <li>MDE (Message Descriptor Extension)</li> <li>OD (Object Descriptor)</li> <li>DLH (Dead Letter Header)</li> <li>CIH (CICS Information Header)</li> <li>IIH (IMS Information Header)</li> <li>XQH (Transmission Queue Header)</li> <li>TM (Trigger Monitor structure)</li> <li>TMC (Trigger Monitor structure character format)</li> <li>WIH (Work Information Header)</li> <li>RMH (Reference Message Header)</li> <li>CFH (Command Format Header)</li> <li>RFH (Rules and Formatting Header)</li> <li>RFH2 (Rules and Formatting Header – version 2)</li> <li>EVEN (Event variables)</li> <li>ALL (all known existing structures)</li> </ul> <p>Multiple structures can be specified, delimited by blanks or commas. The variables IMFMQI_STRUCTURES or IMFQ_STRUCTURES can be used in place of individual entries. The IMFMQI_STRUCTURES variable is returned from an IMFEXEC MQI GET statement and contains a blank-delimited list of structures that exist within the message. The IMFQ_STRUCTURES variable is passed to an EXEC from an MQSeries rule and contains a blank-delimited list of structures that exist within the message.</p>

## CMD TYPE(MQS)

Parameters	Function	Notes
PREFIX	Specifies the prefix of the variables to display.	This parameter is not required.  If omitted, the default variable prefix is IMFMQI_.

The completion code is returned in the IMFCC variable. For additional information, see “How Completion and Reason Codes Are Returned” on page 156.

The reason code is returned in the IMFMQRC and IMFRC variables; contents of both variables are the same. Refer to the *IBM MQSeries Application Programming Reference Guide* for the reason code description.

### Example

---

```
/* REXX */
"IMFEXEC MQI CONN NAME(CSBD)"           /* Connect to the queue manager
IMFMQI_OD_OBJECTNAME = "JOHNB.QUEUE4"  /* Set queue name*/
"IMFEXEC MQI OPEN OOPTS(MQOO_BROWSE)"   /* Open the queue*/
IMFMQI_GMO_OPTIONS = 'MQGMO_BROWSE_NEXT' /* Set Get options*/
/* Browse msg on queue*/
"IMFEXEC MQI GET BUFFER(GETDATA)"
"IMFEXEC DISPLAY MQI STRTYPE("IMFMQI_STRUCTURES")
    PREFIX(IMFMQI_)"
"IMFEXEC MQI CLOSE COPTS(MQCO_NONE)"    /* Close the queue*/
"IMFEXEC MQI DISC"                      /* Disconnect from queue manager
EXIT
```

---

This example highlights the DISPLAYing of a message to a local queue. The following describes the variable values that are in effect during the DISPLY processing:

- Variable IMFHCONN contains the connection handle name.
- Variable IMFHOBJ contains the object handle for the queue.
- Variable IMFMQCC and IMFCC contain the completion code.
- Variable IMFMQRC and IMFRC contain the reason code.
- Variable IMFMQI\_REASON contains the constant (character) reason code.

---

## Appendix A. Diagnosing AutoOPERATOR for MQSeries Errors

This appendix provides resources and suggestions to help you resolve problems that you may encounter while installing or using AutoOPERATOR for MQSeries.

---

### Where You Can Find Additional Information

The following books contain additional information that you may find useful while resolving error messages or deciphering codes:

- *IBM MQSeries for OS/390 Messages and Codes*, SC33-0819

Use this book to look up messages or codes that you might receive from MQSeries.

- *IBM MQSeries Programmable System Management*, SC33-1482

Use this book to read about facilities available in MQSeries products for

- Monitoring instrumentation events in a network of connected systems that use IBM MQSeries products in different operating system environments
- Looking up descriptions of queue manager event messages
- *IBM MQSeries for OS/390 Problem Determination Guide*, GC33-0808

Use this book to help you determine the causes of MQSeries for OS/390 problems.

## Which Tools Are Available to Diagnose AutoOPERATOR for MQSeries Problems

Figure 42 shows the interaction between AutoOPERATOR for MQSeries and MQSeries and indicates the appropriate diagnostic tools to deal with each area.

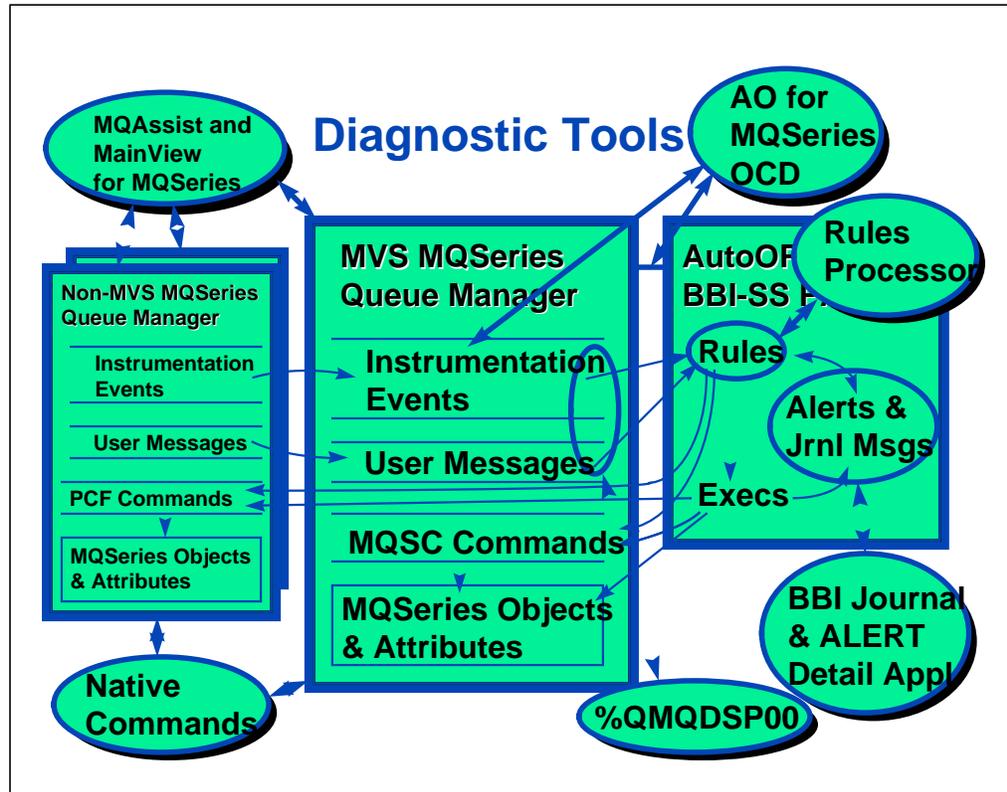


Figure 42. Interactions between AutoOPERATOR for MQSeries and MQSeries Queue Managers

This figure shows AutoOPERATOR for MQSeries running in a BBI-SS PAS connected directly to an OS/390 MQSeries queue manager. The OS/390 queue manager is connected to non-OS/390 queue managers. For AutoOPERATOR for MQSeries to detect the arrival of event messages and non-event messages from the non-OS/390 queue managers, the local event queues on the non-OS/390 queue managers have been deleted and redefined as remote queues that actually reside on the OS/390 queue manager. Therefore, AutoOPERATOR for MQSeries can listen for both OS/390 queue manager events and non-OS/390 queue manager events.

Some of the important dynamics of Figure 42 on page 220 are as follows:

- All non-OS/390 instrumentation events must travel through an OS/390 queue manager to reach AutoOPERATOR for MQSeries.
- OS/390 (and, indirectly non-OS/390) queue manager instrumentation events and non-event messages cause AutoOPERATOR Rules to be fired. A Rule's actions can automate the event and additionally issue an ALERT to notify operators of important situations. A Rule can also schedule an EXEC to take additional automation actions.
- AutoOPERATOR Rules and EXECs can issue MQSC commands for OS/390 queue managers (or PCF commands for non-OS/390 queue managers) to modify MQSeries objects and attributes.
- Both Rules and EXECs can write to the BBI Journal and the ALERT Detail application, which can then be referenced for verifying AutoOPERATOR activities, such as firing a Rule.

## Diagnostic Checklist

When AutoOPERATOR for MQSeries automation does not appear to be functioning correctly, specific diagnostic tools are available to monitor particular areas. These are shown in the circled areas of Figure 42 on page 220. The following checklist is the most systematic way to use the diagnostic tools to determine the solutions for AutoOPERATOR for MQSeries problems:

1. Look up any messages or codes.

Examine the BBI Journal and the ALERT Detail application for error messages or other indications that AutoOPERATOR is not working correctly. For example, if an expected action did not occur, the BBI Journal may display whether or not the Rule fired that would have caused the action.

Refer to the *IBM MQSeries for OS/390 Messages and Codes* for more information about specific OS/390 messages. Depending on the platform, refer to the appropriate messages and codes book for non-OS/390 messages.

2. Verify that QAO is active.

Issue the BBI command `.D A` to verify QAO is active.

3. Verify that the Rule Set and Rules are enabled.

Use the Rules Processor Automation Control panels to verify that the correct Rule Set and Rules are enabled and to alter the selection criteria or actions taken by a given Rule.

If the selection criteria for the Rule are too specific, the Rule may not fire for all the occasions that it is expected to. You can determine if this is the problem by relaxing the selection criteria in the control panels.

For more information, refer to the question “I have written an MQS Rule for an event that is not firing. What can I do?” on page 229.

4. Verify that the queue manager is connected.

Examine the AutoOPERATOR for MQSeries Workstation panel to determine if the queue manager is connected and verify the status of the instrumentation events.

For more information, refer to Question 2 on page 225.

5. Determine which queues are eligible for automation.

Invoke the QMQDSP00 EXEC to determine which queues are eligible for automation. This EXEC lists all the queues defined in a specific queue manager and which are currently enabled for automation.

For more information, refer to the question “How can I find out why AutoOPERATOR is unable to monitor a particular queue?” on page 227.

6. Verify that the OS/390 queue manager can be monitored.

Use MAINVIEW for MQSeries to verify that the OS/390 queue manager is usable and has the correct attributes for automation.

7. Verify that the non-OS/390 queue manager can be monitored.

Use MAINVIEW for MQSeries or PATROL for MQ-Administrator to validate that the non-OS/390 queue manager is usable.

Refer to Figure 42 on page 220 to see how information flows to Rules and how commands flow to queue managers. Refer to “Step 3. Defining Connectivity” on page 22 to see how you must set up the paths for these flows to take place.

---

## Frequently Asked Questions

Below is a list of questions frequently asked when verifying the installation and performance of AutoOPERATOR for MQSeries. The diagnostic checklist shown on page 222 is applied to these problems.

- **After running the IVP EXEC QMQIVP00, how can I find out what is wrong with my installation of AutoOPERATOR for MQSeries?**

- 1a. Review the BBI-SS PAS’ Start messages in the BBI Journal (or in the joblog) and look for the QA1113E message:

```
QA1113E UNABLE TO INITIALIZE AO for MQ TASK
```

The QA1113E error message occurs when you bring up a queue manager after the BBI-SS PAS is started and the system is unable to initialize the AutoOPERATOR for MQSeries task. If you receive this error, you may need to contact BMC Software Customer Support for assistance.

If this message is not in the BBI Journal or the joblog, the BBI-SS PAS was started correctly.

- 1b. Review the joblog messages for any error messages involving the queue manager and its channel initiator.
- 1c. Verify that the MQSeries APF-authorized libraries are added to the STEPLIB DD CARD properly. If they are not added correctly or are missing, you may receive error message QA1114E. The error message and reason are displayed below.

```
QA1114E MQSERIES STEPLIB MISSING/INVALID
REASON  AO for MQ tried to initialize, but failed because the STEPLIB
did not specify a valid MQSeries load library.
SYSTEM  AO for MQ will not initialize.
ACTION
USER    Check the JCL for the BBI/SS to make sure the STEPLIB DD
ACTION  statement contains the MQSeries Authlib.
```

If the MQSeries libraries are missing from the STEPLIB DD concatenation, add them. The MQSeries libraries must be at or above the level of the OS/390 queue managers that are going to be automated. Also review “Adding MQSeries APF-Authorized Libraries to the STEPLIB DD Card” on page 17 in Chapter 2.

2. Verify that the QAO product option key is properly installed and active. For information about how the QAO key is installed, refer to the section “Specifying Product Option Password Keys” on page 15 in Chapter 2.

If you have already followed the instructions, perform the following steps to ensure that the product was installed properly:

- In the BBI Journal, issue the BBI command `.D A` (display active).

This command shows all currently active products in the BBI-SS PAS; for example:

```
AA0101I  AAO VERSION 6.2.0 STARTED ON 28-JAN-2002 AT 09:53:11
AA0102I      CAO - ACTIVE
AA0103I      IAO - ACTIVE
AA0104I      MAO - ACTIVE
AA0106I      QAO - ACTIVE
```

If QAO is not listed as active, AutoOPERATOR for MQSeries either has not been installed or has been installed incorrectly. For information about installing the product, refer to Chapter 2, “Implementing MAINVIEW AutoOPERATOR for MQSeries” on page 9.

- In the BBI Journal, issue the BBI command `.D KEYS` (display keys).

This command shows all currently valid product option keys; for example:

```
CF8701I  COPY PROTECTION KEY STATUS:
CF8702I  PRD      CPUID   EXP     SYS      STATUS
CF8703I  -----
CF8704I  MA0?9672-15- *1317-99365- C4B0      VALID
CF8704I  MA0?9672-08- *3481-99365- 4CA4      CPU- ERR
CF8704I  QA0-9672-15- 11317-99365- D0A5      VALID
CF8704I  QA0-9672-08- 13481-99365- D2DF      CPU- ERR
```

If you have typed an invalid key, as described in “Specifying Product Option Password Keys” on page 15, or you do not have the key, contact BMC Software Customer Support to obtain the key.

3. Verifying that the Rule Set and Rules are enabled is not relevant in this situation.
4. Verify the connection to the queue manager by using the AutoOPERATOR for MQSeries Workstation panel and issue the BBI command `.RESET MQ xx`, where `xx` represents the two-character suffix of the `AAOMQLxx` member, to connect to the queue manager.

If the queue manager is not connected, follow the instructions in Chapter 2 to connect it.

For information about how to use the Workstation display, refer to Chapter 5, “Viewing AutoOPERATOR for MQSeries Automation Statistics” on page 115.

5. You do not need to invoke the `QMQRSP00 EXEC` to determine which queues are available, because this information is not relevant.
6. Use MAINVIEW for MQSeries or OS/390 commands to verify that the OS/390 queue manager is usable. Refer to the *IBM MQSeries for OS/390 Problem Determination Guide* for more information about any problems you may encounter.

7. Use MAINVIEW for MQSeries or PATROL for MQ-Administrator to verify that the event queues are remote and working and the channels, transmission queue, and queue manager alias are working.

- **How can I find out why the queue manager is not connected?**

- 1a. Review the BBI-SS PAS' Start messages in the BBI Journal (or in the joblog) and look for the QA1113E message:

```
QA1113E UNABLE TO INITIALIZE AO for MQ TASK
```

The QA1113E error message occurs when you bring up a queue manager after the BBI-SS PAS is started and the system is unable to initialize the AutoOPERATOR for MQSeries task. If you receive this error, you may need to contact BMC Software Customer Support for assistance.

If this message is not in the BBI Journal or the joblog, the BBI-SS PAS was started correctly.

- 1b. Review the joblog messages for any error messages involving the queue manager and its channel initiator.
- 1c. Verify that the MQSeries APF-authorized libraries are added to the STEPLIB DD CARD properly. If they are not added correctly or are missing, you may receive error message QA1114E. The error message and reason are displayed below.

```
QA1114E MQSERIES STEPLIB MISSING/INVALID
REASON AO for MQ tried to initialize, but failed because the STEPLIB
did not specify a valid MQSeries load library.
SYSTEM AO for MQ will not initialize.
ACTION
USER Check the JCL for the BBI/SS to make sure the STEPLIB DD
ACTION statement contains the MQSeries Authlib.
```

If the MQSeries libraries are missing from the STEPLIB DD concatenation, add them. The MQSeries libraries must be at or above the level of the OS/390 queue managers that are going to be automated. Also review the “Adding MQSeries APF-Authorized Libraries to the STEPLIB DD Card” on page 17.

2. Verify that the QAO product option key is properly installed and active. For information about how the QAO key is installed, refer to the section “Specifying Product Option Password Keys” on page 15.

If you have already followed the instructions, perform the following to ensure that the product was installed properly:

- In the BBI Journal, issue the BBI command **.D A** (display active).

This command shows all currently active products in the BBI-SS PAS; for example:

```
AA0101I AAO VERSION 6.2.0 STARTED ON 17-JAN-2001 AT 09:53:11
AA0102I CAO - ACTIVE
AA0103I IAO - ACTIVE
AA0104I MAO - ACTIVE
AA0106I QAO - ACTIVE
```

If QAO is not listed as active, AutoOPERATOR for MQSeries either has not been installed or has been installed incorrectly. For information about installing the product, refer to Chapter 2.

- In the BBI Journal, issue the BBI command `.D KEYS` (display keys).

This command shows all currently valid product option keys; for example:

CF8701I	COPY PROTECTION KEY STATUS:				
CF8702I	PRD	CPUID	EXP	SYS	STATUS
CF8703I	-----	-----	-----	-----	-----
CF8704I	MA0?9672- 15- *	1317- 99365- C4B0			VALID
CF8704I	MA0?9672- 08- *	3481- 99365- 4CA4			CPU- ERR
CF8704I	QA0- 9672- 15- 11317- 99365- D0A5				VALID
CF8704I	QA0- 9672- 08- 13481- 99365- D2DF				CPU- ERR

If you have typed an invalid key, as described in “Specifying Product Option Password Keys” on page 15, or you do not have the key, contact BMC Software Customer Support to obtain the key.

3. Verifying that the Rule Set and Rules are enabled is not relevant in this situation.
4. Verify the connection to the queue manager by using the AutoOPERATOR for MQSeries Workstation panel and issue the BBI command `.RESET MQ xx`, where `xx` represents the two-character suffix of the `AAOMQLxx` member, to connect to the queue manager.

If the queue manager is not connected, follow the instructions in Chapter 2 to connect it.

For information about how to use the Workstation display, refer to Chapter 5, “Viewing AutoOPERATOR for MQSeries Automation Statistics” on page 115.

5. You do not need to invoke the `QMCDSP00 EXEC` to determine which queues are available, because this information is not relevant.
6. Use `MAINVIEW` for MQSeries or `OS/390` commands to verify that the `OS/390` queue manager is usable. Refer to the *IBM MQSeries for OS/390 Problem Determination Guide* for more information about any problems you may encounter.
7. Use `MAINVIEW` for MQSeries or `PATROL` for MQ-Administrator to verify that the event queues are remote and working and the channels, transmission queue, and queue manager alias are working.

- **How can I find out why AutoOPERATOR is unable to monitor a particular queue?**

- 1a. Review the BBI-SS PAS' Start messages in the BBI Journal (or in the joblog) and look for the QA1113E message:

```
QA1113E UNABLE TO INITIALIZE A0 for MQ TASK
```

The QA1113E error message occurs when you bring up a queue manager after the BBI-SS PAS is started and the system is unable to initialize the AutoOPERATOR for MQSeries task. If you receive this error, you may need to contact BMC Software Customer Support for assistance.

If this message is not in the BBI Journal or the joblog, the BBI-SS PAS was started correctly.

- 1b. Review the joblog messages for any error messages involving the queue manager and its channel initiator.
- 1c. Verify that the MQSeries APF-authorized libraries are added to the STEPLIB DD CARD properly. If they are not added correctly or are missing, you may receive error message QA1114E. The error message and reason are displayed below.

```
QA1114E MQSERIES STEPLIB MISSING/INVALID
REASON  A0 for MQ tried to initialize, but failed because the STEPLIB
did not specify a valid MQSeries load library.
SYSTEM  A0 for MQ will not initialize.
ACTION
USER     Check the JCL for the BBI/SS to make sure the STEPLIB DD
ACTION   statement contains the MQSeries Authlib.
```

If the MQSeries libraries are missing from the STEPLIB DD concatenation, add them. The MQSeries libraries must be at or above the level of the OS/390 queue managers that are going to be automated. Also review the “Specifying Product Option Password Keys” on page 15.

2. Verify that the QAO product option key is properly installed and active. For information about how the QAO key is installed, refer to the section “Specifying Product Option Password Keys” on page 15.

If you have already followed the instructions, perform the following to ensure that the product was installed properly:

- In the BBI Journal, issue the BBI command **.D A** (display active).

This command shows all currently active products in the BBI-SS PAS; for example:

```
AA0101I  AAO VERSION 6.2.0 STARTED ON 17- JAN-2001 AT 09: 53: 11
AA0102I      CAO - ACTIVE
AA0103I      IAO - ACTIVE
AA0104I      MAO - ACTIVE
AA0106I      QAO - ACTIVE
```

If QAO is not listed as active, AutoOPERATOR for MQSeries either has not been installed or has been installed incorrectly. For information about installing the product, refer to Chapter 2.

- In the BBI Journal issue the BBI command **.D KEYS** (display keys).

This command shows all currently valid product option keys; for example:

CF8701I	COPY PROTECTION KEY STATUS:				
CF8702I	PRD	CPUID	EXP	SYS	STATUS
CF8703I	-----	-----	-----	-----	-----
CF8704I	MA0?9672- 15- *	1317- 99365- C4B0			VALID
CF8704I	MA0?9672- 08- *	3481- 99365- 4CA4			CPU- ERR
CF8704I	QA0- 9672- 15- 11317- 99365- DOA5				VALID
CF8704I	QA0- 9672- 08- 13481- 99365- D2DF				CPU- ERR

If you have typed an invalid key, as described in “Specifying Product Option Password Keys” on page 15, or you do not have the key, contact BMC Software Customer Support to obtain the key.

3. Verify that a Rule exists that refers to the queue and the queue manager and that the Rule and its Rule Set are enabled.
4. Verify the connection to the queue manager by using the AutoOPERATOR for MQSeries Workstation panel and issue the BBI command **.RESET MQ xx**, where **xx** represents the two-character suffix of the **AAOMQLxx** member, to reset the queue manager.

If the queue manager is not connected, follow the instructions in Chapter 2 to connect it.

For information about how to use the Workstation display, refer to “Chapter 5, “Viewing AutoOPERATOR for MQSeries Automation Statistics” on page 115”.

5. In the BBI Journal, issue the **%QMCDSP00** command for a list of all queues and their status to determine which queues are eligible for automation. Find the specific queue in the list to verify that it will be monitored. An example of the output from the **%QMCDSP00 EXEC** follows:

<b>.QMCDSP00</b>	EID=00064	excl_by_user	32	local	CSAE	EPESIN.XMI TQ
<b>.QMCDSP00</b>	EID=00064	excl_by_user	32	remote	CSAE	CICSPRD1
<b>.QMCDSP00</b>	EID=00064	excl_by_user	32	alias	CSAE	CICSPRD1.ALIASQ0
<b>.QMCDSP00</b>	EID=00064	included	27	local	CSAE	CICSPRD1.QUEUE1
<b>.QMCDSP00</b>	EID=00064	included	27	local	CSAE	CICSPRD1.QUEUE10
<b>.QMCDSP00</b>	EID=00064	included	27	local	CSAE	CICSPRD1.QUEUE11

The queues named **CICSPRD1.QUEUE\*** are eligible for automation.

If the queue is currently excluded, edit **BBPARM** member **AAOMQLxx** to add it to be included and issue a **.RESET MQ xx** command. Refer to “Parameters for **BBPARM** Member **AAOMQLxx**” on page 42 for more information.

6. Use **MAINVIEW** for MQSeries or **OS/390** commands to verify that the **OS/390** queue manager is usable. Refer to the *IBM MQSeries for OS/390 Problem Determination Guide* for more information about any problems you may encounter.
7. Use **MAINVIEW** for MQSeries or **PATROL** for MQ-Administrator to verify that the event queues are remote and working and the channels, transmission queue, and queue manager alias are working.

- **I have written an MQS Rule for an event that is not firing. What can I do?**

- 1a. Review the BBI-SS PAS' Start messages in the BBI Journal (or in the joblog) and look for the QA1113E message:

```
QA1113E UNABLE TO INITIALIZE A0 for MQ TASK
```

The QA1113E error message occurs when you bring up a queue manager after the BBI-SS PAS is started and the system is unable to initialize the AutoOPERATOR for MQSeries task. If you receive this error, you may need to contact BMC Software Customer Support for assistance.

If this message is not in the BBI Journal or the joblog, the BBI-SS PAS was started correctly.

- 1b. Review the joblog messages for any error messages involving the queue manager and its channel initiator.
- 1c. Verify that the MQSeries APF-authorized libraries are added to the STEPLIB DD CARD properly. If they are not added correctly or are missing, you may receive error message QA1114E. The error message and reason are displayed below.

```
QA1114E MQSERIES STEPLIB MISSING/INVALID
REASON  A0 for MQ tried to initialize, but failed because the STEPLIB
did not specify a valid MQSeries load library.
SYSTEM  A0 for MQ will not initialize.
ACTION
USER     Check the JCL for the BBI/SS to make sure the STEPLIB DD
ACTION   statement contains the MQSeries Authlib.
```

If the MQSeries libraries are missing from the STEPLIB DD concatenation, add them. The MQSeries libraries must be at or above the level of the OS/390 queue managers that are going to be automated. Also review the “Adding MQSeries APF-Authorized Libraries to the STEPLIB DD Card” on page 17.

2. Verify that the QAO product option key is properly installed and active. For information about how the QAO key is installed, refer to the section “Specifying Product Option Password Keys” on page 15.

If you have already followed the instructions, perform the following to ensure that the product was installed properly:

- In the BBI Journal, issue the BBI command **.D A** (display active).

This command shows all currently active products in the BBI-SS PAS; for example:

```
AA0101I  AAO VERSION 6. 2. 0 STARTED ON 17- JAN- 2001 AT 09: 53: 11
AA0102I      CAO - ACTIVE
AA0103I      IAO - ACTIVE
AA0104I      MAO - ACTIVE
AA0106I      QAO - ACTIVE
```

If QAO is not listed as active, AutoOPERATOR for MQSeries either has not been installed or has been installed incorrectly. For information about installing the product, refer to Chapter 2.

- In the BBI Journal, issue the BBI command **.D KEYS** (display keys).

This command shows all currently valid product option keys; for example:

CF8701I	COPY PROTECTION KEY STATUS:				
CF8702I	PRD	CPUID	EXP	SYS	STATUS
CF8703I	-----	-----	-----	-----	-----
CF8704I	MA0?9672- 15- *	1317- 99365- C4B0			VALID
CF8704I	MA0?9672- 08- *	3481- 99365- 4CA4			CPU- ERR
CF8704I	QA0- 9672- 15- 11317- 99365- D0A5				VALID
CF8704I	QA0- 9672- 08- 13481- 99365- D2DF				CPU- ERR

If you have typed an invalid key, as described in “Specifying Product Option Password Keys” on page 15, or you do not have the key, contact BMC Software Customer Support to obtain the key.

3. Verify that the Rule Set and Rules are enabled.

Use the Rules Processor Automation Control panels to verify that the correct Rule Set and Rules are enabled and to alter the selection criteria or actions taken by a given Rule.

If the selection criteria for the Rule are too specific, the Rule may not fire for all the occasions that it is expected to. You can determine if this is the problem by relaxing the selection criteria in the control panels.

4. Verify the connection to the queue manager by using the AutoOPERATOR for MQSeries Workstation panel and issue the BBI command **.RESET MQ xx**, where **xx** represents the two-character suffix of the **AAOMQLxx** member, to reset the queue manager.

If the queue manager is not connected, follow the instructions in Chapter 2 to connect it.

For information about how to use the Workstation display, refer to Chapter 5, “Viewing AutoOPERATOR for MQSeries Automation Statistics” on page 115”.

5. In the BBI Journal, issue the **%QMJDSP00** command for a list of all queues and their status to determine which queues are eligible for automation. Find the specific queue in the list to verify that it will be monitored. An example of the output from the **%QMJDSP00 EXEC** follows:

. QMJDSP00	EID=00064	excl_by_user	32	local	CSAE	EPESIN.XMITQ
. QMJDSP00	EID=00064	excl_by_user	32	remote	CSAE	CICSPRD1
. QMJDSP00	EID=00064	excl_by_user	32	alias	CSAE	CICSPRD1.ALIASQ0
. QMJDSP00	EID=00064	included	27	local	CSAE	CICSPRD1.QUEUE1
. QMJDSP00	EID=00064	included	27	local	CSAE	CICSPRD1.QUEUE10
. QMJDSP00	EID=00064	included	27	local	CSAE	CICSPRD1.QUEUE11

The queues named **CICSPRD1.\*** are eligible for automation.

If the queue is currently excluded, edit **BBPARM** member **AAOMQLxx** to add it to be included. Refer to “Parameters for **BBPARM** Member **AAOMQLxx**” on page 42 for more information.

You can also use the **%QMJDSP00 EXEC** with a specific queue name:

**%QMJDSP00 Q=BBOMVAO.YXP.QUEUE5.**

An example of the output from this EXEC follows:

```
. QMQDSP00 EID=00002 STARTED, PTF=BP05319
. QMQDSP00 EID=00002 qmgr=CSBD is down
. QMQDSP00 EID=00002 qmgr=CSBE is down
. QMQDSP00 EID=00002 qmgr=CSBC is down
. QMQDSP00 EID=00002 excl_by_user 77 local CSQ1 BBOMVA0. YXP. QUEUE5
. QMQDSP00 EID=00002 excl_by_user 77 local CSQA BBOMVA0. YXP. QUEUE5
. QMQDSP00 EID=00002 qmgr=CSQ2 is down
. QMQDSP00 EID=00002 ENDED
```

6. .Use MAINVIEW for MQSeries or OS/390 commands to verify that the OS/390 queue manager is usable. Refer to the *IBM MQSeries for OS/390 Problem Determination Guide* for more information about any problems you may encounter.
  7. Use MAINVIEW for MQSeries or PATROL for MQ-Administrator to verify that the event queues are remote and working and the channels, transmission queue, and queue manager alias are working.
- **I have written an EXEC where an IMFEXEC MQI PUT returns to me a non-zero condition code. What can I do?**
    1. Look up any messages or codes.

Examine the BBI Journal and the ALERT Detail application for error messages or other indications that AutoOPERATOR is not working correctly. For example, if an expected action did not occur, the BBI Journal may display whether or not the Rule fired that would have caused the action.

Refer to the *IBM MQSeries for OS/390 Messages and Codes* for more information about specific OS/390 messages. Depending on the platform, refer to the appropriate messages and codes book for non-OS/390 messages.

If there are no messages, you can include the IMFEXEC DISPLAY MQ statement in the EXEC. This statement displays in the BBI Journal the values for all current variables. Verify that the variable's values are correct and appropriate.
    2. Verify that QAO is active since the MQCONN call worked.
    3. Do not verify that the Rule Set and Rules are enabled because this information is not relevant in this situation.
    4. Verify that the queue manager is connected.
    5. Do not invoke the QMQDSP00 EXEC to verify which queues are available, because this information is not relevant.
    6. Use MAINVIEW for MQSeries or OS/390 commands to verify that the OS/390 queue manager is usable. Refer to the *IBM MQSeries for OS/390 Problem Determination Guide* for more information about any problems you may encounter.
    7. Use MAINVIEW for MQSeries or PATROL for MQ-Administrator to verify that the event queues are remote and working and the channels, transmission queue, and queue manager alias are working.
  - **What else can I do?**

If you have completed the checklist and need additional information, perform the following actions:

- Use MAINVIEW for MQSeries to obtain a general overview of the queues, channels, and queue managers being monitored by AutoOPERATOR. The MAINVIEW for MQSeries interface provides easy navigation to individual queues and queue managers and the ability to issue commands. You may also use the PATROL for MQ-Administrator interface for non-OS/390 queue managers.

- Hyperlink to the appropriate MAINVIEW for MQSeries views to display the MQSeries objects and attributes and overwrite their values to update them.

MAINVIEW for MQSeries issues the appropriate commands depending on whether you are listening to OS/390 or non-OS/390 queue managers. For a complete list of commands, see *MQSeries Command Reference* and the *MQSeries System Management Guide* for the appropriate platform.

- Display local queue manager's objects to determine if they are correctly defined and active.
  - Refer to the *MQSeries for OS/390 Problem Determination Guide*
  - Use MAINVIEW for MQSeries to inspect queues and channels
  - Use SDSF (System Display and Search Facility) or a similar application to issue MQSeries commands to verify that the queue manager and its channel initiator are running correctly
- Display remote queue manager's objects to determine if they are correctly defined and active.
  - Use MAINVIEW for MQSeries or PATROL for MQ-Administrator to inspect queues and channels
  - Refer to the platform-specific system management guide(s)
  - Use platform-specific commands to verify that the queue manager is active and its channel initiator is running

# Examples of Specific Automation Problems and Resolutions

This section provides examples of more specific automation problems and the steps taken to resolve them.

## Dead Letter Queue Solution

**My DLQ Solution Rule does not move normal messages from the dead letter queue. What do I do?**

Follow the “Diagnostic Checklist” on page 222 as follows:

1. Determine if you have received any messages or codes. For this scenario, there are no messages or codes to look up.
2. Issue the **.D A** command to verify that AutoOPERATOR for MQSeries is installed and that the QAO product option key is implemented properly.

Again, for this scenario, the product and the key are properly installed.

- 3a. Use the Automation Control panel to inspect the Rule Set and Rules to verify that the Rule Set and the DLQ Rule are both enabled. Also check the automation strategy setting.

The Automation Control panel shows that the Rule Set is enabled and the automation strategy is set to ALL. If the automation strategy was set to FIRST, you might need to review the individual Rules and ensure that DLQ Rule is high enough in the Rule Set for a FIRST strategy to work. If the automation strategy was set to MOST QUALIFIED, you should also review the individual Rules and ensure that there is not another Rule in the Rule Set that has more matching selection criteria set.

```

BMC Software ----- Automation Control ----- AutoOPERATOR
COMMAND ==>
Primary commands: Add, Statshow, Cmdshow
Automation Status ==> ACTIVE (Active, Inactive)
Automation Strategy ==> ALL (Individual, All, First, Qualified)
Honor MPF Suppression ==> YES (NO/YES)
Automation Library
LC CMDS --- (S)elect, (E)nable, (D)isable, (T)est, (SA)ve
(M)ove, (B)efore or (A)fter, (F)ilter Criteria
LC Rule-Set Status Rules Fired Filtered Date Time Strategy
___ AAORULBQ ENABLED 16 0 0 22-FEB-01 11:17:54 ALL
___ AAORULBA DI SABLED N/A N/A N/A N/A N/A
___ AAORULBB DI SABLED N/A N/A N/A N/A N/A
___ AAORULBC DI SABLED N/A N/A N/A N/A N/A
___ AAORULBD DI SABLED N/A N/A N/A N/A N/A
___ AAORULBE DI SABLED N/A N/A N/A N/A N/A
___ AAORULBF DI SABLED N/A N/A N/A N/A N/A
___ AAORULBG DI SABLED N/A N/A N/A N/A N/A
___ AAORULBH DI SABLED N/A N/A N/A N/A N/A
___ AAORULBP DI SABLED N/A N/A N/A N/A N/A
___ AAORULBR DI SABLED N/A N/A N/A N/A N/A
___ AAORULBS DI SABLED N/A N/A N/A N/A N/A
___ AAORULBV DI SABLED N/A N/A N/A N/A N/A

```

3b. Select the Rule Set.

The figure shows that the three DLQ Rules MQDEDQ01, MQDEDQ02, and MQDEDQ03 are enabled.

```

BMC Software ----- Rule Set Overview ----- AutoOPERATOR
COMMAND ==>                                     TGT --- SYSE
Rule Set ID: AAORULBQ                             DATE --- 01/02/22
Primary commands: Add, Save, Sort, Unsort, Reset, Filter
LC CMDS --- (S)elect, (E)nable, (D)isable, (T)est, (DE)lete, (I)nsert
              (C)opy/(CC)opy, (M)ove/(MM)ove, (B)efore or (A)fter, (R)peat
Sort Criterion:                               Filter ENABLED                               right/left

```

LC	Rule-id	Stat	Text-id	Type	Fired	EXEC	Changed	ID
___	MQINT001	DIS	Q_SERVICE_INTERV	MQS	0		97/03/14 11:09	BMC1
___	MQINT002	DIS	Q_SERVICE_INTERV	MQS	0		97/03/14 11:10	BMC1
___	MQQDP001	DIS	Q_DEPTH_HIGH	MQS	0		97/03/14 11:10	BMC1
___	MQQDP002	DIS	Q_DEPTH_LOW	MQS	0		97/03/14 11:10	BMC1
___	MQDEDQ01	ENA		MQS	0		01/02/14 12:34	BMCUSER
___	MQDEDQ02	ENA		TIME	0	MQDEDQ2	01/02/14 07:53	BMCUSER
___	MQDEDQ03	ENA		MQS	0		01/02/14 12:35	BMCUSER
___	MQARCO01	ENA		MQS	0	MQMUNLDQ	01/02/12 13:20	BMCUSER
___	MQARCO02	ENA	QA6033I	JRNL	0		01/02/12 15:09	BMCUSER
___	MQARCO03	ENA	QA6033I	JRNL	0		01/02/12 15:19	BMCUSER
___	MQARCO04	ENA	QA6033I	JRNL	0		01/02/12 13:24	BMCUSER
___	MQARCO05	ENA	QA6033I	JRNL	0		01/02/16 09:27	BMCUSER
___	MQDIA001	ENA	UNKNOWN_OBJECT_N	MQS	0		97/03/07 09:30	BMC1
___	MQDIA002	ENA	NOT_AUTHORIZED	MQS	0		97/03/14 09:24	BMC1

Editing Rule MQDEDQ01 shows the Rule has the correct queue manager and queue name specified.

```

BMC Software ----- Selection Criteria - MQS ----- AutoOPERATOR
COMMAND ==>                                     TGT --- SYSE

Rule-set == AAORULBQ                             Rule-id == MQDEDQ01

Queue Identification:                             (1 to 12 Queue Managers)
Manager(s) ==> CSBD
Queue Id    ==> &!QDED.IMFOQMGR

Message Identification:
Format      ==> USER                             (Value from MD Format field)
Event Type  ==>                                  (Enter ? for help)

Sub  Len  Op  Value
MsgId ==>  :  :  _____
CorrelId ==> :  :  _____
Msg Buffer ==> :  :  _____

Press ENTER to continue, END return to Detail Control, CANCEL to cancel changes

```

See Chapter 4 for descriptions of the shared variable QDEDxxxx (the name of the dead letter queue) and the TSO variable, IMFQRPQM (the name of the queue manager).

- Use the AutoOPERATOR for MQSeries Workstation panel to determine if the queue manager is connected and enabled for automation.

This figure shows that AutoOPERATOR is connected to the queue manager (CSQ1) and is monitoring 15 queues.

```

BMC Software ----- MQSeries Workstation ----- AutoOPERATOR
COMMAND ==>> TGT ==>> SYSE
Interval ==> 3 INPUT DATE --- 01/02/22
Commands: ALL, EQI xxx, NEQI xxx, EE xxx, NEE xxx TIME --- 16:00:28
----- Performance Statistics -----
Total Queues 228 Total Queues Included 37
Total MQS Messages Included 31 Total MQS Messages Handled 31
Instrument. Events Included 31 Instrument. Events Handled 31
User Messages Included 0 User Messages Handled 0
Current Instr. Arrival Rate/Min 0 Peak Instr. Arrival Rate/Min 27
Rule Generated Alerts 31 Rule Triggered EXECs 31
----- Queue Management -----
Queue Manager Queues Event Ques Enabled
Incl./Total Included Events
-----
CSQ1 37/228 QCP SLIPR
***** Bottom of Data *****

```

- Issue the BBI command .D V QDED.CSQ1 to display the name of the dead letter queue for your OS/390 queue manager. The output will look like this.

```

BMC Software ----- Log Display ----- General services
COMMAND ==>> TGT ==>> LOCAL
Line 2 Log #1 Status INPUT Time 16:07:06 INTV==>> 3
16:07:05 .D V QDED.CSQ1
16:07:06 IM9100I COMMAND ACCEPTED JB61
16:07:06 IM9207I SHARED VARIABLE POOL DISPLAY JB61
16:07:06 IM9211I SELECTED: QDED.CSQ1 JB61
16:07:06 IM9212I QDED.CSQ1 DEAD.LETTER.QUEUE JB61
***** END OF LOG *****

```

- Type the %QM DSP00 command in the BBI Journal to check the status of the dead letter queue.

The dead letter queue (DEAD.LETTER.QUEUE) is not eligible for automation

```

.QM DSP00 EID=00021 excl_by_user 17 local CSQ1 DEAD.LETTER.QUEUE

```

You can also edit and review BBPARM member AAOMQL00 to determine if the dead letter queue has been included. As you can see from the picture below, an incorrect dead letter queue name was used and needs to be changed to DEAD.LETTER.QUEUE.

```
TYPE(INCL) QMGR(CS*) QUEUE(SYSTEM ADMIN. *)
TYPE(INCL) QMGR(CS*) QUEUE(SYSBMC. SYSE. EVENTS)
TYPE(INCL) QMGR(CS*) QUEUE(SYSBMC. DI STRI B. EVENTS)
TYPE(INCL) QMGR(CS*) QUEUE(BBOMVAO. QUEUE*)
TYPE(EXCL) QMGR(CS*) QUEUE(BBOMVAO. SETUP. QUEUE*)
TYPE(INCL) QMGR(CS*) QUEUE(BBOMVAO. LI VE. *)
TYPE(INCL) QMGR(CS*) QUEUE(BBOMVAO. PERM. QUEUE)
TYPE(INCL) QMGR(CS*) QUEUE(SYSTEM DEAD. LETTER. QUEUE)
T(E) QMGR(*) QUEUE(*)
```

**Note:** You must issue the BBI control command .RESET MQ 00 to activate the AAOMQL00 member and at least one enabled Rule must refer to this queue manager for AutoOPERATOR to connect to the queue manager.

The rule did not move this user message to another queue because the wrong name was used in the AAOMQLnn BBPARM member to refer to the active dead letter queue.

7. Change the AAOMQL00 BBIPARM member to include SYSTEM.DEAD.LETTER.QUEUE instead of DEAD.LETTER.QUEUE.

8. Issue the **.E MQ 00** command.

The Rule now fires.

# Non-OS/390 Command Times Out

My EXEC times out when it issues a display command to a non-OS/390 queue manager. How do I fix this?

Use the “Diagnostic Checklist” on page 222 as a guideline when completing the steps below.

1. Look up the code X' 7F1' that is displayed by the EXEC in the BBI Journal.

```

%MQRNTCMD
EM9201I MQGET 0001 LEN=X'0000' CC=X'0002' RC=X'000007F1'
EM0022E ERROR PROCESSING .. MQRNTCMD .. TIMED OUT WAITING FOR RSP
EM0025I FOLLOWING MSG ISSUED FOR EXEC .. MQRNTCMD ..
      11 *- * "IMFEXEC CMD 'DISPLAY QMGR ALL' TYPE (MQS) QM("mvs_qn") PC
          LM("mvs_qn") CQ (SYSTEM.ADMIN.COMMAND.QUEUE) "
          +++ RC(4) +++
  
```

According to *MQSeries for OS/390 Messages and Codes*, X' 7F1' means an MQGET request timed out.

2. We know the QAO is active because the EXEC used the MQI successfully.
3. Since there are no Rules to consider, you do not need to refer to the Rules Processor Automation Control panels.
4. Look at the AutoOPERATOR for MQSeries OCD.

The panel indicates that AutoOPERATOR is connected to queue manager CSQ4, which is correct.

5. There are no Rules, so you do not need to determine which queues are eligible for automation.
6. Return to MAINVIEW for MQSeries to complete the following procedures:
  - a. Validate the OS/390 queue manager.

- 1) Obtain the name of the transmission queue.

In this case, the queue manager alias is PCOCHRAN and its transmission queue is CSQ4.

```

25FEB1997 14:55:53 ----- INFORMATION DISPLAY --
COMMAND ==>
CURR WIN ==> 1      ALT WIN ==>
W1 =RQD=====CSQ4====CSQ4====25FEB1997==14:55:52====MUMQS=====1=
Queue..... PCOCHRAN
Description ..... (none)
Queue Manager Name.. CSQ4
Transmission Queue.. PCOCHRAN.XMITQ
  
```

- 2) Verify that the XMITQ is empty by hyperlinking from XMITQ to a detailed view.

```

25FEB1997 14:59:12 ----- INFORMATION DISPLAY --
COMMAND ==>
CURR WIN ==> 1      ALT WIN ==>
W1 =LQD=====CSQ4====CSQ4====25FEB1997==14:59:09====MUMQS=====1=
Current Depth..... 0      Queue..... PCOCHRAN.XMITQ
Maximum Depth..... 99999999 Description..... (none)
  
```

3) Verify that the Sender channel is running by hyperlinking to the CHNLS view.

```

25FEB1997 14:50:08 ----- INFORMATION DISPLAY -----
COMMAND ==>
CURR WIN ==> 1      ALT WIN ==>
W1 =CHNLS===CHNLAD===CSQ4===CSQ4===25FEB1997==14:49:00===MUMQS=====1=
Channel Name..... CSQ4.TO.PCOCHRAN      Description... (none)
Type..... SENDER                          Queue Manager.. CSQ4
Status..... RUNNING                       Xmit Queue.... PCOCHRAN.XMITQ
                                           Put Authority.. CONTEXT

```

4) Verify that the Receiver Channel is running.

```

25FEB1997 16:16:44 ----- INFORMATION DISPLAY -----
COMMAND ==>
CURR WIN ==> 1      ALT WIN ==>
W1 =CHNLS===CHNLAD===CSQ4===CSQ4===25FEB1997==16:13:41===MUMQS=====1=
Channel Name..... PCOCHRAN.TO.CSQ4      Description... RECEIVER CHANNEL FOR PCOCHRAN (NT)
Type..... RECEIVER                        Queue Manager.. CSQ4
Status..... RUNNING                       Xmit Queue.... N/A

```

7. Validate the non-OS/390 queue manager.

In this example, we are using native MQSeries commands on Windows NT.

a. Verify that the command queue is empty and enabled.

```

dis q(system.admin.command.queue) curdepth put get
3 : dis q(system.admin.command.queue) curdepth put get
AMQ8409: Display Queue details.
  QUEUE(SYSTEM.ADMIN.COMMAND.QUEUE)
  GET(ENABLED)
  PUT(ENABLED)
  CURDEPTH(0)

```

b. Stop the command server and reissue the command.

The command is present in the queue.

```

dis q(system.admin.command.queue) curdepth
1 : dis q(system.admin.command.queue) curdepth
AMQ8409: Display Queue details.
  QUEUE(SYSTEM.ADMIN.COMMAND.QUEUE)
  CURDEPTH(1)

```

c. Restart the command server.

d. Determine if the queue manager alias is PUT inhibited.

```

dis q(csq4) put
16 : dis q(csq4) put
AMQ8409: Display Queue details.
  QUEUE(CSQ4)
  PUT(DISABLED)

```

e. Alter the queue manager alias to PUT (ENABLED).

The non-OS/390 command no longer times out.

## Rule Will Not Enable a Queue

### A Rule will not enable a non-OS/390 queue. What is the solution?

Use the “Diagnostic Checklist” on page 222 as a guideline when completing the steps below.

1. There are no error messages or codes.
2. The Rule fired, so QAO must be active.
3. Write the text of the command issued by the Rule as an ALERT to verify that the syntax is correct. If it is correct, proceed with Step 4.
4. Since the Rule fires, AutoOPERATOR must be connected to the queue manager.
5. The event queues are eligible for automation. There is no need for further checking.
- 6a. Use MAINVIEW for MQSeries to verify that the channel to the non-OS/390 queue manager is running.

```

W1 =CHNLS===CHNLAD===CSQ4=====25FEB1997==23:16:30===MVMQS=====1=
Channel Name..... CSQ4.TO.PCOCHRAN   Description... (none)
Type..... SENDER                       Queue Manager.. CSQ4
Status..... RUNNING                     Xmit Queue.... PCOCHRAN.XMITQ
  
```

- 6b. Verify that the transmission queue is empty and enabled by hyperlinking from its name to a detailed view of its attributes.

```

W1 =LQD=====CSQ4====CSQ4=====25FEB1997==23:13:13===MVMQS=====1=
Current Depth..... 0                   Queue..... PCOCHRAN.XMITQ
Maximum Depth..... 99999999           Description..... (none)

Inhibited Actions.... Trigger.....
Gets..... No                       Control..... On
Puts..... No                       Type..... First
  
```

- 7a. Determine the status of the non-OS/390 command queue. Use the MQSC display commands on Windows NT.

The remote command queue is GET and PUT enabled, but it is not empty.

```

dis q(system.admin.command.queue) curdepth put get
1 : dis q(system.admin.command.queue) curdepth put get
AMQ8409: Display Queue details.
QUEUE(SYSTEM.ADMIN.COMMAND.QUEUE)
GET(ENABLED)
PUT(ENABLED)
CURDEPTH(126)
  
```

- 7b. Start the command server on the non-OS/390 queue manager.

The command from the Rule will now execute successfully for the non-OS/390 queue.



---

## Appendix B. MQI Sample Coding

To help users write MQ Automation more quickly, AutoOPERATOR for MQSeries provides sample code for the more common AutoOPERATOR commands. These examples contain a shell consisting of all the pieces necessary to execute an MQSeries request, such as putting a message to a queue or manipulating an MQSeries resource. The example code can be copied or cut and pasted into your EXECs, where you can then modify the examples, or portions of the examples, to fit the needs for your particular request.

These examples are located in BBSAMP and are named QMQNB001 – QMQNB009.

Table 20 lists the nine examples and the actions that they perform.

Table 20. MQI Code Examples

Example name	Description
QMQNB001	<p>PUTs a message to a queue on an OS/390 queue manager</p> <p>The code is divided into five sections that each complete a specific task:</p> <ul style="list-style-type: none"><li>• Connects to a queue manager</li><li>• Opens an MQSeries queue</li><li>• PUTs a message to the queue</li><li>• Closes the queue</li><li>• Disconnects from the queue manager</li></ul>
QMQNB002	<p>PUTs a message to a queue on an OS/390 queue manager using PUT1</p> <p>The code is divided into three tasks:</p> <ul style="list-style-type: none"><li>• Connects to a queue manager</li><li>• PUT1s a message to a queue</li><li>• Disconnects from the queue manager</li></ul>
QMQNB003	<p>Browses a local queue defined to an OS/390 queue manager</p> <p>The code is divided into five tasks:</p> <ul style="list-style-type: none"><li>• Connects to a queue manager</li><li>• Opens an MQSeries queue</li><li>• Gets a message for browse from an opened queue</li><li>• Closes the queue</li><li>• Disconnects from the queue manager</li></ul>

Table 20. MQI Code Examples (Continued)

Example name	Description
QMQNB004	<p>PUTs a message to two different queues using syncpoint</p> <p>The code is divided into nine tasks:</p> <ul style="list-style-type: none"> <li>• Connects to a queue manager</li> <li>• Opens the first queue</li> <li>• PUTs a message to the first queue and begins syncpoint processing</li> <li>• Closes that queue</li> <li>• Opens the second queue</li> <li>• PUTs a message to the second queue using the same syncpoint</li> <li>• Closes the second queue</li> <li>• Commits all work since last syncpoint</li> <li>• Disconnects from the queue manager</li> </ul>
QMQNB005	<p>Displays the status of a queue manager resource</p> <p>The code contains one task:</p> <p style="padding-left: 40px;">Displays the status of performance events for the queue manager</p>
QMQNB006	<p>Alters the status of a queue manager</p> <p>The code is contains one task:</p> <p style="padding-left: 40px;">Alters the QMGR to allow INHIBIT events for the queue manager</p>
QMQNB007	<p>Alters the status of a queue</p> <p>The code contains one task:</p> <p style="padding-left: 40px;">Alters a queue to allow Q_Depth_High monitoring and to set a threshold of 80%</p>
QMQNB008	<p>GETs 10 messages for browse from a queue and writes the message text to the BBI Journal</p> <p>The code is divided into five tasks:</p> <ul style="list-style-type: none"> <li>• Connects to the queue manager</li> <li>• Opens a queue</li> <li>• GETs 10 messages from the queue and displays each of them in the BBI Journal</li> <li>• Closes the queue</li> <li>• Disconnects the queue manager</li> </ul>
QMQNB009	<p>Contains a diagnostic procedure that can be copied into any EXEC that uses AutoOPERATOR MQI or AutoOPERATOR for MQSeries command facility</p> <p>The code is divided into three tasks:</p> <ul style="list-style-type: none"> <li>• Displays non-zero return codes</li> <li>• Displays MQ messages</li> <li>• Returns to caller</li> </ul>

In the following example, QMQNB001 PUTs a message to a queue on an OS/390 queue manager:

---

```
/*-----REXX-----*/
/* QMQNB001 - */
/* This sample PUTs a message to a queue on an OS/390 Queue Manager. */
/* Keyword parameters will be used to set the appropriate values for */
/* input to each call to the MQI. */
/* Change Activity: */
/* 09-29-2000: Updated for new interface */
/*-----*/

/*-----*/
/* This statement causes a connection to the Queue Manager. */
/* NAME = Queue Manager ID */
/*-----*/
"IMFEXEC MQI CONN NAME(mqi d) "

/*-----*/
/* This statement opens an MQSeries Queue. */
/* HOBJ = Queue name */
/* HCONN = connection handle (can be omitted, default is IMFHCONN) */
/* OOPTS = queue is opened for output */
/*-----*/
IMFMQI_OD_OBJECTNAME = qname
"IMFEXEC MQI OPEN HCONN(IMFHCONN) OOPTS(MQOO_OUTPUT) "

/*-----*/
/* This statement PUTs a message to a previously opened queue. */
/* HCONN = connection handle (can be omitted, default is IMFHCONN) */
/* HOBJ = queue name, set by previous OPEN */
/* POPTS = PUT options */
/* BUFFLEN = length of variable containing application data */
/* BUFFER = name of variable that contains application data */
/*-----*/
"IMFEXEC MQI PUT HCONN(IMFHCONN) HOBJ(IMFHOBJ) POPTS(MQPMO_NO_SYNCPOINT) ",
"BUFFLEN("LENGTH(message)") BUFFER(message) "

/*-----*/
/* This statement closes an MQSeries Queue. */
/* HOBJ = Queue name */
/* HCONN = connection handle (can be omitted, default is IMFHCONN) */
/* COPTS = None is specified for pre-defined queues */
/*-----*/
"IMFEXEC MQI CLOSE COPTS(MQCO_NONE) HOBJ(IMFHOBJ) "

/*-----*/
/* This statement causes a disconnection from the Queue Manager. */
/* HCONN = Connection handle set during previous connect */
/*-----*/
"IMFEXEC MQI DISC HCONN(IMFHCONN) "
```

---



## Appendix C. Rules Variables

### EVENT Variables – Rules and Rule-Invoked EXECs

This table lists the event variables - Rules and Rule-Invoked EXECs.

Table 21. EVENT Variables – Rules and Rule-Invoked EXECs

Variable Name	Notes
IMFQ_EVENT_TYPE	Event type-not an MQSeries field
IMFQ_EVENT_QMGRNAME	Name of queue manager that generated event, whether local or remote
IMFQ_EVENT_QNAME	Name of queue from object descriptor
IMFQ_EVENT_BASEQNAME	Name of queue manager (to which the alias resolves)
IMFQ_EVENT_APPLNAME	Name of application
IMFQ_EVENT_OBJECTQMGRNAME	Name of the object queue manager
IMFQ_EVENT_CHANNEL_NAME	Name of channel
IMFQ_EVENT_CONNECTIONNAME	Connection name. For TCP/IP, this is the internet address only if the channel has successfully established a connection. Otherwise, it is the contents of the ConnectionName field in the channel definition.
IMFQ_EVENT_XMITQNAME	Transmission queue name
IMFQ_EVENT_FORMAT	Name of format
IMFQ_EVENT_QTYPE	Type of queue
IMFQ_EVENT_APPLTYPE	Type of queue to which the alias resolves (A for alias, M for model)
IMFQ_EVENT_REASONQUALIFIER	Reason qualifier: MQRQ_CHANNEL_STOPPED_OK MQRQ_CHANNEL_STOPPED_ERROR MQRQ_CHANNEL_STOPPED_RETRY MQRQ_CHANNEL_STOPPED_DISABLED
IMFQ_EVENT_ERRORIDENTIFIER	See the section regarding the CHANNEL_STOPPED event, field Error identifier, in the IBM publication <i>MQSeries Programmable System Management</i> for a description of this data.
IMFQ_EVENT_PROCESSNAME	Name of process
IMFQ_EVENT_USERIDENTIFIER	Name of user identifier
IMFQ_EVENT_AUXERRORDATAINT1	First integer of auxiliary error data for channel errors
IMFQ_EVENT_AUXERRORDATAINT2	Second integer of auxiliary error data for channel errors

Table 21. EVENT Variables – Rules and Rule-Invoked EXECs

Variable Name	Notes
IMFQ_EVENT_AUXERRORDATASTR1	First string of auxiliary error data for channel errors
IMFQ_EVENT_AUXERRORDATASTR2	Second string of auxiliary error data for channel errors
IMFQ_EVENT_AUXERRORDATASTR3	Third string of auxiliary error data for channel errors
IMFQ_EVENT_TIMESINCERESET	Time (in seconds) since the statistics were last reset; for this event, this value is greater than the service interval
IMFQ_EVENT_HIGHQDEPTH	Max number of messages on the queue since the statistics were last reset
IMFQ_EVENT_MSGENQCOUNT	Number of messages enqueued since statistics were last reset
IMFQ_EVENT_MSGDEQCOUNT	Number of messages dequeued since statistics were last reset
IMFQ_EVENT_BRIDGENAME	Name of bridge
IMFQ_EVENT_BRIDGETYPE	Only current value is 1 (OTMA)
IMFQ_EVENT_OPTIONS	Open options
IMFQ_EVENT_COMMAND	Name of command event
IMFQ_EVENT_CURDEPTH	Current depth of the event queue
IMFQ_EVENT_MAXDEPTH	Maximum depth of the event queue
IMFQ_EVENT_PERCENTFULL	Percent full of the event queue

## Dead Letter Header Variables – Rules and Rule-Invoked EXECs

This table lists the Dead Letter Header variables – Rules and Rule-Invoked EXECs.

Table 22. Dead Letter Header Variables – Rules and Rule-Invoked EXECs

Variable Name	Notes
<b>Note:</b>	This list may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i> .
IMFQ_DLH_STRUCID	Contains MQDLH_STRUC_ID.
IMFQ_DLH_VERSION	Contains MQDLH_VERSION_1.
IMFQ_DLH_REASON	Contains a numeric reason code between 0 and 999999999.
IMFQ_DLH_DESTQNAME	Contains the 48-character destination queue name.
IMFQ_DLH_DESTQMGRNAME	Contains the 48-character destination queue manager name.
IMFQ_DLH_ENCODING	Contains decimal data.
IMFQ_DLH_CODEDCHARSETID	Contains decimal data.
IMFQ_DLH_FORMAT	Contains one of the following format constants: <ul style="list-style-type: none"> <li>MQFMT_NONE (IBM default)</li> <li>MQFMT_ADMIN</li> <li>MQFMT_CHANNEL_COMPLETED</li> <li>MQFMT_CICS</li> <li>MQFMT_COMMAND_1</li> <li>MQFMT_COMMAND_2</li> <li>MQFMT_DEAD_LETTER_HEADER</li> <li>MQFMT_EVENT</li> <li>MQFMT_IMS</li> <li>MQFMT_IMS_VAR_STRING</li> <li>MQFMT_MD_EXTENSION</li> <li>MQFMT_PCF</li> <li>MQFMT_REF_MSG_HEADER</li> <li>MQFMT_STRING</li> <li>MQFMT_TRIGGER</li> <li>MQFMT_WORK_INFO_HEADER</li> <li>MQFMT_XMIT_Q_HEADER</li> <li>MQFMT_RF_HEADER</li> <li>MQFMT_RF_HEADER_2</li> </ul>

Table 22. Dead Letter Header Variables – Rules and Rule-Invoked EXECs

Variable Name	Notes
IMFQ_DLH_PUTAPPLTYPE	<p>Contains a numeric user-defined type or one of the following standard types:</p> <p>MQAT_AIX            MQAT_CICS            MQAT_CICS_BRIDGE            MQAT_CICS_VSE            MQAT_DOS            MQAT_GUARDIAN            MQAT_IMS            MQAT_IMS_BRIDGE            MQAT_MVS            MQAT_NOTES_AGENT            MQAT_NSK            MQAT_OS2            MQAT_OS390            MQAT_OS400            MQAT_QMGR            MQAT_UNIX            MQAT_VMS            MQAT_VOS            MQAT_WINDOWS            MQAT_WINDOWS_NT            MQAT_XCF            MQAT_UNKNOWN</p> <p><i>See note at the beginning of this table.</i></p>
IMFQ_DLH_PUTAPPLNAME	Contains up to a 28-character name for PutApplName.
IMFQ_DLH_PUTDATE	Contains an 8-character date stamp.
IMFQ_DLH_PUTTIME	Contains an 8-character time stamp.

## Trigger Message Variables – Rules and Rule-Invoked EXECs

This table lists the Trigger Message variables – Rules and Rule-Invoked EXECs.

Table 23. Trigger Message Variables – Rules and Rule-Invoked EXECs

Variable Name	Notes
<b>Note:</b>	This list may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i> .
IMFQ_TM_STRUCID	Contains MQTM_STRUC_ID.
IMFQ_TM_VERSION	Contains MQTM_VERSION_1.
IMFQ_TM_QNAME	Contains 48-character name of triggered queue.
IMFQ_TM_PROCESSNAME	Contains 48-character name of process object.
IMFQ_TM_TRIGGERDATA	Contains 64-character free-format trigger data.
IMFQ_TM_APPLTYPE	<p>Contains a numeric user-defined type or one of the following standard types:</p> <ul style="list-style-type: none"> <li>MQAT_AIX</li> <li>MQAT_CICS</li> <li>MQAT_CICS_BRIDGE</li> <li>MQAT_CICS_VSE</li> <li>MQAT_DOS</li> <li>MQAT_GUARDIAN</li> <li>MQAT_IMS</li> <li>MQAT_IMS_BRIDGE</li> <li>MQAT_MVS</li> <li>MQAT_NOTES_AGENT</li> <li>MQAT_NSK</li> <li>MQAT_OS2</li> <li>MQAT_OS390</li> <li>MQAT_OS400</li> <li>MQAT_QMGR</li> <li>MQAT_UNIX</li> <li>MQAT_VMS</li> <li>MQAT_VOS</li> <li>MQAT_WINDOWS</li> <li>MQAT_WINDOWS_NT</li> <li>MQAT_XCF</li> <li>MQAT_UNKNOWN</li> </ul> <p><i>See note at the beginning of this table.</i></p>
IMFQ_TM_APPLID	Contains a 256-character string.
IMFQ_TM_ENVDATA	Contains a 128-character string.
IMFQ_TM_USERDATA	Contains a 128-character string.

## CICS Information Header Variables – Rules and Rule-Invoked EXECs

This table lists the CICS Information Header variables – Rules and Rule-Invoked EXECs.

Table 24. CICS Information Header Variables – Rules and Rule-Invoked EXECs

Variable Name	Notes
<b>Note:</b>	This list may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i> .
IMFQ_CIH_STRUCID	Contains the value MQCIH_STRUC_ID.
IMFQ_CIH_VERSION	Contains the value MQCIH_VERSION_2.
IMFQ_CIH_STRUCLength	Contains the value MQCIH_LENGTH_2.
IMFQ_CIH_ENCODING	Contains the decimal 0.
IMFQ_CIH_CODEDCHARSETID	Contains the decimal 0.
IMFQ_CIH_FORMAT	Contains one of the following format constants: <ul style="list-style-type: none"> <li>MQFMT_NONE (IBM default)</li> <li>MQFMT_ADMIN</li> <li>MQFMT_CHANNEL_COMPLETED</li> <li>MQFMT_CICS</li> <li>MQFMT_COMMAND_1</li> <li>MQFMT_COMMAND_2</li> <li>MQFMT_DEAD_LETTER_HEADER</li> <li>MQFMT_EVENT</li> <li>MQFMT_IMS</li> <li>MQFMT_IMS_VAR_STRING</li> <li>MQFMT_MD_EXTENSION</li> <li>MQFMT_PCF</li> <li>MQFMT_REF_MSG_HEADER</li> <li>MQFMT_STRING</li> <li>MQFMT_TRIGGER</li> <li>MQFMT_WORK_INFO_HEADER</li> <li>MQFMT_XMIT_Q_HEADER</li> <li>MQFMT_RF_HEADER</li> <li>MQFMT_RF_HEADER_2</li> </ul>
IMFQ_CIH_FLAGS	Contains MQCIH_NONE.
IMFQ_CIH_RETURNCODE	Contains one of the following values: <ul style="list-style-type: none"> <li>MQCRC_OK</li> <li>MQCRC_CICS_EXEC_ERROR</li> <li>MQCRC_MQ_API_ERROR</li> <li>MQCRC_BRIDGE_ERROR</li> <li>MQCRC_BRIDGE_ABEND</li> <li>MQCRC_APPLICATION_ABEND</li> <li>MQCRC_SECURITY_ERROR</li> <li>MQCRC_PROGRAM_NOT_AVAILABLE</li> <li>MQCRC_BRIDGE_TIMEOUT</li> <li>MQCRC_TRANSID_NOT_AVAILABLE</li> </ul>

Table 24. CICS Information Header Variables – Rules and Rule-Invoked EXECs

Variable Name	Notes
IMFQ_CIH_COMPCODE	Contains decimal data.
IMFQ_CIH_REASON	Contains decimal data.
IMFQ_CIH_UOWCONTROL	Contains one of the following values: MQCUOWC_ONLY MQCUOWC_CONTINUE MQCUOWC_FIRST MQCUOWC_MIDDLE MQCUOWC_LAST MQCUOWC_COMMIT MQCUOWC_BACKOUT
IMFQ_CIH_GETWAITINTERVAL	Contains a decimal or one of the following constants: MQCGWI_DEFAULT MQWI_UNLIMITED
IMFQ_CIH_LINKTYPE	Contains one of the following values: MQCLT_PROGRAM MQCLT_TRANSACTION
IMFQ_CIH_OUTPUTDATALENGTH	Contains decimal data or the constant: MQCLT_TRANSACTION
IMFQ_CIH_FACILITYKEEPTIME	Contains decimal data.
IMFQ_CIH_ADSDDESCRIPTOR	Contains one of the following values: MQCADSD_NONE MQCADSD_SEND MQCADSD_RECV MQCADSD_MSGFORMAT  <i>See note at the beginning of this table.</i>
IMFQ_CIH_CONVERSATIONALTASK	Contains one of the following values: MQCCT_YES MQCCT_NO
IMFQ_CIH_TASKENDSTATUS	Contains one of the following values: MQCTES_NOSYNC MQCTES_COMMIT MQCTES_BACKOUT MQCTES_ENDTASK
IMFQ_CIH_FACILITY	Contains a 8-byte hexadecimal or the following constant: MQCFAC_NONE. No conversion of hexadecimal data is done.

Table 24. CICS Information Header Variables – Rules and Rule-Invoked EXECs

Variable Name	Notes
IMFQ_CIH_FUNCTION	<p>Contains a 4-character value of one of the following constants:</p> <p>MQCFUNC_MQCONN            MQCFUNC_MQGET            MQCFUNC_MQINQ            MQCFUNC_MQOPEN            MQCFUNC_MQPUT            MQCFUNC_MQPUT1            MQCFUNC_NONE</p> <p><i>See note at the beginning of this table.</i></p>
IMFQ_CIH_ABENDCODE	Contains a 4-character value.
IMFQ_CIH_AUTHENTICATOR	Contains an 8-character value.
IMFQ_CIH_RESERVED1	Contains 8 blanks.
IMFQ_CIH_REPLYTOFORMAT	<p>Contains one of the following format constants:</p> <p>MQFMT_NONE (IBM default)            MQFMT_ADMIN            MQFMT_CHANNEL_COMPLETED            MQFMT_CICS            MQFMT_COMMAND_1            MQFMT_COMMAND_2            MQFMT_DEAD_LETTER_HEADER            MQFMT_EVENT            MQFMT_IMS            MQFMT_IMS_VAR_STRING            MQFMT_MD_EXTENSION            MQFMT_PCF            MQFMT_REF_MSG_HEADER            MQFMT_STRING            MQFMT_TRIGGER            MQFMT_WORK_INFO_HEADER            MQFMT_XMIT_Q_HEADER            MQFMT_RF_HEADER            MQFMT_RF_HEADER_2</p>
IMFQ_CIH_REMOTESYSID	Contains 4 blanks.
IMFQ_CIH_REMOTETRANSID	Contains 4 blanks.
IMFQ_CIH_TRANSACTIONID	Contains a 4-character value.
IMFQ_CIH_FACILITYLIKE	Contains a 4-character value.
IMFQ_CIH_ATTENTIONID	Contains a 4-character value.

Table 24. CICS Information Header Variables – Rules and Rule-Invoked EXECs

Variable Name	Notes
IMFQ_CIH_STARTCODE	Contains one of the following constants: MQCSC_START MQCSC_STARTDATA MQCSC_TERMINPUT MQCSC_NONE  <i>See note at the beginning of this table.</i>
IMFQ_CIH_CANCELCODE	Contains a 4-character value.
IMFQ_CIH_NEXTTRANSACTIONID	Contains a 4-character value.
IMFQ_CIH_RESERVED2	Contains 8 blanks.
IMFQ_CIH_RESERVED3	Contains 8 blanks.
IMFQ_CIH_CURSORPOSITION	Contains decimal data.
IMFQ_CIH_ERROROFFSET	Contains decimal data.
IMFQ_CIH_INPUTITEM	Contains the decimal 0.
IMFQ_CIH_RESERVED4	Contains the decimal 0.

## IMS Information Header Variables – Rules and Rule-Invoked EXECs

This table lists the IMS Information Header variables – Rules and Rule-Invoked EXECs.

Table 25. IMS Information Header Variables – Rules and Rule-Invoked EXECs

Variable Name	Notes
IMFQ_III_STRUCID	Contains MQIIH_STRUC_ID.
IMFQ_III_VERSION	Contains MQIIH_VERSION_1.
IMFQ_III_STRUCLength	Contains MQIIH_LENGTH_1.
IMFQ_III_ENCODING	Contains the decimal 0.
IMFQ_III_CODEDCHARSETID	Contains the decimal 0.
IMFQ_III_FORMAT	Contains one of the following format constants: <ul style="list-style-type: none"> <li>MQFMT_NONE (IBM default)</li> <li>MQFMT_ADMIN</li> <li>MQFMT_CHANNEL_COMPLETED</li> <li>MQFMT_CICS</li> <li>MQFMT_COMMAND_1</li> <li>MQFMT_COMMAND_2</li> <li>MQFMT_DEAD_LETTER_HEADER</li> <li>MQFMT_EVENT</li> <li>MQFMT_IMS</li> <li>MQFMT_IMS_VAR_STRING</li> <li>MQFMT_MD_EXTENSION</li> <li>MQFMT_PCF</li> <li>MQFMT_REF_MSG_HEADER</li> <li>MQFMT_STRING</li> <li>MQFMT_TRIGGER</li> <li>MQFMT_WORK_INFO_HEADER</li> <li>MQFMT_XMIT_Q_HEADER</li> <li>MQFMT_RF_HEADER</li> <li>MQFMT_RF_HEADER_2</li> </ul>
IMFQ_III_FLAGS	Contains MQIIH_NONE.
IMFQ_III_LTERM_OVERRIDE	Contains an 8-character value.
IMFQ_III_MFSMAPNAME	Contains an 8-character value.

Table 25. IMS Information Header Variables – Rules and Rule-Invoked EXECs

Variable Name	Notes
IMFQ_IIH_REPLYTOFORMAT	Contains one of the following format constants:  MQFMT_NONE (IBM default) MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PCF MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2
IMFQ_IIH_AUTHENTICATOR	Contains an 8-character value or the value MQIAUT_NONE.
IMFQ_IIH_TRANINSTANCEID	Contains a 16-byte hexadecimal or the constant MQITII_NONE.  No conversion of hexadecimal data is done.
IMFQ_IIH_TRANSTATE	Contains one of the following constants:  MQITS_IN_CONVERSATION MQITS_NOT_IN_CONVERSATION MQITS_ARCHITECTED
IMFQ_IIH_COMMITMODE	Contains one of the following constants:  MQICM_COMMIT_THEN_SEND MQICM_SEND_THEN_COMMIT
IMFQ_IIH_SECURITYSCOPE	Contains one of the following constants:  MQISS_CHECK MQISS_FULL
IMFQ_IIH_RESERVED	Contains one blank.

## Transmission Queue Header Variables – Rules and Rule-Invoked EXECs

This table lists the Transmission Queue Header variables – Rules and Rule-Invoked EXECs.

Table 26. Transmission Queue Header Variables – Rules and Rule-Invoked EXECs

Variable Name	Notes
<b>Note:</b>	This list may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i> .
IMFQ_XQH_STRUCID	Contains MQXQH_STRUC_ID.
IMFQ_XQH_VERSION	Contains MQXQH_VERSION_1.
IMFQ_XQH_REMOTEQNAME	Contains a 48-character value.
IMFQ_XQH_REMOTEQMGRNAME	Contains a 48-character value.
IMFQ_XQH_MD_STRUCID	Contains MQMD_STRUC_ID.
IMFQ_XQH_MD_VERSION	Contains MQMD_VERSION_1.
IMFQ_XQH_MD_REPORT	<p>Contains one or more of the following constants delimited by blanks:</p> <p>           MQRO_EXCEPTION            MQRO_EXCEPTION_WITH_DATA            MQRO_EXCEPTION_WITH_FULL_DATA            MQRO_EXPIRATION            MQRO_EXPIRATION_WITH_DATA            MQRO_EXPIRATION_WITH_FULL_DATA            MQRO_COA            MQRO_COA_WITH_DATA            MQRO_COA_WITH_FULL_DATA            MQRO_COD            MQRO_COD_WITH_DATA            MQRO_COD_WITH_FULL_DATA            MQRO_PAN            MQRO_NAN            MQRO_NEW_MSG_ID            MQRO_PASS_MSG_ID            MQRO_COPY_MSG_ID_TO_CORREL_ID            MQRO_PASS_CORREL_ID            MQRO_DEAD_LETTER_Q            MQRO_DISCARD_MSG            MQRO_NONE         </p> <p><i>See note at the beginning of this table.</i></p>

Table 26. Transmission Queue Header Variables – Rules and Rule-Invoked EXECs

Variable Name	Notes
IMFQ_XQH_MD_MSGTYPE	<p>Contains one of the following constants:</p> <p style="padding-left: 40px;">MQMT_DATAGRAM MQMT_REQUEST MQMT_REPLY MQMT_REPORT</p> <p><i>See note at the beginning of this table.</i></p>
IMFQ_XQH_MD_EXPIRY	<p>Contains a decimal between 1 and 999999999 or MQEI_UNLIMITED.</p>
IMFQ_XQH_MD_FEEDBACK	<p>Contains one of the following codes for report messages:</p> <p style="padding-left: 40px;">MQFB_NONE MQFB_COA MQFB_COD MQFB_EXPIRATION MQFB_PAN MQFB_NAN MQFB_QUIT</p> <p>In addition, this field can contain any reason code from the following sources:</p> <ul style="list-style-type: none"> <li>• IMS-bridge feedback codes</li> <li>• CICS-bridge feedback codes</li> <li>• MQSeries reason codes</li> </ul> <p><i>See note at the beginning of this table.</i></p>
IMFQ_XQH_MD_ENCODING	<p>Contains MQENC_NATIVE or a decimal up to 999999999.</p>
IMFQ_XQH_MD_CODEDCHARSETID	<p>Contains</p> <p style="padding-left: 40px;">MQCCSI_Q_MGR MQCCSI_EMBEDDED</p> <p>Or a decimal up to 999999999. <i>See IBM MQSeries Application Programming Reference Guide for details on how the CodedCharSetId field is used as input.</i></p>

Table 26. Transmission Queue Header Variables – Rules and Rule-Invoked EXECs

Variable Name	Notes
IMFQ_XQH_MD_FORMAT	<p>Contains one of the following formats:</p> <p>MQFMT_NONE  MQFMT_ADMIN  MQFMT_CHANNEL_COMPLETED  MQFMT_CICS  MQFMT_COMMAND_1  MQFMT_COMMAND_2  MQFMT_DEAD_LETTER_HEADER  MQFMT_EVENT  MQFMT_IMS  MQFMT_IMS_VAR_STRING  MQFMT_MD_EXTENSION  MQFMT_PCF  MQFMT_REF_MSG_HEADER  MQFMT_STRING  MQFMT_TRIGGER  MQFMT_WORK_INFO_HEADER  MQFMT_XMIT_Q_HEADER  MQFMT_RF_HEADER  MQFMT_RF_HEADER_2</p> <p><i>See note at the beginning of this table.</i></p>
IMFQ_XQH_MD_PRIORITY	Contains a decimal between 0 and 999999999.
IMFQ_XQH_MD_PERSISTENCE	<p>Contains one of the following values:</p> <p>MQPER_PERSISTENT  MQPER_NOT_PERSISTENT</p>
IMFQ_XQH_MD_MSGID	Contains a 32-byte MsgId. No conversion of hexadecimal data is done.
IMFQ_XQH_MD_CORRELID	Contains a 32-byte CorrelId. No conversion of hexadecimal data is done.
IMFQ_XQH_MD_BACKOUTCOUNT	Contains a decimal between 0 and 255.
IMFQ_XQH_MD_REPLYTOQ	Contains up to a 48-character name of the ReplyToQ.
IMFQ_XQH_MD_REPLYTOQMGR	Contains up to a 48-character name of the ReplyToQMgr.
IMFQ_XQH_MD_USERIDENTIFIER	Contains up to a 12-character UserIdentifier.
IMFQ_XQH_MD_ACCOUNTINGTOKEN	Contains MQACT_NONE or up to a 32-byte AccountingToken. It is processed exactly as received from the GET. No conversion of hexadecimal data is done.
IMFQ_XQH_MD_APPLIDENTITYDATA	Contains up to a 32-character value for ApplIdentityData.

Table 26. Transmission Queue Header Variables – Rules and Rule-Invoked EXECs

Variable Name	Notes
IMFQ_XQH_MD_PUTAPPLTYPE	<p>Contains a user-defined type or one of the following standard types:</p> <p style="text-align: center;">                     MQAT_AIX                      MQAT_CICS                      MQAT_CICS_BRIDGE                      MQAT_CICS_VSE                      MQAT_DOS                      MQAT_GUARDIAN                      MQAT_IMS                      MQAT_IMS_BRIDGE                      MQAT_MVS                      MQAT_NOTES_AGENT                      MQAT_NSK                      MQAT_OS2                      MQAT_OS390                      MQAT_OS400                      MQAT_QMGR                      MQAT_UNIX                      MQAT_VMS                      MQAT_VOS                      MQAT_WINDOWS                      MQAT_WINDOWS_NT                      MQAT_XCF                      MQAT_UNKNOWN                 </p> <p><i>See note at the beginning of this table.</i></p>
IMFQ_XQH_MD_PUTAPPLNAME	Contains up to a 28-character value for PutApplName.
IMFQ_XQH_MD_PUTDATE	Contains an 8-character date stamp.
IMFQ_XQH_MD_PUTTIME	Contains an 8-character time stamp.
IMFQ_XQH_MD_APPLORIGINDATA	Contains a 4-character value for ApplOriginData.
IMFQ_XQH_MD_GROUPID	If an MQMD_VERSION_2 MD is present, this variable may contain a 24-byte GroupId. It is processed exactly as received from the GET. No conversion of hexadecimal data is done.
IMFQ_XQH_MD_MSGSEQNUMBER	If an MQMD_VERSION_2 MD is present, this variable may contain a decimal between 1 and 999999999.
IMFQ_XQH_MD_OFFSET	If an MQMD_VERSION_2 MD is present, this variable may contain a decimal between 0 and 999999999.

Table 26. Transmission Queue Header Variables – Rules and Rule-Invoked EXECs

Variable Name	Notes
IMFQ_XQH_MD_MSGFLAGS	<p>If an MQMD_VERSION_2 MD is present, this variable may contain one or more of the following character strings delimited by blanks:</p> <p style="text-align: center;">MQMF_SEGMENTATION_INHIBITED                      MQMF_SEGMENTATION_ALLOWED                      MQMF_MSG_IN_GROUP                      MQMF_LAST_MSG_IN_GROUP                      MQMF_SEGMENT                      MQMF_LAST_SEGMENT                      MQMF_NONE</p> <p><i>See note at the beginning of this table.</i></p>
IMFQ_XQH_MD_ORIGINALLENGTH	<p>Contains a decimal between 1 and 999999999 or MQOL_UNDEFINED.</p> <p><i>See note at the beginning of this table.</i></p>

## Message Descriptor Extension Variables – Rules and Rule-Invoked EXECs

This table lists the Message Descriptor Extension variables – Rules and Rule-Invoked EXECs.

Table 27. Message Descriptor Extension Variables – Rules and Rule-Invoked EXECs

Variable Name	Notes
<p><b>Note:</b> This list may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i>.</p>	
IMFQ_MDE_STRUCID	Contains the value MQMDE_STRUC_ID.
IMFQ_MDE_VERSION	Contains the value MQMDE_VERSION_2.
IMFQ_MDE_STRUCLength	Contains the value MQMDE_LENGTH_2.
IMFQ_MDE_ENCODING	Contains decimal data.
IMFQ_MDE_CODEDCHARSETID	Contains decimal data.
IMFQ_MDE_FORMAT	<p>Contains one of the following formats:</p> <p>MQFMT_NONE  MQFMT_ADMIN  MQFMT_CHANNEL_COMPLETED  MQFMT_CICS  MQFMT_COMMAND_1  MQFMT_COMMAND_2  MQFMT_DEAD_LETTER_HEADER  MQFMT_EVENT  MQFMT_IMS  MQFMT_IMS_VAR_STRING  MQFMT_MD_EXTENSION  MQFMT_PCF  MQFMT_REF_MSG_HEADER  MQFMT_STRING  MQFMT_TRIGGER  MQFMT_WORK_INFO_HEADER  MQFMT_XMIT_Q_HEADER  MQFMT_RF_HEADER  MQFMT_RF_HEADER_2</p> <p><i>See note at the beginning of this table.</i></p>
IMFQ_MDE_FLAGS	Contains MQMDEF_NONE.
IMFQ_MDE_GROUPID	Contains a 24-byte GroupId. No conversion of hexadecimal data is done.
IMFQ_MDE_MSGSEQNUMBER	Contains a decimal between 1 and 999999999.
IMFQ_MDE_OFFSET	Contains a decimal between 0 and 999999999.

Table 27. Message Descriptor Extension Variables – Rules and Rule-Invoked EXECs

Variable Name	Notes
IMFQ_MDE_MSGFLAGS	<p>Contains one or more of the following character strings delimited by blanks:</p> <p style="text-align: center;">MQMF_SEGMENTATION_INHIBITED  MQMF_SEGMENTATION_ALLOWED  MQMF_MSG_IN_GROUP  MQMF_LAST_MSG_IN_GROUP  MQMF_SEGMENT  MQMF_LAST_SEGMENT  MQMF_NONE</p> <p><i>See note at the beginning of this table.</i></p>
IMFQ_MDE_ORIGINALLENGTH	<p>Contains a decimal between 1 and 999999999 or MQOL_UNDEFINED.</p>

## Work Information Header Variables – Rules and Rule-Invoked EXECs

This table lists the Work Information Header variables – Rules and Rule-Invoked EXECs.

Table 28. Work Information Header Variables – Rules and Rule-Invoked EXECs

Variable Name	Notes
IMFQ_WIH_STRUCID	Contains MQWIH_STRUC_ID.
IMFQ_WIH_VERSION	Contains MQWIH_VERSION_1.
IMFQ_WIH_STRUCLength	Contains MQWIH_LENGTH_1.
IMFQ_WIH_ENCODING	Contains decimal data.
IMFQ_WIH_CODEDCHARSETID	Contains decimal data.
IMFQ_WIH_FORMAT	Contains one of the following formats: MQFMT_NONE MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PCF MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2
IMFQ_WIH_FLAGS	Contains the value MQWIH_NONE.
IMFQ_WIH_SERVICENAME	Contains a 32-character value.
IMFQ_WIH_SERVICESTEP	Contains an 8-character value.
IMFQ_WIH_MSGTOKEN	Contains a 16-byte hexadecimal. No conversion of hexadecimal data is done.
IMFQ_WIH_RESERVED	Contains blanks.

## Reference Message Header Variables – Rules and Rule-Invoked EXECs

This table lists the Reference Message Header variables – Rules and Rule-Invoked EXECs.

Table 29. Reference Message Header Variables – Rules and Rule-Invoked EXECs

Variable Name	Notes
IMFQ_RMH_STRUCID	Contains the value MQRMH_STRUC_ID.
IMFQ_RMH_VERSION	Contains the value MQRMH_VERSION_1.
IMFQ_RMH_STRUCLENGTH	Contains decimal data.
IMFQ_RMH_ENCODING	Contains decimal data.
IMFQ_RMH_CODEDCHARSETID	Contains decimal data.
IMFQ_RMH_FORMAT	Contains one of the following format constants:  MQFMT_NONE (IBM default) MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PCF MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2
IMFQ_RMH_FLAGS	Contains one of the following values:  MQRMHF_LAST MQRMHF_NOT_LAST
IMFQ_RMH_OBJECTTYPE	Contains an 8-character value.
IMFQ_RMH_OBJECTINSTANCEID	Contains a 24-byte hexadecimal. No conversion of hexadecimal data is done.
IMFQ_RMH_SRCENVLENGTH	Contains decimal data.
IMFQ_RMH_SRCENVOFFSET	Contains decimal data.
IMFQ_RMH_SRCNAMELENGTH	Contains decimal data.
IMFQ_RMH_SRCNAMEOFFSET	Contains decimal data.
IMFQ_RMH_DESTENVLENGTH	Contains decimal data.

Table 29. Reference Message Header Variables – Rules and Rule-Invoked EXECs

Variable Name	Notes
IMFQ_RMH_DESTENVOFFSET	Contains decimal data.
IMFQ_RMH_DESTNAMELENGTH	Contains decimal data.
IMFQ_RMH_DESTNAMEOFFSET	Contains decimal data.
IMFQ_RMH_DATALOGICALENGTH	Contains decimal data.
IMFQ_RMH_DATALOGICALOFFSET	Contains decimal data.
IMFQ_RMH_DATALOGICALOFFSET2	Contains decimal data.

---

## Trigger Message 2 (Character Format) Variables – Rules and Rule-Invoked EXECs

This table lists the trigger message 2 (character format) variables – Rules and Rule-Invoked EXECs.

Table 30. Trigger Message 2 (Character format) Variables – Rules and Rule-Invoked EXECs

Variable Name	Notes
IMFQ_TMC_STRUCID	Contains MQTMC_STRUC_ID.
IMFQ_TMC_VERSION	Contains MQTMC_VERSION_1.
IMFQ_TMC_QNAME	Contains 48-character name of triggered queue.
IMFQ_TMC_PROCESSNAME	Contains 48-character name of process object.
IMFQ_TMC_TRIGGERDATA	Contains 64-character free-format trigger data.
IMFQ_TMC_APPLTYPE	Contains 4 blanks.
IMFQ_TMC_APPLID	Contains a 256-character string.
IMFQ_TMC_ENVDATA	Contains a 128-character string.
IMFQ_TMC_USERDATA	Contains a 128-character string.
IMFQ_TMC_QMGRNAME	Contains a 48-character value.

## Rules and Formatting Header Variables – Rules and Rule-Invoked EXECs

This table lists the Rules and formatting header variables – Rules and Rule-Invoked EXECs.

Table 31. Rules and Formatting Header Variables – Rules and Rule-Invoked EXECs

Variable Name	Notes
<p><b>Note:</b> This list may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i>.</p>	
IMFQ_RFH_STRUCID	Contains MQRFH_STRUC_ID.
IMFQ_RFH_VERSION	Contains MQRFH_VERSION_1
IMFQ_RFH_STRUCLength	Contains decimal data.
IMFQ_RFH_ENCODING	Contains decimal data or MQENC_NATIVE.
IMFQ_RFH_CODEDCHARSETID	Contains decimal data.
IMFQ_RFH_FORMAT	<p>Contains one of the following formats:</p> <p>MQFMT_NONE  MQFMT_ADMIN  MQFMT_CHANNEL_COMPLETED  MQFMT_CICS  MQFMT_COMMAND_1  MQFMT_COMMAND_2  MQFMT_DEAD_LETTER_HEADER  MQFMT_EVENT  MQFMT_IMS  MQFMT_IMS_VAR_STRING  MQFMT_MD_EXTENSION  MQFMT_PCF  MQFMT_REF_MSG_HEADER  MQFMT_STRING  MQFMT_TRIGGER  MQFMT_WORK_INFO_HEADER  MQFMT_XMIT_Q_HEADER  MQFMT_RF_HEADER  MQFMT_RF_HEADER_2</p>
IMFQ_RFH_FLAGS	<p>Contains following character string delimited by blanks:</p> <p>MQRFH_NONE.</p> <p><i>See note at the beginning of this table.</i></p>
IMFQ_RFH_NAMEVALUESTRING	<p>Contains a character string that is the length of IMFQ_RFH_STRUCLength minus MQRMH_STRUC_LENGTH_FIXED.</p> <p><i>See the IBM MQSeries Application Programming Reference Guide for information about these constants.</i></p>

## Rules and Formatting Header Variables Version 2 – Rules and Rule-Invoked EXECs

This table lists the Rules and formatting header variables version 2 – Rules and Rule-Invoked EXECs.

Table 32. Rules and Formatting Header Variables Version 2 – Rules and Rule-Invoked EXECs

Variable Name	Notes
<p><b>Note:</b> This list may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i>.</p>	
IMFQ_RFH2_STRUCID	Contains the constant MQRFH_STRUC_ID.
IMFQ_RFH2_VERSION_2	Contains the constant MQRFH_VERSION_2.
IMFQ_RFH2_STRUCLength	Contains decimal data.
IMFQ_RFH2_ENCODING	Contains decimal data or the constant MQENC_NATIVE.
IMFQ_RFH2_CODEDCHARSETID	Contains decimal data.
IMFQ_RFH2_FORMAT	<p>Contains one of the following formats:</p> <p>MQFMT_NONE  MQFMT_ADMIN  MQFMT_CHANNEL_COMPLETED  MQFMT_CICS  MQFMT_COMMAND_1  MQFMT_COMMAND_2  MQFMT_DEAD_LETTER_HEADER  MQFMT_EVENT  MQFMT_IMS  MQFMT_IMS_VAR_STRING  MQFMT_MD_EXTENSION  MQFMT_PCF  MQFMT_REF_MSG_HEADER  MQFMT_STRING  MQFMT_TRIGGER  MQFMT_WORK_INFO_HEADER  MQFMT_XMIT_Q_HEADER  MQFMT_RF_HEADER  MQFMT_RF_HEADER_2</p>
IMFQ_RFH2_FLAGS	<p>Contains the following character string delimited by blanks:</p> <p>MQRFH_NONE.</p> <p><i>See note at the beginning of this table.</i></p>
IMFQ_RFH2_NAMEVALUECCSID	Contains decimal data.
IMFQ_RFH2_NAMEVALUEDATA	Contains the exact string of data contained in the message buffer in the position following the NameValueCCSID field.

## PCF Variables – Rules and Rule-Invoked EXECs

This table lists the PCF variables – Rules and Rule-Invoked EXECs.

Table 33. PCF Variables – Rules and Rule-Invoked EXECs

Variable Name	Notes
<b>Note:</b>	This list may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i> .
IMFQ_CFH_TYPE	Contains one or more of the following character strings delimited by blanks:  MQCFT_EVENT MQCFT_COMMAND  <i>See note at the beginning of this table.</i>
IMFQ_CFH_STRUCLength	Contains decimal data.
IMFQ_CFH_VERSION	Contains the constant MQCFH_VERSION_1.
IMFQ_CFH_COMMAND	Contains one of the following constants:  MQCMD_Q_MGR_EVENT MQCMD_PERFM_EVENT MQCMD_CHANNEL_EVENT  <i>See note at the beginning of this table.</i>
IMFQ_CFH_MSGSEQNUMBER	Contains a decimal.
IMFQ_CFH_CONTROL	Contains the constant MQCFC_LAST.
IMFQ_CFH_COMPCODE	Contains one of the following constants:  MQCC_OK MQCC_WARNING  <i>See note at the beginning of this table.</i>
IMFQ_CFH_REASON	Contains a numeric reason code between 0 and 999999999.
IMFQ_CFH_PARAMETERCOUNT	Contains decimal data.



---

## Appendix D. EXECs Variables

---

### General Purpose Variables – EXECs

This table lists the general purpose variable.

Table 34. General Purpose Variables - EXECs

Variable Name	Data Type	Notes
IMFMQI_STRUCTURES	Character	Contains names of all structures in a message, for example: MD XQH.
IMFMQI_OFFSETS	Character	Contains a blank-delimited list of offsets to structures and application data from the beginning of the message buffer. For example: if the message buffer contains a Dead Letter Header (DLH) and application data, the variable value may look like this: 'NONE 0 172', indicating NONE for MD (since it is not in the message buffer), 0 for the DLH (first structure in the buffer) and 172 for application data (follows the DLH).
IMFMQI_REASON	Character	Contains the constant reason code from the IMFEXEC MQI statement and the IMFEXEC CMD ... TYPE(MQS) command. For example, if IMFMQRC = 2033, IMFMQI_REASON = 'MQRC_NO_MSG_AVAILABLE'

---

### Options and Miscellaneous Variables

This table lists the options and miscellaneous variables.

Table 35. Options and Miscellaneous Variables

Variable	Description	Input/Output	Call Type(s)
IMFMQI_CO_OPTIONS	Close options	Input	CLOSE
IMFMQI_GMO_OPTIONS	Get options	Input	GET
IMFMQI_PMO_OPTIONS	Put options	Input	PUT, PUT1
IMFMQI_OO_OPTIONS	Open options	Input	OPEN
IMFMQI_BUFFER	Message buffer	Input	GET, PUT, PUT1
IMFMQI_BUFFLEN	Message Buffer Length	Input	GET, PUT, PUT1
IMFMQI_DATALEN	Length of returned message	Output	GET

## EVENT Variables – EXECs

This table lists the event variables – EXECs.

Table 36. Event Variables - EXECs

Variable Name	Notes
IMFMQI_EVENT_TYPE	Event type-not an MQSeries field
IMFMQI_EVENT_QMGRNAME	Name of queue manager that generated event, whether local or remote
IMFMQI_EVENT_QNAME	Name of queue from Object Descriptor
IMFMQI_EVENT_BASEQNAME	Name of queue manager (to which the alias resolves)
IMFMQI_EVENT_APPLNAME	Name of application
IMFMQI_EVENT_OBJECTQMGRNAME	Name of the object queue manager
IMFMQI_EVENT_CHANNEL_NAME	Name of channel
IMFMQI_EVENT_CONNECTIONNAME	Name of connection. For TCP/IP, this is the internet address when the channel has established a connection successfully. Otherwise, it is the contents of the ConnectionName field in the channel definition.
IMFMQI_EVENT_XMITQNAME	Transmission queue name
IMFMQI_EVENT_FORMAT	Contains one of the following format constants:  MQFMT_NONE (IBM default) MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PCF MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2
IMFMQI_EVENT_QTYPE	Type of queue
IMFMQI_EVENT_APPLTYPE	Type of queue to which the alias resolves (A for alias, M for model)

Table 36. Event Variables - EXECs

Variable Name	Notes
IMFMQI_EVENT_REASONQUALIFIER	Reason qualifier: MQRQ_CHANNEL_STOPPED_OK MQRQ_CHANNEL_STOPPED_ERROR MQRQ_CHANNEL_STOPPED_RETRY MQRQ_CHANNEL_STOPPED_DISABLED
IMFMQI_EVENT_ERRORIDENTIFIER	See the section regarding the CHANNEL_STOPPED event, field Error identifier, in the IBM publication <i>MQSeries Programmable System Management</i> for a more detail description.
IMFMQI_EVENT_ERRORMSGID	Last three characters of IMFMQI_EVENT_ERRORIDENTIFIER (ErrorIdentifier). Use this number to look up the message ID in the “Distributed Queuing Message Codes” section of the <i>IBM MQSeries for MVS/ESA Messages and Codes</i> manual.
IMFMQI_EVENT_PROCESSNAME	Name of process
IMFMQI_EVENT_USERIDENTIFIER	Name of user identifier
IMFMQI_EVENT_AUXERRORDATAINT1	First integer of auxiliary error data for channel errors
IMFMQI_EVENT_AUXERRORDATAINT2	Second integer of auxiliary error data for channel errors
IMFMQI_EVENT_AUXERRORDATASTR1	First string of auxiliary error data for channel errors
IMFMQI_EVENT_AUXERRORDATASTR2	Second string of auxiliary error data for channel errors
IMFMQI_EVENT_AUXERRORDATASTR3	Third string of auxiliary error data for channel errors
IMFMQI_EVENT_TIMESINCERESET	Time (in seconds) since the statistics were last reset; for this event, this is greater than the service interval
IMFMQI_EVENT_HIGHQDEPTH	Max number of message on the queue since the statistics were last reset
IMFMQI_EVENT_MSGENQCOUNT	Number of messages enqueued since statistics were last reset
IMFMQI_EVENT_MSGDEQCOUNT	Number of messages dequeued since statistics were last reset
IMFMQI_EVENT_BRIDGENAME	Name of bridge
IMFMQI_EVENT_BRIDGETYPE	Only current value is 1 (OTMA)
IMFMQI_EVENT_OPTIONS	Open options
IMFMQI_EVENT_COMMAND	Name of command event
IMFMQI_EVENT_CURDEPTH	Current depth of the event queue
IMFMQI_EVENT_MAXDEPTH	Maximum depth of the event queue
IMFMQI_EVENT_PERCENTFULL	Percent full of the event queue

## Dead Letter Header Variables – EXECs

This table lists the Dead Letter Header variables - EXECs.

Table 37. Dead Letter Header Variables - EXECs

Variable Name	Notes
<b>Note:</b>	This list may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i> .
IMFMQI_DLH_STRUCID	Contains the valueMQDLH_STRUC_ID.
IMFMQI_DLH_VERSION	Contains the valueMQDLH_VERSION_1.
IMFMQI_DLH_REASON	Contains a numeric reason code between 0 and 999999999.
IMFMQI_DLH_DESTQNAME	Contains the 48-character destination queue name.
IMFMQI_DLH_DESTQMGRNAME	Contains the 48-character destination queue manager name.
IMFMQI_DLH_ENCODING	Contains decimal data.
IMFMQI_DLH_CODEDCHARSETID	Contains decimal data.
IMFMQI_DLH_FORMAT	Contains one of the following format constants: <ul style="list-style-type: none"> <li>MQFMT_NONE (IBM default)</li> <li>MQFMT_ADMIN</li> <li>MQFMT_CHANNEL_COMPLETED</li> <li>MQFMT_CICS</li> <li>MQFMT_COMMAND_1</li> <li>MQFMT_COMMAND_2</li> <li>MQFMT_DEAD_LETTER_HEADER</li> <li>MQFMT_EVENT</li> <li>MQFMT_IMS</li> <li>MQFMT_IMS_VAR_STRING</li> <li>MQFMT_MD_EXTENSION</li> <li>MQFMT_PC</li> <li>MQFMT_REF_MSG_HEADER</li> <li>MQFMT_STRING</li> <li>MQFMT_TRIGGER</li> <li>MQFMT_WORK_INFO_HEADER</li> <li>MQFMT_XMIT_Q_HEADER</li> <li>MQFMT_RF_HEADER</li> <li>MQFMT_RF_HEADER_2</li> </ul>

Table 37. Dead Letter Header Variables - EXECs

Variable Name	Notes
IMFMQI_DLH_PUTAPPLTYPE	<p>Contains a numeric user-defined type or one of the following standard types:</p> <p>MQAT_AIX            MQAT_CICS            MQAT_CICS_BRIDGE            MQAT_CICS_VSE            MQAT_DOS            MQAT_GUARDIAN            MQAT_IMS            MQAT_IMS_BRIDGE            MQAT_MVS            MQAT_NOTES_AGENT            MQAT_NSK            MQAT_OS2            MQAT_OS390            MQAT_OS400            MQAT_QMGR            MQAT_UNIX            MQAT_VMS            MQAT_VOS            MQAT_WINDOWS            MQAT_WINDOWS_NT            MQAT_XCF            MQAT_UNKNOWN</p> <p><i>See note at the beginning of this table.</i></p>
IMFMQI_DLH_PUTAPPLNAME	Contains up to a 28-character name for PutApplName.
IMFMQI_DLH_PUTDATE	Contains an 8-character date stamp.
IMFMQI_DLH_PUTTIME	Contains an 8-character time stamp.

## Trigger Message Variables – EXECs

This table lists the trigger message variables - EXECs.

Table 38. Trigger Message Variables - EXECs

Variable Name	Data Type	Notes
<p><b>Note:</b> This this list may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i>.</p>		
IMFMQI_TM_STRUCID	Character	Contains the valueMQTM_STRUC_ID.
IMFMQI_TM_VERSION	Decimal	Contains the valueMQTM_VERSION_1.
IMFMQI_TM_QNAME	Character	Contains 48-character name of the triggered queue.
IMFMQI_TM_PROCESSNAME	Character	Contains 48-character name of the process object.
IMFMQI_TM_TRIGGERDATA	Character	Contains 64-character free-format trigger data.
IMFMQI_TM_APPLTYPE	Decimal	<p>Contains a numeric user-defined type or one of the following standard types:</p> <ul style="list-style-type: none"> <li>MQAT_AIX</li> <li>MQAT_CICS</li> <li>MQAT_CICS_BRIDGE</li> <li>MQAT_CICS_VSE</li> <li>MQAT_DOS</li> <li>MQAT_GUARDIAN</li> <li>MQAT_IMS</li> <li>MQAT_IMS_BRIDGE</li> <li>MQAT_MVS</li> <li>MQAT_NOTES_AGENT</li> <li>MQAT_NSK</li> <li>MQAT_OS2</li> <li>MQAT_OS390</li> <li>MQAT_OS400</li> <li>MQAT_QMGR</li> <li>MQAT_UNIX</li> <li>MQAT_VMS</li> <li>MQAT_VOS</li> <li>MQAT_WINDOWS</li> <li>MQAT_WINDOWS_NT</li> <li>MQAT_XCF</li> <li>MQAT_UNKNOWN</li> </ul> <p><i>See note at the beginning of this table.</i></p>
IMFMQI_TM_APPLID	Character	Contains a 256-character string.
IMFMQI_TM_ENVDATA	Character	Contains a 128-character string.
IMFMQI_TM_USERDATA	Character	Contains a 128-character string.

## CICS Information Header Variables – EXECs

This table lists the CICS Information Header variables - EXECs.

Table 39. CICS Information Header Variables - EXECs

Variable Name	Notes
<b>Note:</b>	This list may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i> .
IMFMQI_CIH_STRUCID	Contains MQCIH_STRUC_ID.
IMFMQI_CIH_VERSION	Contains MQCIH_VERSION_2.
IMFMQI_CIH_STRUCLength	Contains MQCIH_LENGTH_2.
IMFMQI_CIH_ENCODING	Contains the decimal 0.
IMFMQI_CIH_CODEDCHARSETID	Contains the decimal 0.
IMFMQI_CIH_FORMAT	Contains one of the following format constants: <ul style="list-style-type: none"> <li>MQFMT_NONE (IBM default)</li> <li>MQFMT_ADMIN</li> <li>MQFMT_CHANNEL_COMPLETED</li> <li>MQFMT_CICS</li> <li>MQFMT_COMMAND_1</li> <li>MQFMT_COMMAND_2</li> <li>MQFMT_DEAD_LETTER_HEADER</li> <li>MQFMT_EVENT</li> <li>MQFMT_IMS</li> <li>MQFMT_IMS_VAR_STRING</li> <li>MQFMT_MD_EXTENSION</li> <li>MQFMT_PC</li> <li>MQFMT_REF_MSG_HEADER</li> <li>MQFMT_STRING</li> <li>MQFMT_TRIGGER</li> <li>MQFMT_WORK_INFO_HEADER</li> <li>MQFMT_XMIT_Q_HEADER</li> <li>MQFMT_RF_HEADER</li> <li>MQFMT_RF_HEADER_2</li> </ul>
IMFMQI_CIH_FLAGS	Contains MQCIH_NONE.
IMFMQI_CIH_RETURNCODE	Contains one of the following values: <ul style="list-style-type: none"> <li>MQCRC_OK</li> <li>MQCRC_CICS_EXEC_ERROR</li> <li>MQCRC_MQ_API_ERROR</li> <li>MQCRC_BRIDGE_ERROR</li> <li>MQCRC_BRIDGE_ABEND</li> <li>MQCRC_APPLICATION_ABEND</li> <li>MQCRC_SECURITY_ERROR</li> <li>MQCRC_PROGRAM_NOT_AVAILABLE</li> <li>MQCRC_BRIDGE_TIMEOUT</li> <li>MQCRC_TRANSID_NOT_AVAILABLE</li> </ul>

Table 39. CICS Information Header Variables - EXECs

Variable Name	Notes
IMFMQI_CIH_COMPCODE	Contains decimal data.
IMFMQI_CIH_REASON	Contains decimal data.
IMFMQI_CIH_UOWCONTROL	Contains one of the following values: MQCUOWC_ONLY MQCUOWC_CONTINUE MQCUOWC_FIRST MQCUOWC_MIDDLE MQCUOWC_LAST MQCUOWC_COMMIT MQCUOWC_BACKOUT
IMFMQI_CIH_GETWAITINTERVAL	Contains decimal data or one of the following constants: MQCGWI_DEFAULT MQWI_UNLIMITED
IMFMQI_CIH_LINKTYPE	Contains one of the following values: MQCLT_PROGRAM MQCLT_TRANSACTION
IMFMQI_CIH_OUTPUTDATALENGTH	Contains decimal data or the constant MQCLT_TRANSACTION
IMFMQI_CIH_FACILITYKEEPTIME	Contains decimal data.
IMFMQI_CIH_ADSDESCRIPTOR	Contains one of the following values: MQCADSD_NONE MQCADSD_SEND MQCADSD_RECV MQCADSD_MSGFORMAT  <i>See note at the beginning of this table.</i>
IMFMQI_CIH_CONVERSATIONALTASK	Contains one of the following values: MQCCT_YES MQCCT_NO
IMFMQI_CIH_TASKENDSTATUS	Contains one of the following values: MQCTES_NOSYNC MQCTES_COMMIT MQCTES_BACKOUT MQCTES_ENDTASK
IMFMQI_CIH_FACILITY	Contains a 8-byte hexadecimal value or the constant MQCFAC_NONE.  No conversion of hexadecimal data is done.

Table 39. CICS Information Header Variables - EXECs

Variable Name	Notes
IMFMQI_CIH_FUNCTION	<p>Contains a 4-character value of one of the following constants:</p> <p>MQCFUNC_MQCONN  MQCFUNC_MQGET  MQCFUNC_MQINQ  MQCFUNC_MQOPEN  MQCFUNC_MQPUT  MQCFUNC_MQPUT1  MQCFUNC_NONE</p> <p><i>See note at the beginning of this table.</i></p>
IMFMQI_CIH_ABENDCODE	Contains a 4-character value.
IMFMQI_CIH_AUTHENTICATOR	Contains an 8-character value.
IMFMQI_CIH_RESERVED1	Contains 8 blanks.
IMFMQI_CIH_REPLYTOFORMAT	<p>Contains one of the following format constants:</p> <p>MQFMT_NONE (IBM default)  MQFMT_ADMIN  MQFMT_CHANNEL_COMPLETED  MQFMT_CICS  MQFMT_COMMAND_1  MQFMT_COMMAND_2  MQFMT_DEAD_LETTER_HEADER  MQFMT_EVENT  MQFMT_IMS  MQFMT_IMS_VAR_STRING  MQFMT_MD_EXTENSION  MQFMT_PC  MQFMT_REF_MSG_HEADER  MQFMT_STRING  MQFMT_TRIGGER  MQFMT_WORK_INFO_HEADER  MQFMT_XMIT_Q_HEADER  MQFMT_RF_HEADER  MQFMT_RF_HEADER_2</p>
IMFMQI_CIH_REMOTESYSID	Contains 4 blanks.
IMFMQI_CIH_REMOTETRANSID	Contains 4 blanks.
IMFMQI_CIH_TRANSACTIONID	Contains a 4-character value.
IMFMQI_CIH_FACILITYLIKE	Contains a 4-character value.
IMFMQI_CIH_ATTENTIONID	Contains a 4-character value.

Table 39. CICS Information Header Variables - EXECs

Variable Name	Notes
IMFMQI_CIH_STARTCODE	Contains one of the following constants: MQCSC_START MQCSC_STARTDATA MQCSC_TERMINPUT MQCSC_NONE  <i>See note at the beginning of this table.</i>
IMFMQI_CIH_CANCELCODE	Contains a 4-character value.
IMFMQI_CIH_NEXTTRANSACTIONID	Contains a 4-character value.
IMFMQI_CIH_RESERVED2	Contains 8 blanks.
IMFMQI_CIH_RESERVED3	Contains 8 blanks.
IMFMQI_CIH_CURSORPOSITION	Contains decimal data.
IMFMQI_CIH_ERROROFFSET	Contains decimal data.
IMFMQI_CIH_INPUTITEM	Contains the decimal 0.
IMFMQI_CIH_RESERVED4	Contains the decimal 0.

## IMS Information Header Variables – EXECs

This table lists the IMS Information Header variables - EXECs.

Table 40. IMS Information Header Variables - EXECs

Variable Name	Notes
IMFMQI_IIH_STRUCID	Contains MQIIH_STRUC_ID.
IMFMQI_IIH_VERSION	Contains MQIIH_VERSION_1.
IMFMQI_IIH_STRUCLength	Contains MQIIH_LENGTH_1.
IMFMQI_IIH_ENCODING	Contains the decimal 0.
IMFMQI_IIH_CODEDCHARSETID	Contains the decimal 0.
IMFMQI_IIH_FORMAT	Contains one of the following format constants: MQFMT_NONE (IBM default) MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PC MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2
IMFMQI_IIH_FLAGS	Contains MQIIH_NONE.
IMFMQI_IIH_LTERMOVERRIDE	Contains an 8-character value.
IMFMQI_IIH_MFSMAPNAME	Contains an 8-character value.

Table 40. IMS Information Header Variables - EXECs

Variable Name	Notes
IMFMQI_III_REPLYTOFORMAT	Contains one of the following format constants: MQFMT_NONE (IBM default) MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PC MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2
IMFMQI_III_AUTHENTICATOR	Contains an 8-character value or the value MQIAUT_NONE.
IMFMQI_III_TRANINSTANCEID	Contains a 16-byte hexadecimal value or the constant MQITII_NONE. No conversion of hexadecimal data is done.
IMFMQI_III_TRANSTATE	Contains one of the following constants: MQITS_IN_CONVERSATION MQITS_NOT_IN_CONVERSATION MQITS_ARCHITECTED
IMFMQI_III_COMMITMODE	Contains one of the following constants: MQICM_COMMIT_THEN_SEND MQICM_SEND_THEN_COMMIT
IMFMQI_III_SECURITYSCOPE	Contains one of the following constants: MQISS_CHECK MQISS_FULL
IMFMQI_III_RESERVED	Contains 1 blank.

## Transmission Queue Header Variables – EXECs

This table lists the Transmission Queue Header variables - EXECs.

Table 41. Transmission Queue Header Variables - EXECs

Variable Name	Notes
<b>Note:</b>	This list may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i> .
IMFMQI_XQH_STRUCID	Contains MQXQH_STRUC_ID.
IMFMQI_XQH_VERSION	Contains MQXQH_VERSION_1.
IMFMQI_XQH_REMOTEQNAME	Contains a 48-character value.
IMFMQI_XQH_REMOTEQMGRNAME	Contains a 48-character value.
IMFMQI_XQH_MD_STRUCID	Contains MQMD_STRUC_ID.
IMFMQI_XQH_MD_VERSION	Contains MQMD_VERSION_1.
IMFMQI_XQH_MD_REPORT	<p>Contains one or more of the following delimited by blanks:</p> <p style="margin-left: 40px;">MQRO_EXCEPTION  MQRO_EXCEPTION_WITH_DATA  MQRO_EXCEPTION_WITH_FULL_DATA  MQRO_EXPIRATION  MQRO_EXPIRATION_WITH_DATA  MQRO_EXPIRATION_WITH_FULL_DATA  MQRO_COA  MQRO_COA_WITH_DATA  MQRO_COA_WITH_FULL_DATA  MQRO_COD  MQRO_COD_WITH_DATA  MQRO_COD_WITH_FULL_DATA  MQRO_PAN  MQRO_NAN  MQRO_NEW_MSG_ID  MQRO_PASS_MSG_ID  MQRO_COPY_MSG_ID_TO_CORREL_ID  MQRO_PASS_CORREL_ID  MQRO_DEAD_LETTER_Q  MQRO_DISCARD_MSG  MQRO_NONE</p> <p><i>See note at the beginning of this table.</i></p>

Table 41. Transmission Queue Header Variables - EXECs

Variable Name	Notes
IMFMQI_XQH_MD_MSGTYPE	<p>Contains one of the following values:</p> <p>MQMT_DATAGRAM  MQMT_REQUEST  MQMT_REPLY  MQMT_REPORT</p> <p><i>See note at the beginning of this table.</i></p>
IMFMQI_XQH_MD_EXPIRY	<p>Contains a decimal between 1 and 999999999, or MQEI_UNLIMITED.</p>
IMFMQI_XQH_MD_FEEDBACK	<p>Contains one of the following values for report messages:</p> <p>MQFB_NONE  MQFB_COA  MQFB_COD  MQFB_EXPIRATION  MQFB_PAN  MQFB_NAN  MQFB_QUIT</p> <p>In addition, this field can contain a reason code from the following sources:</p> <ul style="list-style-type: none"> <li>• IMS-bridge feedback codes</li> <li>• CICS-bridge feedback codes</li> <li>• MQSeries reason codes</li> </ul> <p><i>See note at the beginning of this table.</i></p>
IMFMQI_XQH_MD_ENCODING	<p>Contains MQENC_NATIVE or any decimal up to 999999999.</p>
IMFMQI_XQH_MD_CODEDCHARSETID	<p>Contains</p> <p>MQCCSI_Q_MGR  MQCCSI_EMBEDDED  Or a decimal up to 999999999.</p> <p><i>See the IBM MQSeries Application Programming Reference Guide for details about how the CodedCharSetId field is used as input.</i></p>

Table 41. Transmission Queue Header Variables - EXECs

Variable Name	Notes
IMFMQI_XQH_MD_FORMAT	<p>Contains one of the following format constants:</p> <p>MQFMT_NONE  MQFMT_ADMIN  MQFMT_CHANNEL_COMPLETED  MQFMT_CICS  MQFMT_COMMAND_1  MQFMT_COMMAND_2  MQFMT_DEAD_LETTER_HEADER  MQFMT_EVENT  MQFMT_IMS  MQFMT_IMS_VAR_STRING  MQFMT_MD_EXTENSION  MQFMT_PCF  MQFMT_REF_MSG_HEADER  MQFMT_STRING  MQFMT_TRIGGER  MQFMT_WORK_INFO_HEADER  MQFMT_XMIT_Q_HEADER  MQFMT_RF_HEADER  MQFMT_RF_HEADER_2</p> <p><i>See note at the beginning of this table.</i></p>
IMFMQI_XQH_MD_PRIORITY	Contains a decimal between 0 and 999999999.
IMFMQI_XQH_MD_PERSISTENCE	<p>Contains one of the following values:</p> <p>MQPER_PERSISTENT  MQPER_NOT_PERSISTENT</p>
IMFMQI_XQH_MD_MSGID	Contains a 32-byte MsgId. No conversion of hexadecimal data is done.
IMFMQI_XQH_MD_CORRELID	Contains a 32-byte CorrelId. No conversion of hexadecimal data is done.
IMFMQI_XQH_MD_BACKOUTCOUNT	Contains a number between 0 and 255.
IMFMQI_XQH_MD_REPLYTOQ	Contains up to a 48-character name of the ReplyToQ.
IMFMQI_XQH_MD_REPLYTOQMGR	Contains up to a 48-character name of the ReplyToQMgr.
IMFMQI_XQH_MD_USERIDENTIFIER	Contains up to a 12-character UserIdentifier.
IMFMQI_XQH_MD_ACCOUNTINGTOKEN	Contains MQACT_NONE or up to a 32-byte AccountingToken. It is processed exactly as received from the GET. No conversion of hexadecimal data is done.
IMFMQI_XQH_MD_APPLIDENTITYDATA	Contains up to a 32-character value for ApplIdentityData.

Table 41. Transmission Queue Header Variables - EXECs

Variable Name	Notes
IMFMQI_XQH_MD_PUTAPPLTYPE	<p>Contains a numeric user-defined type or one of the following standard types:</p> <p>MQAT_AIX            MQAT_CICS            MQAT_CICS_BRIDGE            MQAT_CICS_VSE            MQAT_DOS            MQAT_GUARDIAN            MQAT_IMS            MQAT_IMS_BRIDGE            MQAT_MVS            MQAT_NOTES_AGENT            MQAT_NSK            MQAT_OS2            MQAT_OS390            MQAT_OS400            MQAT_QMGR            MQAT_UNIX            MQAT_VMS            MQAT_VOS            MQAT_WINDOWS            MQAT_WINDOWS_NT            MQAT_XCF            MQAT_UNKNOWN</p> <p><i>See note at the beginning of this table.</i></p>
IMFMQI_XQH_MD_PUTAPPLNAME	Contains up to a 28-character value for PutApplName.
IMFMQI_XQH_MD_PUTDATE	Contains an 8-character date stamp.
IMFMQI_XQH_MD_PUTTIME	Contains an 8-character time stamp.
IMFMQI_XQH_MD_APPLORIGINDATA	Contains a 4-character value for ApplOriginData.
IMFMQI_XQH_MD_GROUPID	If an MQMD_VERSION_2 MD is present, this variable may contain a 24-byte GroupId. It is processed exactly as received from the GET. No conversion of hexadecimal data is done.
IMFMQI_XQH_MD_MSGSEQNUMBER	If an MQMD_VERSION_2 MD is present, this variable may contain a decimal between 1 and 999999999.
IMFMQI_XQH_MD_OFFSET	If an MQMD_VERSION_2 MD is present, this variable may contain a decimal between 0 and 999999999.

Table 41. Transmission Queue Header Variables - EXECs

Variable Name	Notes
IMFMQI_XQH_MD_MSGFLAGS	<p>If an MQMD_VERSION_2 MD is present, this variable may contain one or more of the following character strings delimited by blanks:</p> <p style="padding-left: 40px;">MQMF_SEGMENTATION_INHIBITED            MQMF_SEGMENTATION_ALLOWED            MQMF_MSG_IN_GROUP            MQMF_LAST_MSG_IN_GROUP            MQMF_SEGMENT            MQMF_LAST_SEGMENT            MQMF_NONE</p> <p><i>See note at the beginning of this table.</i></p>
IMFMQI_XQH_MD_ORIGINALLENGTH	<p>Contains a decimal between 1 and 999999999 or MQOL_UNDEFINED.</p> <p><i>See note at the beginning of this table.</i></p>

## Message Descriptor Extension Variables – EXECs

This table lists the Message Descriptor Extension variables - EXECs.

Table 42. Message Descriptor Extension Variables - EXECs

Variable Name	Notes
<b>Note:</b>	This list may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i> .
IMFMQI_MDE_STRUCID	Contains MQMDE_STRUC_ID.
IMFMQI_MDE_VERSION	Contains MQMDE_VERSION_2.
IMFMQI_MDE_STRUCLength	Contains MQMDE_LENGTH_2.
IMFMQI_MDE_ENCODING	Contains decimal data.
IMFMQI_MDE_CODEDCHARSETID	Contains decimal data.
IMFMQI_MDE_FORMAT	Contains one of the following format constants: <ul style="list-style-type: none"> <li>MQFMT_NONE</li> <li>MQFMT_ADMIN</li> <li>MQFMT_CHANNEL_COMPLETED</li> <li>MQFMT_CICS</li> <li>MQFMT_COMMAND_1</li> <li>MQFMT_COMMAND_2</li> <li>MQFMT_DEAD_LETTER_HEADER</li> <li>MQFMT_EVENT</li> <li>MQFMT_IMS</li> <li>MQFMT_IMS_VAR_STRING</li> <li>MQFMT_MD_EXTENSION</li> <li>MQFMT_PCF</li> <li>MQFMT_REF_MSG_HEADER</li> <li>MQFMT_STRING</li> <li>MQFMT_TRIGGER</li> <li>MQFMT_WORK_INFO_HEADER</li> <li>MQFMT_XMIT_Q_HEADER</li> <li>MQFMT_RF_HEADER</li> <li>MQFMT_RF_HEADER_2</li> </ul>
IMFMQI_MDE_FLAGS	Contains MQMDEF_NONE.
IMFMQI_MDE_GROUPID	Contains a 24-byte GroupId. No conversion of hexadecimal data is done.
IMFMQI_MDE_MSGSEQNUMBER	Contains a decimal between 1 and 999999999.
IMFMQI_MDE_OFFSET	Contains a decimal between 0 and 999999999.

Table 42. Message Descriptor Extension Variables - EXECs

Variable Name	Notes
IMFMQI_MDE_MSGFLAGS	<p>Contains one or more of the following character strings delimited by blanks:</p> <p style="margin-left: 40px;">MQMF_SEGMENTATION_INHIBITED            MQMF_SEGMENTATION_ALLOWED            MQMF_MSG_IN_GROUP            MQMF_LAST_MSG_IN_GROUP            MQMF_SEGMENT            MQMF_LAST_SEGMENT            MQMF_NONE</p> <p><i>See note at the beginning of this table.</i></p>
IMFMQI_MDE_ORIGINALLENGTH	<p>Contains a decimal between 1 and 999999999 or MQOL_UNDEFINED.</p>

## Work Information Header Variables – EXECs

This table lists the Work Information Header variables - EXECs.

Table 43. Work Information Header Variables - EXECs

Variable Name	Notes
IMFMQI_WIH_STRUCID	Contains the value MQWIH_STRUC_ID.
IMFMQI_WIH_VERSION	Contains the value MQWIH_VERSION_1.
IMFMQI_WIH_STRUCLength	Contains the value MQWIH_LENGTH_1.
IMFMQI_WIH_ENCODING	Contains decimal data.
IMFMQI_WIH_CODEDCHARSETID	Contains decimal data.
IMFMQI_WIH_FORMAT	Contains one of the following format constants: MQFMT_NONE MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PCF MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2
IMFMQI_WIH_FLAGS	Contains the value MQWIH_NONE.
IMFMQI_WIH_SERVICENAME	Contains a 32-character value.
IMFMQI_WIH_SERVICESTEP	Contains an 8-character value.
IMFMQI_WIH_MSGTOKEN	Contains a 16-byte hexadecimal value. No conversion of hexadecimal data is done.
IMFMQI_WIH_RESERVED	Contains blanks.

## Reference Message Header Variables – EXECs

This table lists the Reference Message Header variables - EXECs.

Table 44. Reference Message Header Variables - EXECs

Variable Name	Note
IMFMQI_RMH_STRUCID	Contains MQRMH_STRUC_ID.
IMFMQI_RMH_VERSION	Contains MQRMH_VERSION_1.
IMFMQI_RMH_STRUCLENGTH	Contains decimal data.
IMFMQI_RMH_ENCODING	Contains decimal data.
IMFMQI_RMH_CODEDCHARSETID	Contains decimal data.
IMFMQI_RMH_FORMAT	Contains one of the following format constants: MQFMT_NONE MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PCF MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2
IMFMQI_RMH_FLAGS	Contains one of the following values: MQRMHF_LAST MQRMHF_NOT_LAST
IMFMQI_RMH_OBJECTTYPE	Contains an 8-character value.
IMFMQI_RMH_OBJECTINSTANCEID	Contains a 24-byte hexadecimal value. No conversion of hexadecimal data is done.
IMFMQI_RMH_SRCENVLENGTH	Contains decimal data.
IMFMQI_RMH_SRCENVOFFSET	Contains decimal data.
IMFMQI_RMH_SRCNAMELENGTH	Contains decimal data.
IMFMQI_RMH_SRCNAMEOFFSET	Contains decimal data.
IMFMQI_RMH_DESTENVLENGTH	Contains decimal data.

Table 44. Reference Message Header Variables - EXECs

Variable Name	Note
IMFMQI_RMH_DESTENVOFFSET	Contains decimal data.
IMFMQI_RMH_DESTNAMELENGTH	Contains decimal data.
IMFMQI_RMH_DESTNAMEOFFSET	Contains decimal data.
IMFMQI_RMH_DATALOGICALENGTH	Contains decimal data.
IMFMQI_RMH_DATALOGICALOFFSET	Contains decimal data.
IMFMQI_RMH_DATALOGICALOFFSET2	Contains decimal data.

---

## Trigger Message 2 (Character format) Variables – EXECs

This table lists the trigger message 2 (character format) variables – EXECs.

Table 45. Trigger Message 2 (Character format) Variables – EXECs

Variable Name	Notes
IMFMQI_TMC_STRUCID	Contains MQTMC_STRUC_ID.
IMFMQI_TMC_VERSION	Contains MQTMC_VERSION_1.
IMFMQI_TMC_QNAME	Contains 48-character name of the triggered queue.
IMFMQI_TMC_PROCESSNAME	Contains 48-character name of the process object.
IMFMQI_TMC_TRIGGERDATA	Contains 64-character free-format trigger data.
IMFMQI_TMC_APPLTYPE	Contains 4 blanks.
IMFMQI_TMC_APPLID	Contains a 256-character string.
IMFMQI_TMC_ENVDATA	Contains a 128-character string.
IMFMQI_TMC_USERDATA	Contains a 128-character string.
IMFMQI_TMC_QMGRNAME	Contains a 48-character value.

## Rules and Formatting Header Variables – EXECs

This table lists the Rules and formatting header variables – EXECs.

Table 46. Rules and Formatting Header Variables – EXECs

Variable Name	Notes
<b>Note:</b> This list may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i> .	
IMFMQI_RFH_STRUCID	Contains MQRFH_STRUC_ID.
IMFMQI_RFH_VERSION	Contains MQRFH_VERSION_1.
IMFMQI_RFH_STRUCLength	Contains decimal data.
IMFMQI_RFH_ENCODING	Contains decimal data or MQENC_NATIVE.
IMFMQI_RFH_CODEDCHARSETID	Contains decimal data.
IMFMQI_RFH_STRUCID	Contains MQRFH_STRUC_ID.
IMFMQI_RFH_FORMAT	<p>Contains one of the following format constants:</p> <p>MQFMT_NONE  MQFMT_ADMIN  MQFMT_CHANNEL_COMPLETED  MQFMT_CICS  MQFMT_COMMAND_1  MQFMT_COMMAND_2  MQFMT_DEAD_LETTER_HEADER  MQFMT_EVENT  MQFMT_IMS  MQFMT_IMS_VAR_STRING  MQFMT_MD_EXTENSION  MQFMT_PCF  MQFMT_REF_MSG_HEADER  MQFMT_STRING  MQFMT_TRIGGER  MQFMT_WORK_INFO_HEADER  MQFMT_XMIT_Q_HEADER  MQFMT_RF_HEADER  MQFMT_RF_HEADER_2</p>
IMFMQI_RFH_FLAGS	<p>Contains the following character string delimited by blanks:  MQRFH_NONE.</p> <p><i>See note at the beginning of this table.</i></p>
IMFMQI_RFH_NAMEVALUESTRING	<p>Contains a character string of the length of  IMFQ_RFH_STRUCLength minus  MQRMH_STRUC_LENGTH_FIXED.</p> <p><i>See note at the beginning of this table.</i></p>

## Rules and Formatting Header Variables Version 2 – EXECs

This table lists the Rules and formatting header variables version 2 – EXECs.

Table 47. Rules and Formatting Header Variables Version 2 – EXECs

Variable Name	Notes
<b>Note:</b>	This list may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i> .
IMFMQI_RFH2_STRUCID	Contains MQRFH_STRUC_ID.
IMFMQI_RFH2_VERSION_2	Contains MQRFH_VERSION_2.
IMFMQI_RFH2_STRUCLNGTH	Contains decimal data.
IMFMQI_RFH2_ENCODING	Contains decimal data or MQENC_NATIVE
IMFMQI_RFH2_CODEDCHARSETID	Contains decimal data.
IMFMQI_RFH2_FORMAT	<p>Contains one of the following format constants:</p> <p>MQFMT_NONE  MQFMT_ADMIN  MQFMT_CHANNEL_COMPLETED  MQFMT_CICS  MQFMT_COMMAND_1  MQFMT_COMMAND_2  MQFMT_DEAD_LETTER_HEADER  MQFMT_EVENT  MQFMT_IMS  MQFMT_IMS_VAR_STRING  MQFMT_MD_EXTENSION  MQFMT_PCF  MQFMT_REF_MSG_HEADER  MQFMT_STRING  MQFMT_TRIGGER  MQFMT_WORK_INFO_HEADER  MQFMT_XMIT_Q_HEADER  MQFMT_RF_HEADER  MQFMT_RF_HEADER_2</p>
IMFMQI_RFH2_FLAGS	<p>Contains the following character string delimited by blanks:  MQRFH_NONE.</p> <p><i>See note at the beginning of this table.</i></p>
IMFMQI_RFH2_NAMEVALUECCSID	Contains decimal data.
IMFMQI_RFH2_NAMEVALUEDATA	Contains the exact string of data that is contained in the message buffer in the position following the NameValueCCSID field.

## PCF Variables – EXECs

This table lists the PCF variables – EXECs.

Table 48. PCF Variables – (Rules and Rule-Invoked EXECs)

Variable Name	Notes
<b>Note:</b>	This list may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i> .
IMFMQI_CFH_TYPE	Contains one or more of the following character strings delimited by blanks:  MQCFT_EVENT MQCFT_COMMAND  <i>See note at the beginning of this table.</i>
IMFMQI_CFH_STRUCLength	Contains decimal data.
IMFMQI_CFH_VERSION	Contains MQCFH_VERSION_1.
IMFMQI_CFH_COMMAND	Contains one of the following constants:  MQCMD_Q_MGR_EVENT MQCMD_PERFM_EVENT MQCMD_CHANNEL_EVENT  <i>See note at the beginning of this table.</i>
IMFMQI_CFH_MSGSEQNUMBER	Contains decimal data.
IMFMQI_CFH_CONTROL	Contains the constant MQCFC_LAST.
IMFMQI_CFH_COMPCODE	Contains one of the following constants:  MQCC_OK MQCC_WARNING  <i>See note at the beginning of this table.</i>
IMFMQI_CFH_REASON	Contains a numeric reason code between 0 and 999999999.
IMFMQI_CFH_PARAMETERCOUNT	Contains decimal data.

---

## Appendix E. Conversion Checklist

---

### Converting to the Current Release of MAINVIEW AutoOPERATOR

In MAINVIEW AutoOPERATOR release 6.2, your 5.1 automation works exactly as it did in release 5.1. However, by converting your 5.1 automation, you can use the new features that are available in release 6.2. This checklist describes how you can change your 5.1 automation to use 6.2 functions.

**Note:** These changes are not required when migrating from AutoOPERATOR 6.1 to AutoOPERATOR 6.2.

### Rule Processing

In AutoOPERATOR 5.1, two types of MQSeries messages are recognized by the rule processor. They are MQEVENT and USER messages, specified under the Type field on the Selection Criteria panel. In MAINVIEW AutoOPERATOR 6.2, the Format field replaces the Type field used in release 5.1. Any known IBM format name constant can be specified. The following list describes valid entries for the Format field:

MQFMT_EVENT	Event message
MQFMT_NONE	No format name
MQFMT_ADMIN	Command server request/reply message
MQFMT_CHANNEL_COMPLETED	Channel completed message
MQFMT_CICS	CICS information header
MQFMT_COMMAND_1	Type 1 command reply message
MQFMT_COMMAND_2	Type 2 command reply message
MQFMT_DEAD_LETTER_HEADER	Dead-letter header
MQFMT_IMS	IMS information header
MQFMT_IMS_VAR_STRING	IMS variable string
MQFMT_MD_EXTENSION	Message-descriptor extension
MQFMT_PCF	User-defined message in programmable command format
MQFMT_REF_MSG_HEADER	Reference message header
MQFMT_STRING	Message consisting entirely of characters
MQFMT_TRIGGER	Trigger message
MQFMT_WORK_INFO_HEADER	Work information header
MQFMT_XMIT_Q_HEADER	Transmission queue header

USER	Any message format except MQFMT_EVENT
RFH	Rules and Formatting Header
RFH2	Rules and Formatting Header – version 2

**Note:** To preserve compatibility, Rules created in 5.1 using MQEVENT and USER will continue to work and will not be changed by the MAINVIEW AutoOPERATOR 6.2 Rule Processor.

- MQSeries variables (IMFQxxx) used in Rules should be changed to use the corresponding new-style variable for MAINVIEW AutoOPERATOR 6.2, for example: IMFQ\_MD\_\*. See Chapter 4, “Automating MQSeries Events”.
- New operators are available to use when comparing MQSeries data that is hexadecimal or binary. These operators are available on the MsgId, CorrelId and Msg Buffer fields of the Selection Criteria and the Variable Dependency panels. See Chapter 4, “Automating MQSeries Events”.
- The Selection Criteria panel has new Substring and length fields adjacent to the MsgId, CorrelId and Msg Buffer fields that you can use to target comparisons at specific locations within the respective field or buffer.
- The Action Specification panel has two new fields:
  - **Destination QMgr** -- you can use this field to specify that a message being copied or moved should go to another queue manager.
  - **Remove DLH** -- you can use this field to remove a message dead letter header and the original message that was rebuilt during a copy or move operation.

**Note:** A message passed to a Rule-invoked EXEC will not contain the dead letter header when using this option.
- The AAOMQLxx BBPARM member can be updated to specify that a queue should be opened as SHARE or EXCLUSIVE. See Chapter 2, “Implementing MAINVIEW AutoOPERATOR for MQSeries” on page 9 for details.
- The AAOMQLxx BBPARM member can be updated to specify that the Rule processor should see messages present on the queue when AutoOPERATOR opens the queue for possible automation. See Chapter 2, “Implementing MAINVIEW AutoOPERATOR for MQSeries” on page 9 for details.

## EXEC Manager

The following information addresses the comparison between AutoOPERATOR 5.1 and MAINVIEW AutoOPERATOR 6.2 EXEC managers:

- For every IMFEXEC MQ statement in AutoOPERATOR 5.1, there is a corresponding IMFEXEC MQI statement in MAINVIEW AutoOPERATOR 6.2. To use the new statements, you must use the new variables. In addition, MAINVIEW AutoOPERATOR 6.2 uses the IBM constant names as variable values. See “Rule Processing” on page 297 for details.
- Variables used for IMFEXEC MQI statements have a different format than those used with the IMFEXEC MQ statement from AutoOPERATOR 5.1. The variables are made up of a common prefix (IMFMQI\_), followed by the MQSeries structure ID (MD\_, DLH\_,

etc.), followed by the field name from the MQSeries structure (Format, Expire, etc.). You must use these variables when converting to IMFEXEC MQI. See Chapter 2, “Implementing MAINVIEW AutoOPERATOR for MQSeries” on page 9 for details.

- Options variables in MAINVIEW AutoOPERATOR 6.2 now use complete constant names instead of partial constant names as in AutoOPERATOR 5.1. For example, options for an IMFEXEC MQI OPEN statement may be set as follows: `IMFMQI_OO_OPTIONS = 'MQOO_OUTPUT'`. See Chapter 4, “Automating MQSeries Events”.
- The Options parameter that was available on AutoOPERATOR 5.1 IMFEXEC MQ statements (OOPTS, COPTS, GOPTS and POPTS) is still available. However, the values specified in MAINVIEW AutoOPERATOR 6.2 consist of the full IBM constant name. For example, `MQGMO_NO_SYNCPOINT` instead of `NO_SYNCPOINT`, as specified in AutoOPERATOR 5.1. See Chapter 4, “Automating MQSeries Events”.
- The IMFEXEC COPY MQ statement in AutoOPERATOR 5.1 has been replaced in MAINVIEW AutoOPERATOR 6.2 with the IMFEXEC COPY MQI statement. Using this statement, you can specify which structures in the message should have the COPY of their variables performed. See Chapter 4, “Automating MQSeries Events”.
- The IMFEXEC DISPLAY MQ statement in AutoOPERATOR 5.1 has been replaced in MAINVIEW AutoOPERATOR 6.2 with the IMFEXEC DISPLAY MQI statement. Using this statement, you can specify which structure variables should be written to the BBI journal. See Chapter 4, “Automating MQSeries Events”.



---

# Glossary

This glossary defines BMC Software terminology. Other dictionaries and glossaries can be used in conjunction with this glossary.

Since this glossary pertains to BMC Software-related products, some of the terms defined might not appear in this book.

To help you find the information you need, this glossary uses the following cross-references:

<b>Contrast with</b>	Indicates a term that has a contrary or contradictory meaning.
<b>See</b>	Indicates an entry that is a synonym or contains expanded information.
<b>See also</b>	Indicates an entry that contains related information.

---

## A

**action.** Defined operation, such as modifying a MAINVIEW window, that is performed in response to a command. *See* object.

**active window.** Any MAINVIEW window in which data can be refreshed. *See* alternate window, current window, window.

**administrative view.** Display from which a product's management tasks are performed, such as the DSLIST view for managing historical data sets. *See* view.

**ALT WIN field.** Input field that allows you to specify the window identifier for an alternate window where the results of a hyperlink are displayed. *See* alternate window.

**Alternate Access.** *See* MAINVIEW Alternate Access.

**alternate form.** View requested through the FORM command that changes the format of a previously displayed view to show related information. *See also* form, query.

**alternate window.** (1) Window that is specifically selected to display the results of a hyperlink. (2) Window whose identifier is defined to the ALT WIN field. *Contrast with* current window. *See* active window, window, ALT WIN field.

**analyzer.** (1) Online display that presents a snapshot of status and activity data and indicates problem areas. (2) Component of CMF MONITOR. *See* CMF MONITOR Analyzer.

**application.** (1) Program that performs a specific set of tasks within a MAINVIEW product. (2) In MAINVIEW VistaPoint, combination of workloads to enable display of their transaction performance data in a single view.

**application trace.** *See* trace.

**ASCH workload.** Workload comprising Advanced Program-to-Program Communication (APPC) address spaces.

**AutoCustomization.** Online facility for customizing the installation of products. AutoCustomization provides an ISPF panel interface that both presents customization steps in sequence and provides current status information about the progress of the installation.

**automatic screen update.** Usage mode wherein the currently displayed screen is refreshed automatically with new data at an interval you specify. Invoked by the ASU command.

## B

**batch workload.** Workload consisting of address spaces running batch jobs.

**BBI.** Basic architecture that distributes work between workstations and multiple OS/390 targets for BMC Software MAINVIEW products.

**BBI-SS PAS.** *See* BBI subsystem product address space.

**BBI subsystem product address space (BBI-SS PAS).** OS/390 subsystem address space that manages communication between local and remote systems and that contains one or more of the following products:

- Command MQ for S/390
- MAINVIEW AutoOPERATOR
- MAINVIEW for CICS
- MAINVIEW for DB2
- MAINVIEW for DBCTL
- MAINVIEW for IMS Online
- MAINVIEW for MQSeries
- MAINVIEW SRM
- MAINVIEW VistaPoint (for CICS, DB2, DBCTL, and IMS workloads)

**BBPARM.** *See* parameter library.

**BBPROC.** *See* procedure library.

**BBPROF.** See profile library.

**BBSAMP.** See sample library.

**BBV.** See MAINVIEW Alternate Access.

**BBXS.** BMC Software Subsystem Services. Common set of service routines loaded into common storage and used by several BMC Software MAINVIEW products.

**border.** Visual indication of the boundaries of a window.

**bottleneck analysis.** Process of determining which resources have insufficient capacity to provide acceptable service levels and that therefore can cause performance problems.

## C

**CA-Disk.** Data management system by Computer Associates that replaced the DMS product.

**CAS.** Coordinating address space. One of the address spaces used by the MAINVIEW windows environment architecture. The CAS supplies common services and enables communication between linked systems. Each OS/390 or z/OS image requires a separate CAS. Cross-system communication is established through the CAS using VTAM and XCF communication links.

**CFMON.** See coupling facility monitoring.

**chart.** Display format for graphical data. See also graph.

**CICSplex.** User-defined set of one or more CICS systems that are controlled and managed as a single functional entity.

**CMF MONITOR.** Comprehensive Management Facility MONITOR. Product that measures and reports on all critical system resources, such as CPU, channel, and device usage; memory, paging, and swapping activity; and workload performance.

**CMF MONITOR Analyzer.** Batch component of CMF MONITOR that reads the SMF user and 70 series records created by the CMF MONITOR Extractor and/or the RMF Extractor and formats them into printed system performance reports.

**CMF MONITOR Extractor.** Component of CMF that collects performance statistics for CMF MONITOR Analyzer, CMF MONITOR Online, MAINVIEW for OS/390, and RMF postprocessor. See CMF MONITOR Analyzer, CMF MONITOR Online, MAINVIEW for OS/390.

**CMF MONITOR Online.** Component of CMF that uses the MAINVIEW window interface to present data on all address spaces, their use of various system resources, and the delays that each address space incurs while waiting for access to these resources. See CMF MONITOR, MAINVIEW for OS/390.

**CMF Type 79 API.** Application programming interface, provided by CMF, that provides access to MAINVIEW SMF-type 79 records.

**CMFMON.** Component of CMF MONITOR that simplifies online retrieval of information about system hardware and application performance and creates MAINVIEW SMF-type 79 records.

The CMFMON *online facility* can be used to view data in one or more formatted screens.

The CMFMON *write facility* can be used to write collected data as MAINVIEW SMF-type 79 records to an SMF or sequential data set.

**CMRDETL.** MAINVIEW for CICS data set that stores detail transaction records (type 6E) and abend records (type 6D). Detail records are logged for each successful transaction. Abend records are written when an abend occurs. Both records have the same format when stored on CMRDETL.

**CMRSTATS.** MAINVIEW for CICS data set that stores both CICS operational statistic records, at five-minute intervals, and other records, at intervals defined by parameters specified during customization (using CMRSOPT).

**column.** Vertical component of a view or display, typically containing fields of the same type of information, that varies by the objects associated in each row.

**collection interval.** Length of time data is collected. See also delta mode, total mode.

**command delimiter.** Special character, usually a ; (semicolon), used to stack commands typed concurrently on the COMMAND line for sequential execution.

**COMMAND line.** Line in the control area of the display screen where primary commands can be typed. Contrast with line command column.

**Command MQ Automation D/S.** Command MQ agents, which provide local proactive monitoring for both MQSeries and MSMQ (Microsoft message queue manager). The Command MQ agents operate at the local node level where they continue to perform functions regardless of the availability of the MQM (message queue manager) network. Functionality includes automatic monitoring and restarts of channels, queue managers, queues and command servers. In cases where automated recovery is not possible, the agents transport critical alert information to a central console.

**Command MQ Automation S/390.** Command MQ component, which monitors the MQM (message queue manager) networks and intercedes to perform corrective actions when problems arise. Solutions include:

- Dead-Letter Queue management
- System Queue Archival
- Service Interval Performance solutions
- Channel Availability

These solutions help ensure immediate relief to some of the most pressing MQM operations and performance problems.

**Command MQ for D/S.** Command MQ for D/S utilizes a true client/server architecture and employs resident agents to provide configuration, administration, performance monitoring and operations management for the MQM (message queue manager) network.

**Command MQ for S/390.** See MAINVIEW for MQSeries.

**COMMON STORAGE MONITOR.** Component of MAINVIEW for OS/390 that monitors usage and reconfigures OS/390 or z/OS common storage blocks.

**composite workload.** Workload made up of a WLM workload or other workloads, which are called *constituent workloads*.

**constituent workload.** Member of a composite workload. Constituent workloads in a composite usually belong to a single workload class, but sometimes are mixed.

**contention.** Occurs when there are more requests for service than there are servers available.

**context.** In a Plex Manager view, field that contains the name of a target or group of targets specified with the CONTEXT command. See scope, service point, SSI context, target context.

**CONTEXT command.** Specifies either a MAINVIEW product and a specific target for that product (see target context) or a MAINVIEW product and a name representing one or more targets (see *SSI context*) for that product.

**control statement.** (1) Statement that interrupts a sequence of instructions and transfers control to another part of the program. (2) Statement that names samplers and other parameters that configure the MAINVIEW components to perform specified functions. (3) In CMF MONITOR, statement in a parameter library member used to identify a sampler in the extractor or a report in the analyzer, or to describe either component's processing requirements to the operating system.

**coupling facility monitoring (CFMON).** Coupling facility views that monitor the activity of your system's coupling facilities.

**current data.** Data that reflects the system in its current state. The two types of current data are realtime data and interval data. Contrast with historical data. See also interval data and realtime data.

**current window.** In the MAINVIEW window environment, window where the main dialog with the application takes place. The current window is used as the default window destination for commands issued on the COMMAND line when no window number is specified. Contrast with alternate window. See active window, window.

## D

**DASD.** Direct Access Storage Device. (1) A device with rotating recording surfaces that provides immediate access to stored data. (2) Any device that responds to a DASD program.

**data collector.** Program that belongs to a MAINVIEW product and that collects data from various sources and stores the data in records used by views. For example, MAINVIEW for OS/390 data collectors obtain data from OS/390 or z/OS services, OS/390 or z/OS control blocks, CMF MONITOR

Extractor control blocks, and other sources. Contrast with extractor.

**delta mode.** (1) In MAINVIEW for DB2 analyzer displays, difference between the value sampled at the start of the current statistics interval and the value sampled by the current analyzer request. See also *statistics interval*. (2) In CMFMON, usage mode wherein certain columns of data reflect the difference in values between one sample cycle and the next. Invoked by the DELTA ON command. See also collection interval, sample cycle, total mode.

**DFSMS.** Data Facility Storage Management System. Data management, backup, and HSM software from IBM for OS/390 or z/OS mainframes.

**DMR.** See MAINVIEW for DB2.

**DMS.** Data Management System. See CA-Disk.

**DMS2HSM.** See MAINVIEW SRM DMS2HSM.

**DSO.** Data Set Optimizer. CMF MONITOR Extractor component that uses CMF MONITOR Extractor data to produce reports specifying the optimal ordering of data sets on moveable head devices.

## E

**EasyHSM.** See MAINVIEW SRM EasyHSM.

**EasyPOOL.** See MAINVIEW SRM EasyPOOL.

**EasySMS.** See MAINVIEW SRM EasySMS.

**element.** (1) Data component of a data collector record, shown in a view as a field. (2) Internal value of a field in a view, used in product functions.

**element help.** Online help for a field in a view. The preferred term is *field help*.

**Enterprise Storage Automation.** See MAINVIEW SRM Enterprise Storage Automation.

**event.** A message issued by Enterprise Storage Automation. User-defined storage occurrences generate events in the form of messages. These events provide an early warning system for storage problems and are routed to user-specified destinations for central viewing and management.

**Event Collector.** Component for MAINVIEW for IMS Online, MAINVIEW for IMS Offline, and MAINVIEW for DBCTL that collects data about events in the IMS environment. This data is required for Workload Monitor and optional for Workload Analyzer (except for the workload trace service). This data also is recorded as transaction records (X'FA') and program records (X'F9') on the IMS system log for later use by the MAINVIEW for IMS Offline components: Performance Reporter and Transaction Accountant.

**expand.** Predefined link from one display to a related display. See also hyperlink.

**extractor.** Program that collects data from various sources and keeps the data control blocks to be written as records. Extractors obtain data from services, control blocks, and other sources. *Contrast with* data collector.

**extractor interval.** *See* collection interval.

## F

**fast path.** Predefined link between one screen and another. To use the fast path, place the cursor on a single value in a field and press Enter. The resulting screen displays more detailed information about the selected value. *See also* hyperlink.

**field.** Group of character positions within a screen or report used to type or display specific information.

**field help.** Online help describing the purpose or contents of a field on a screen. To display field help, place the cursor anywhere in a field and press PF1 (HELP). In some products, field help is accessible from the screen help that is displayed when you press PF1.

**filter.** Selection criteria used to limit the number of rows displayed in a view. Data that does not meet the selection criteria is not displayed. A filter is composed of an element, an operator, and an operand (a number or character string). Filters can be implemented in view customization, through the PARM/QPARM commands, or through the Where/QWhere commands. Filters are established against elements of data.

**fire.** The term used to indicate that an event has triggered an action. In MAINVIEW AutoOPERATOR, when a rule selection criteria matches an incoming event and *fires*, the user-specified automation actions are performed. This process is also called *handling* the event.

**fixed field.** Field that remains stationary at the left margin of a screen that is scrolled either right or left.

**FOCAL POINT.** MAINVIEW product that displays a summary of key performance indicators across systems, sites, and applications from a single terminal.

**form.** One of two constituent parts of a view; the other is query. A form defines how the data is presented; a query identifies the data required for the view. *See also* *query*, *view*.

**full-screen mode.** Display of a MAINVIEW product application or service on the entire screen. There is no window information line. *Contrast with* windows mode.

## G

**global command.** Any MAINVIEW window interface command that can affect all windows in the window area of a MAINVIEW display.

**graph.** Graphical display of data that you select from a MAINVIEW window environment view. *See also* chart.

## H

**hilevel.** For MAINVIEW products, high-level data set qualifier required by a site's naming conventions.

**historical data.** (1) Data that reflects the system as it existed at the end of a past recording interval or the duration of several intervals. (2) Any data stored in the historical database and retrieved using the TIME command. *Contrast with* current data, interval data and realtime data.

**historical database.** Collection of performance data written at the end of each installation-defined recording interval and containing up to 100 VSAM clusters. Data is extracted from the historical database with the TIME command. *See* historical data.

**historical data set.** In MAINVIEW products that display historical data, VSAM cluster file in which data is recorded at regular intervals.

**HSM.** (Hierarchical Storage Management) Automatic movement of files from hard disk to slower, less-expensive storage media. The typical hierarchy is from magnetic disk to optical disk to tape.

**hyperlink.** (1) Preset field in a view or an EXPAND line on a display that permits you to

- Access cursor-sensitive help
- Issue commands
- Link to another view or display

The transfer can be either within a single product or to a related display/view in a different BMC Software product. Generally, hyperlinked fields are highlighted. (2) Cursor-activated short path from a topic or term in online help to related information. *See also* fast path.

## I

**Image log.** Collection of screen-display records. Image logs can be created for both the BBI-SS PAS and the BBI terminal session (TS).

The BBI-SS PAS Image log consists of two data sets that are used alternately: as one fills up, the other is used. Logging to the BBI-SS PAS Image log stops when both data sets are filled and the first data set is not processed by the archive program.

The TS Image log is a single data set that wraps around when full.

**IMSplex System Manager (IPSM).** MVIMS Online and MVDBC service that provides Single System Image views of resources and bottlenecks for applications across one or more IMS regions and systems.

**interval data.** Cumulative data collected during a collection interval. Intervals usually last from 15 to 30 minutes depending on how the recording interval is specified during product customization. *Contrast with* historical data.

**Note:** If change is made to the workloads, a new interval will be started.

*See also* current data and realtime data.

**InTune.** Product for improving application program performance. It monitors the program and provides information used to reduce bottlenecks and delays.

**IRUF.** IMS Resource Utilization File (IRUF). IRUFs can be either detail (one event, one record) or summarized (more than one event, one record). A detail IRUF is created by processing the IMS system log through a program called IMFLEDIT. A summarized IRUF is created by processing one or more detail IRUFs, one or more summarized IRUFs, or a combination of both, through a sort program and the TASCOSTR program.

## J

**job activity view.** Report about address space consumption of resources. *See* view.

**journal.** Special-purpose data set that stores the chronological records of operator and system actions.

**Journal log.** Collection of messages. Journal logs are created for both the BBI-SS PAS and the BBI terminal session (TS).

The BBI-SS PAS Journal log consists of two data sets that are used alternately: as one fills up, the other is used. Logging to the BBI-SS PAS Journal log stops when both data sets are filled and the first data set is not being processed by the archive program.

The TS Journal log is a single data set that wraps around when full.

## L

**line command.** Command that you type in the line command column in a view or display. Line commands initiate actions that apply to the data displayed in that particular row.

**line command column.** Command input column on the left side of a view or display. *Contrast with* COMMAND line.

**Log Edit.** In the MAINVIEW for IMS Offline program named IMFLEDIT, function that extracts transaction (X'FA') and program (X'F9') records from the IMS system log. IMFLEDIT also extracts certain records that were recorded on the system log by IMS. IMFLEDIT then formats the records into a file called the IMS Resource Utilization File (IRUF).

## M

**MAINVIEW.** BMC Software integrated systems management architecture.

**MAINVIEW Alarm Manager.** In conjunction with other MAINVIEW products, notifies you when an exception condition occurs. MAINVIEW Alarm Manager is capable of monitoring multiple systems simultaneously, which means that MAINVIEW Alarm Manager installed on one system keeps track of your entire sysplex. You can then display a single view that show exceptions for all MAINVIEW performance monitors within your OS/390 or z/OS enterprise.

**MAINVIEW Alternate Access.** Enables MAINVIEW products to be used without TSO by providing access through EXCP and VTAM interfaces.

**MAINVIEW Application Program Interface.** REXX- or CLIST-based, callable interface that allows MAINVIEW AutoOPERATOR EXECs to access MAINVIEW monitor product view data.

**MAINVIEW AutoOPERATOR.** Product that uses tools, techniques, and facilities to automate routine operator tasks and provide online performance monitoring, and that achieves high availability through error minimization, improved productivity, and problem prediction and prevention.

**MAINVIEW control area.** In the MAINVIEW window environment, first three lines at the top of the view containing the window information line and the COMMAND, SCROLL, CURR WIN, and ALT WIN lines. The control area cannot be customized and is part of the information display. *Contrast with* MAINVIEW display area, MAINVIEW window area.

**MAINVIEW display area.** *See* MAINVIEW window area.

**MAINVIEW Explorer.** Product that provides access to MAINVIEW products from a Web browser running under Windows. MAINVIEW Explorer replaces MAINVIEW Desktop.

**MAINVIEW for CICS.** Product (formerly MV MANAGER for CICS) that provides realtime application performance analysis and monitoring for CICS system management.

**MAINVIEW for DB2.** Product (formerly MV MANAGER for DB2) that provides realtime and historical application performance analysis and monitoring for DB2 subsystem management.

**MAINVIEW for DBCTL.** Product (formerly MV MANAGER for DBCTL) that provides realtime application performance analysis and monitoring for DBCTL management.

**MAINVIEW for IMS (MVIMS) Offline.** Product with a Performance Reporter component that organizes data and prints reports used to analyze IMS performance and a Transaction Accountant component that produces cost accounting and user charge-back records and reports.

**MAINVIEW for IMS (MVIMS) Online.** Product that provides realtime application performance analysis and monitoring for IMS management.

**MAINVIEW for IP.** Product that monitors OS/390 and z/OS mission-critical application performance as it relates to TCP/IP stack usage. Collected data includes availability, connections, response times, routers, service levels, storage, traffic, Web cache, and so on.

**MAINVIEW for Linux-Servers.** Product that allows you to monitor the performance of your Linux systems from the MAINVIEW windows interface.

**MAINVIEW for MQSeries.** Delivers comprehensive capabilities for configuration, administration, performance monitoring and operations management for an entire MQM (message queue manager) network.

**MAINVIEW for OS/390.** System management application (known as MAINVIEW for MVS prior to version 2.5). Built upon the MAINVIEW window environment architecture, it uses the window interface to provide access to system performance data and other functions necessary in the overall management of an enterprise.

**MAINVIEW for UNIX System Services.** System management application that allows you to monitor the performance of the Unix System Services from a MAINVIEW window interface.

**MAINVIEW for VTAM.** Product that displays application performance data by application, transaction ID, and LU name. This collected data includes: connections, response time statistics, application availability, and application throughput.

**MAINVIEW for WebSphere Application Server (formerly known as MAINVIEW for WebSphere).** Product that provides extensive monitoring for the IBM WebSphere Application Server for z/OS and OS/390 environment.

**MAINVIEW Selection Menu.** ISPF selection panel that provides access to all MAINVIEW windows-mode and full-screen mode products.

**MAINVIEW SRM.** *See* MAINVIEW Storage Resource Manager (SRM).

**MAINVIEW SRM DMS2HSM.** Product that facilitates the conversion of CA-Disk, formerly known as DMS, to HSM.

**MAINVIEW SRM EasyHSM.** Product that provides online monitoring and reporting to help storage managers use DFHSM efficiently.

**MAINVIEW SRM EasyPOOL.** Product that provides control over data set allocation and enforcement of allocation and naming standards. EasyPOOL functions operate at the operating system level to intercept normal job processing, thus providing services without any JCL changes.

**MAINVIEW SRM EasySMS.** Product that provides tools that aid in the conversion to DFSMS and provides enhancement to the DFSMS environment after implementation. EasySMS consists of the EasyACS functions, the SMSACSTE function, and the Monitoring and Positioning Facility.

**MAINVIEW SRM Enterprise Storage Automation.** Product that delivers powerful event generation and storage automation technology across the storage enterprise. Used in conjunction with MAINVIEW AutoOPERATOR, automated solutions to perform pool, volume, application, or data set-level manipulation can be created and used in response to any condition or invoked to perform ad hoc requests

**MAINVIEW SRM SG-Auto.** Product that provides early warning notification of storage anomalies and automated responses to those anomalies based on conditions in the storage subsystem.

**MAINVIEW SRM SG-Control.** Product that provides real-time monitoring, budgeting, and control of DASD space utilization.

**MAINVIEW SRM StopX37/II.** Product that provides enhancements to OS/390 or z/OS space management, reducing the incidence of space-related processing problems. The StopX37/II functions operate at the system level to interceptabend conditions or standards violations, thus providing services without any JCL changes.

**MAINVIEW SRM StorageGUARD.** Product that monitors and reports on DASD consumption and provides historical views to help control current and future DASD usage.

**MAINVIEW Storage Resource Manager (SRM).** Suite of products that assists in all phases of OS/390 or z/OS storage management. MAINVIEW SRM consists of products that perform automation, reporting, trend analysis, and error correction for storage management.

**MAINVIEW SYSPROG Services.** *See* SYSPROG Services.

**MAINVIEW VistaPoint.** Product that provides enterprise-wide views of performance. Application and workload views are available for CICS, DB2, DBCTL, IMS, and OS/390. Data is summarized at the level of detail needed; for example, views can be for a single target, an OS/390 or z/OS image, or an entire enterprise.

**MAINVIEW window area.** Portion of the information display that is not the control area and in which views are displayed and windows opened. It includes all but the first three lines of the information display. *Contrast with* MAINVIEW control area.

**monitor.** Online service that measures resources or workloads at user-defined intervals and issues warnings when user-defined thresholds are exceeded.

**Multi-Level Automation (MLA).** The user-defined, multiple step process in Enterprise Storage Automation that implements solutions in a tiered approach, where solutions are invoked one after another until the condition is resolved.

**MVALARM.** *See* MAINVIEW Alarm Manager.

**MVAPI.** *See* MAINVIEW Application Program Interface.

**MVCICS.** *See* MAINVIEW for CICS.

**MVDB2.** *See* MAINVIEW for DB2.

**MVDBC.** *See* MAINVIEW for DBCTL.

**MVIMS.** *See* MAINVIEW for IMS.

**MVIP.** *See* MAINVIEW for IP.

**MVLNX.** *See* MAINVIEW for Linux-Servers.

**MVMQ.** *See* MAINVIEW for MQSeries.

**MVMVS.** *See* MAINVIEW for OS/390.

**MVScope.** MAINVIEW for OS/390 application that traces both CPU usage down to the CSECT level and I/O usage down to the channel program level.

**MVSRM.** *See* MAINVIEW Storage Resource Manager (SRM).

**MVSRMHSM.** *See* MAINVIEW SRM EasyHSM.  
**MVSRMSGC.** *See* MAINVIEW SRM SG-Control.  
**MVSRMSGD.** *See* MAINVIEW SRM StorageGUARD.  
**MVSRMSGP.** *See* MAINVIEW SRM StorageGUARD.  
**MVVP.** *See* MAINVIEW VistaPoint.  
**MVVTAM.** *See* MAINVIEW for VTAM.  
**MVWEB.** *See* MAINVIEW for WebSphere Application Server.

## N

**nested help.** Multiple layers of help pop-up windows. Each successive layer is accessed by clicking a hyperlink from the previous layer.

## O

**object.** Anything you can manipulate as a single unit. MAINVIEW objects can be any of the following: product, secondary window, view, row, column, or field.

You can issue an action against an object by issuing a line command in the line command column to the left of the object. *See* action.

**OMVS workload.** Workload consisting of OS/390 OpenEdition address spaces.

**online help.** Help information that is accessible online.

**OS/390 and z/OS Installer.** BMC Software common installation system for mainframe products.

**OS/390 product address space (PAS).** Address space containing OS/390 or z/OS data collectors, including the CMF MONITOR Extractor. Used by the MAINVIEW for OS/390, MAINVIEW for Unix System Services, and CMF MONITOR products. *See* PAS.

## P

**parameter library.** Data set consisting of members that contain parameters for specific MAINVIEW products or a support component. There can be several versions:

- The distributed parameter library, called BBPARAM
- A site-specific parameter library or libraries

These can be

- A library created by AutoCustomization, called UBBPARAM
- A library created manually, with a unique name

**PAS.** Product address space. Used by the MAINVIEW products. Contains data collectors and other product functions.

*See* OS/390 product address space (PAS), BBI subsystem product address space (BBI-SS PAS).

**performance group workload.** Collection of address spaced defined to OS/390 or z/OS. If you are running OS/390 or z/OS with WLM in compatibility mode, MAINVIEW for OS/390 creates a performance group workload instead of a service class. *See* service class workload, workload definition.

**PERFORMANCE MANAGER.** MAINVIEW for CICS online service for monitoring and managing current performance of CICS regions.

**Performance Reporter (MVIMS Offline).** MVIMS Offline component that organizes data and prints reports that can be used to analyze IMS performance.

**Performance Reporter.** Product component that generates offline batch reports. The following products can generate these reports:

- MAINVIEW for DB2
- MAINVIEW for CICS

**Plex Manager.** Product through which cross-system communication, MAINVIEW security, and an SSI context are established and controlled. Plex Manager is shipped with MAINVIEW window environment products as part of the coordinating address space (CAS) and is accessible as a menu option from the MAINVIEW Selection Menu.

**PRGP workload.** In MVS/SP 5.0 or earlier, or in compatibility mode in MVS/SP 5.1 or later, composite of service classes. MAINVIEW for OS/390 creates a performance group workload for each performance group defined in the current IEAIPStt member.

**procedure library.** Data set consisting of members that contain executable procedures used by MAINVIEW AutoOPERATOR. These procedures are execute command lists (EXECs) that automate site functions. There can be several versions:

- The distributed parameter library, called BBPROC
- A site-specific parameter library or libraries

These can be

- A library created by AutoCustomization, called UBBPROC
- A library created manually, with a unique name

The site-created EXECs can be either user-written or customized MAINVIEW AutoOPERATOR-supplied EXECs from BBPROC.

**product address space.** *See* PAS.

**profile library.** Data set consisting of members that contain profile information and cycle refresh definitions for a terminal session connected to a BBI-SS PAS. Other members are dynamically created by MAINVIEW applications. There can be several versions:

- The distributed profile library, called BBPROF
- A site-specific profile library or libraries

These can be

- A library created by AutoCustomization, called SBBPROF
- A library created manually, with a unique name

The site library is a common profile shared by all site users. The terminal session CLIST creates a user profile automatically if one does not exist; it is called userid.BBPROF, where userid is your logon ID. User profile libraries allow each user to specify unique PF keys, CYCLE commands, target system defaults, a Primary Option Menu, and a unique set of application profiles.

## Q

**query.** One of two constituent parts of a view; the other is form. A query defines the data for a view; a form defines the display format. *See also* form, view.

## R

**realtime data.** Performance data as it exists at the moment of inquiry. Realtime data is recorded during the smallest unit of time for data collection. *Contrast with* historical data. *See also* current data and interval data.

**Resource Analyzer.** Online realtime displays used to analyze IMS resources and determine which are affected by specific workload problems.

**Resource Monitor.** Online data collection services used to monitor IMS resources and issue warnings when defined utilization thresholds are exceeded.

**row.** (1) Horizontal component of a view or display comprising all the fields pertaining to a single device, address space, user, etc. (2) Horizontal component of a DB2 table consisting of a sequence of values, one for each column of the table.

**RxD2.** Product that provides access to DB2 from REXX. It provides tools to query the DB2 catalog, issue dynamic SQL, test DB2 applications, analyze EXPLAIN data, generate DDL or DB2 utility JCL, edit DB2 table spaces, perform security administration, and much more.

## S

**sample cycle.** Time between data samples.

For the CMF MONITOR Extractor, this is the time specified in the extractor control statements (usually 1 to 5 seconds).

For realtime data, the cycle is not fixed. Data is sampled each time you press Enter.

**sample library.** Data set consisting of members each of which contains one of the following:

- Sample JCL that can be edited to perform specific functions
- A macro that is referenced in the assembly of user-written services
- A sample user exit routine

There can be several versions:

- The distributed sample library, called BBSAMP
- A site-specific sample library or libraries

These can be

- A library created by AutoCustomization, called UBBSAMP
- A library created manually, with a unique name

**sampler.** Program that monitors a specific aspect of system performance. Includes utilization thresholds used by the Exception Monitor. The CMF MONITOR Extractor contains samplers.

**SBBPROF.** *See* profile library.

**scope.** Subset of an SSI context. The scope could be all the data for the context or a subset of data within the context. It is user- or site-defined. *See* SSI context, target.

**screen definition.** Configuration of one or more views that have been stored with the SAVEScr command and assigned a unique name. A screen includes the layout of the windows and the view, context, system, and product active in each window.

**selection view.** In MAINVIEW products, view displaying a list of available views.

**service class workload.** Collection of address spaces defined to OS/390 or z/OS. If you are running Workload Manager (WLM) in goal mode, MAINVIEW for OS/390 creates a service class workload for each service class that you define through WLM definition dialogs.

If you are running MVS 4.3 or earlier, or MVS/SP 5.1 or later with WLM in compatibility mode, MVS creates a performance group workload instead of a service class. *See* performance group workload.

**service objective.** Workload performance goal, specified in terms of response time for TSO workloads or turnaround time for batch workloads. Performance group workloads can be measured by either objective. Composite workload service objectives consist of user-defined weighting factors assigned to each constituent workload. For compatibility mode, neither OS/390 nor z/OS provides any way to measure service.

**service point.** Specification, to MAINVIEW, of the services required to enable a specific product. Services can be actions, selectors, or views. Each target (for example, CICS, DB2, or IMS) has its own service point.

The PLEX view lists all the defined service points known to the CAS to which the terminal session is connected.

**service request block (SRB).** Control block that represents a routine to be dispatched. SRB mode routines generally perform work for the operating system at a high priority. An SRB is similar to a task control block (TCB) in that it identifies a unit of work to the system. *See also* task control block.

**service select code.** Code entered to invoke analyzers, monitors, and general services. This code is also the name of the individual service.

**session.** Total period of time an address space has been active. A session begins when monitoring can be performed. If the product address space (PAS) starts after the job, the session starts with the PAS.

**SG-Auto.** *See* MAINVIEW SRM SG-Auto.

**SG-Control.** *See* MAINVIEW SRM SG-Control.

**single system image (SSI).** Feature of the MAINVIEW window environment architecture where you can view and perform actions on multiple OS/390 systems as though they were a single system. The rows of a single tabular view can contain rows from different OS/390 or z/OS images.

**Skeleton Tailoring Facility.** A facility in MAINVIEW AutoOPERATOR that allows skeleton JCL to be used during job submission. Skeleton JCL can contain variables within the JCL statements to be substituted with data values at job submission time. Directive statements can be used in the skeleton JCL to cause the repetition of a set of skeleton statements. This facility functions similar to the TSO skeleton tailoring facility.

**SRB.** *See* service request block.

**SSI.** *See* single system image.

**SSI context.** Name created to represent one or more targets for a given product. *See* context, target.

**started task workload.** Address spaces running jobs that were initiated programmatically.

**statistics interval.** For MAINVIEW for DB2, cumulative count within a predefined interval (30-minute default set by the DB2STATS parameter in the distributed BBPARM member BBIISP00) for an analyzer service DELTA or RATE display. Specifying the DELTA parameter displays the current value as the difference between the value sampled by the current analyzer request and the value sampled at the start of the current interval. Specifying the RATE parameter displays the current value by minute (DELTA divided by the number of elapsed minutes).

**stem variables.** A REXX facility, supported in MAINVIEW AutoOPERATOR REXX EXECs and the Skeleton Tailoring Facility, where variable names end with a period followed by a

number, such as &POOL.1. This configuration allows each variable to actually represent a table or array of data, with the zero variable containing the number of entries in the array. For example, &POOL.0 = 5 would indicate variables &POOL.1 through &POOL.5 exist.

**StopX37/II.** *See* MAINVIEW SRM StopX37/II.

**StorageGUARD.** *See* MAINVIEW SRM StorageGUARD.

**summary view.** View created from a tabular view using the Summarize option in view customization. A summary view compresses several rows of data into a single row based on the summarize criteria.

**SYSPROG services.** Component of MAINVIEW for OS/390. Over 100 services that detect, diagnose, and correct OS/390 or z/OS system problems as they occur. Accessible from the OS/390 Performance and Control Main Menu. Note that this component is also available as a stand-alone product MAINVIEW SYSPROG Services.

**system resource.** *See* object.

## T

**target.** Entity monitored by one or more MAINVIEW products, such as an OS/390 or z/OS image, an IMS or DB2 subsystem, a CICS region, or related workloads across systems. *See* context, scope, SSI context.

**target context.** Single target/product combination. *See* context.

**TASCOSTR.** MAINVIEW for IMS Offline program that summarizes detail and summary IMS Resource Utilization Files (IRUFs) to be used as input to the offline components.

**task control block (TCB).** Address space-specific control block that represents a unit of work that is dispatched in the address space in which it was created. *See also* service request block.

**TCB.** *See* task control block.

**terminal session (TS).** Single point of control for MAINVIEW products, allowing data manipulation and data display and providing other terminal user services for MAINVIEW products. The terminal session runs in a user address space (either a TSO address space or a standalone address space for EXCP/VTAM access).

**TDIR.** *See* trace log directory.

**threshold.** Specified value used to determine whether the data in a field meets specific criteria.

**TLDS.** *See* trace log data set.

**total mode.** Usage mode in CMFMON wherein certain columns of data reflect the cumulative value between collection intervals. Invoked by the DELTA OFF command. *See also* collection interval, delta mode.

**trace.** (1) Record of a series of events chronologically listed as they occur. (2) Online data collection and display services that track transaction activity through DB2, IMS, or CICS.

**trace log data set (TLDS).** Single or multiple external VSAM data sets containing summary or detail trace data for later viewing or printing. The trace log(s) can be defined as needed or dynamically allocated by the BBI-SS PAS. Each trace request is assigned its own trace log data set(s).

**trace log directory (TDIR).** VSAM linear data set containing one entry for each trace log data set. Each entry indicates the date and time of data set creation, the current status of the data set, the trace target, and other related information.

**transaction.** Specific set of input data that initiates a predefined process or job.

**Transaction Accountant.** MVIMS Offline component that produces cost accounting and user charge-back records and reports.

**TS.** *See* terminal session.

**TSO workload.** Workload that consists of address spaces running TSO sessions.

## U

**UAS.** *See* user address space.

**UBBPARM.** *See* parameter library.

**UBBPROC.** *See* procedure library.

**UBBSAMP.** *See* sample library.

**user address space.** Runs a MAINVIEW terminal session (TS) in TSO, VTAM, or EXCP mode.

**User BBPROF.** *See* profile library.

## V

**view.** Formatted data within a MAINVIEW window, acquired from a product as a result of a view command or action. A view consists of two parts: query and form. *See also* form, job activity view, query.

**view definition.** Meaning of data that appears online, including source of data, selection criteria for data field inclusion and placement, data format, summarization, context, product, view name, hyperlink fields, and threshold conditions.

**view command.** Name of a view that you type on the COMMAND line to display that view.

**view command stack.** Internal stack of up to 10 queries. For each command, the stack contains the filter parameters, sort order, context, product, and timeframe that accompany the view.

**view help.** Online help describing the purpose of a view. To display view help, place the cursor on the view name on the window information line and press PF1 (HELP).

## W

**window.** Area of the MAINVIEW screen in which views and resources are presented. A window has visible boundaries and can be smaller than or equal in size to the MAINVIEW window area. *See* active window, alternate window, current window, MAINVIEW window area.

**window information line.** Top border of a window. Shows the window identifier, the name of the view displayed in the window, the system, the scope, the product reflected by the window, and the timeframe for which the data in the window is relevant. *See also* window status field.

**window number.** Sequential number assigned by MAINVIEW to each window when it is opened. The window number is the second character in the window status field. *See also* window status field.

**window status.** One-character letter in the window status field that indicates when a window is ready to receive commands, is busy processing commands, is not to be updated, or contains no data. It also indicates when an error has occurred in a window. The window status is the first character in the window status field. *See also* window information line, window status field.

**window status field.** Field on the window information line that shows the current status and assigned number of the window. *See also* window number, window status.

**windows mode.** Display of one or more MAINVIEW product views on a screen that can be divided into a maximum of 20 windows. A window information line defines the top border of each window. *Contrast with* full-screen mode.

**WLM workload.** In goal mode in MVS/SP 5.1 and later, a composite of service classes. MAINVIEW for OS/390 creates a workload for each WLM workload defined in the active service policy.

**workflow.** Measure of system activity that indicates how efficiently system resources are serving the jobs in a workload.

**workload.** (1) Systematic grouping of units of work (e.g., address spaces, CICS transactions, IMS transactions) according to classification criteria established by a system administrator. (2) In OS/390 or z/OS, a group of service classes within a service definition.

**workload activity view.** Tracks workload activity as the workload accesses system resources. A workload activity view measures workload activity in terms of resource consumption and how well the workload activity meets its service objectives.

**Workload Analyzer.** Online data collection and display services used to analyze IMS workloads and determine problem causes.

**workload definition.** Workload created through the WKLIST view. Contains a unique name, a description, an initial status, a current status, and selection criteria by which address spaces are selected for inclusion in the workload. *See* Workload Definition Facility.

**Workload Definition Facility.** In MAINVIEW for OS/390, WKLIST view and its associated dialogs through which workloads are defined and service objectives set.

**workload delay view.** Tracks workload performance as the workload accesses system resources. A workload delay view measures any delay a workload experiences as it contends for those resources.

**Workload Monitor.** Online data collection services used to monitor IMS workloads and issue warnings when defined thresholds are exceeded.

**workload objectives.** Performance goals for a workload, defined in WKLIST. Objectives can include measures of performance such as response times and batch turnaround times.



---

# Index

## A

- AAOMQL00
  - modifying settings 17
  - parameters 42
- AAOPRM00
  - modifying settings 17
  - parameters 39
- Action Specification - MQS
  - field description
    - Cmd(Type) 111
    - Destination QMgr 113
    - Destination Queue 113
    - DOM Id 112
    - EXEC Name/Parms 110
    - Info 112
    - Issue WTO Msg 112
    - Keep Message 113
    - Notify 112
    - Remove DLH 113
    - Set Variable 112
  - panel 59, 68, 69, 110
- Action Specification - MQS Panel 69
- Action Specification - MQS panel 110
- Action Specification Panel 68
- ACTN 114
- Alert Action(s) I - MQS panel 60, 69
- Alert Action(s) II - MQS panel 60, 70
- ALIAS\_BASE\_Q\_TYPE\_ERROR
  - IMFQ\_EVENT\_APPLNAME 74
  - IMFQ\_EVENT\_APPLTYPE 74
  - IMFQ\_EVENT\_BASEQNAME 74
  - IMFQ\_EVENT\_QMGRNAME 74
  - IMFQ\_EVENT\_QNAME 74
  - IMFQ\_EVENT\_QTYPE 74
  - instrumentation event 74
- APF-Authorized libraries
  - adding 17
- APL variables
  - APLCC 146
  - APLLN1 147
  - APLLNxx 147
  - APLNOL 147
  - APLRC 146
  - IMFRC 146
- APLCC
  - APL variable 146
- APLLN1
  - APL variable 147
- APLLNxx
  - APL variable 147
- APLNOL
  - APL variable 147
- APLRC
  - APL variable 146
- applications
  - communicating with MQSeries objects 156
- AUTHOREV
  - NOT\_AUTHORIZED TYPE 82
  - queue manager attribute 5
- Authority event type 5
- automating events 47
- automation
  - problems
    - examples 233
    - tracking MQS event statistics 114
    - viewing statistics 115
- Automation Control panel 52, 64
- Automation Power Line 143–148
  - APLCC variables 146
  - APLRC variables 146
  - CMD parameter 144
  - DEBUG parameter 145
  - examples 148
  - HELP parameter 145
  - IMFRC variables 146
  - NODE parameter 144
  - parameters 144
  - PORT parameter 144
  - RM parameter 144
  - WAIT parameter 145
- AutoOPERATOR
  - connecting to MQSeries 9
  - interaction with MQSeries 7–8
- AutoOPERATOR for MQSeries
  - activating 15
  - customizing 39
  - description 1
  - diagnosing errors 219
  - IMFEXEC statements
    - using 153
  - implementing 9
  - MQI EXEC interface
    - description 156
  - not monitoring 227
  - obtaining status 37
  - OCD 115
  - securing 151
  - software requirements 10
  - solutions 121
  - terms and concepts 2
  - workstation panel 115
- AutoOPERATOR-supplied variable
  - IMFOJOB 98
  - IMFOQMGR 98
  - IMFQ\_MD\_ACCOUNTINGTOKEN 101
  - IMFQ\_MD\_APPLIDENTITYDATA 101
  - IMFQ\_MD\_APPLORIGINDATA 102
  - IMFQ\_MD\_BACKOUTCOUNT 101
  - IMFQ\_MD\_CODEDCHARSETID 100
  - IMFQ\_MD\_CORRELID 101

IMFQ\_MD\_ENCODING 100  
 IMFQ\_MD\_EXPIRY 99  
 IMFQ\_MD\_FEEDBACK 100  
 IMFQ\_MD\_FORMAT 101  
 IMFQ\_MD\_GROUPID 102  
 IMFQ\_MD\_MSGFLAGS 103  
 IMFQ\_MD\_MSGID 101  
 IMFQ\_MD\_MSGSEQNUMBER 102  
 IMFQ\_MD\_MSGTYPE 99  
 IMFQ\_MD\_OFFSET 102  
 IMFQ\_MD\_ORIGINALLENGTH 103  
 IMFQ\_MD\_PERSISTENCE 101  
 IMFQ\_MD\_PRIORITY 101  
 IMFQ\_MD\_PUTAPPLNAME 102  
 IMFQ\_MD\_PUTAPPLTYPE 102  
 IMFQ\_MD\_PUTDATE 102  
 IMFQ\_MD\_PUTTIME 102  
 IMFQ\_MD\_REPLYTOQ 101  
 IMFQ\_MD\_REPLYTOQMGR 101  
 IMFQ\_MD\_REPORT 99  
 IMFQ\_MD\_STRUCID 98  
 IMFQ\_MD\_USERIDENTIFIER 101  
 IMFQ\_MD\_VERSION 99  
 IMFQ\_QAO\_CATCH\_UP\_MSG 98  
 IMFQ\_QATTR\_BACKOUTREQUEUEQNAME 103  
 IMFQ\_QATTR\_BACKOUTTHRESHOLD 103  
 IMFQ\_QATTR\_CREATIONDATE 103  
 IMFQ\_QATTR\_CREATIONTIME 103  
 IMFQ\_QATTR\_CURRENTQDEPTH 103  
 IMFQ\_QATTR\_DEFINITIONTYPE 103  
 IMFQ\_QATTR\_DEFINPUTOPENOPTION 103  
 IMFQ\_QATTR\_DEFPERSISTENCE 103  
 IMFQ\_QATTR\_DEFPRIORITY 103  
 IMFQ\_QATTR\_HARDENGETBACKOUT 103  
 IMFQ\_QATTR\_INHIBITGET 103  
 IMFQ\_QATTR\_INHIBITPUT 104  
 IMFQ\_QATTR\_INITIATIONQNAME 103  
 IMFQ\_QATTR\_MAXMSGLENGTH 104  
 IMFQ\_QATTR\_MAXQDEPTH 104  
 IMFQ\_QATTR\_MSGDELIVERYSEQUENCE 104  
 IMFQ\_QATTR\_OPENINPUTCOUNT 104  
 IMFQ\_QATTR\_OPENOUTPUTCOUNT 104  
 IMFQ\_QATTR\_PROCESSNAME 104  
 IMFQ\_QATTR\_QDESC 103  
 IMFQ\_QATTR\_RETENTIONINTERVAL 104  
 IMFQ\_QATTR\_SHAREABILITY 104  
 IMFQ\_QATTR\_STORAGECLASS 104  
 IMFQ\_QATTR\_TRIGGERCONTROL 104  
 IMFQ\_QATTR\_TRIGGERDAT 104  
 IMFQ\_QATTR\_TRIGGERDEPTH 104  
 IMFQ\_QATTR\_TRIGGERMSGPRIORITY 104  
 IMFQ\_QATTR\_TRIGGERTYPE 104  
 IMFQ\_QATTR\_USAGE 104  
 IMFQ\_STRUCTURES 98  
 IMFQCPF 98  
 IMFQRMT 98  
 AutoOPERATOR-supplied variables 98  
 AutoOPERATOR-supplied variables for all MQSeries  
 messages 98

## B

BACK (MQI command) 163  
 basic intercommunication solution 137  
   implementing 140  
   MQCOR000 Rule 138  
   MQCOR001 Rule 138  
   MQCOR006 Rule 138  
   MQCOR007 Rule 138  
   MQCOR008 Rule 138  
   MQCOR009 Rule 138  
   MQCOR010 Rule 139  
   MQCOR011 Rule 139  
   MQCOR012 Rule 139  
   MQCOR013 Rule 139  
   MQCOR014 Rule 139  
   MQCOR015 Rule 139  
   MQCOR016 Rule 139  
   MQCOR017 Rule 139  
   MQCOR018 Rule 139  
   MQCOR019 Rule 139  
   MQCORMOD Rule 138  
   QMQCORCF EXEC 138  
   stopping 141  
 BBI Control Command  
   .D ACTIVE 37  
   .D KEYS 37  
   .D PRODUCTS 37  
   .DISPLAY ACTIVE 37  
   .DISPLAY KEYS 37  
   .RESET MQ 37  
   .RESET MQ xx 37  
 BBI-SS PAS  
   restart 20  
 BBOMVAO.EXEC.REPLY 211  
 BBPARM  
   AAOMQL00 parameters 42  
   parameters for AAOPRM00 39  
 BRIDGE\_STARTED  
   IMFQ\_EVENT\_BRIDGENAME 74  
   IMFQ\_EVENT\_BRIDGETYPE 74  
   IMFQ\_EVENT\_QMGRNAME 74  
   instrumentation event 74  
 BRIDGE\_STOPPED  
   IMFQ\_EVENT\_BRIDGENAME 75  
   IMFQ\_EVENT\_BRIDGETYPE 75  
   IMFQ\_EVENT\_ERRORIDENTIFIER 75  
   IMFQ\_EVENT\_QMGRNAME 75  
   IMFQ\_EVENT\_REASONQUALIFIER 75  
   instrumentation event 75  
 BUFFER 189  
 BUFFLEN 189

## C

CHADEV  
   queue manager attribute 5  
 channel availability solution 134  
   implementing 135  
   MQP00001 134

- MQP2023C 134
- MQP2023D 134
- MQP20636 134
- MQP20832 134
- MQP2083C 134
- MQP54500 134
- MQPRMT01 134
- MQTCK001 134
- MQTCK001 Rule 135
- Rule IDs 134
- stopping 136
- channel event
  - description 5
- CHANNEL\_ACTIVATED
  - IMFQ\_EVENT\_CHANNELNAME 75
  - IMFQ\_EVENT\_CONNECTIONNAME 75
  - IMFQ\_EVENT\_XMITQNAME 75
  - instrumentation event 75
- CHANNEL\_ACTIVATEDIMFQ\_EVENT\_QMGRNAME 75
- CHANNEL\_CONV\_ERROR
  - IMFQ\_EVENT\_CHANNELNAME 76
  - IMFQ\_EVENT\_CONNECTIONNAME 76
  - IMFQ\_EVENT\_FORMAT 76
  - IMFQ\_EVENT\_QMGRNAME 76
  - IMFQ\_EVENT\_REASONQUALIFIER 76
  - IMFQ\_EVENT\_XMITQNAME 76
  - instrumentation event 76
- CHANNEL\_NOT\_ACTIVATED
  - IMFQ\_EVENT\_CHANNELNAME 77
  - IMFQ\_EVENT\_CONNECTIONNAME 77
  - IMFQ\_EVENT\_QMGRNAME 77
  - IMFQ\_EVENT\_XMITQNAME 77
  - instrumentation event 77
- CHANNEL\_START event 134
- CHANNEL\_STARTED
  - IMFQ\_EVENT\_CHANNELNAME 77
  - IMFQ\_EVENT\_CONNECTIONNAME 77
  - IMFQ\_EVENT\_QMGRNAME 77
  - IMFQ\_EVENT\_XMITQNAME 77
  - instrumentation event 77
- CHANNEL\_STOP event 134
- CHANNEL\_STOPPED
  - IMFQ\_EVENT\_AUXERRORDATAINT1 78
  - IMFQ\_EVENT\_AUXERRORDATAINT2 78
  - IMFQ\_EVENT\_AUXERRORDATASTR1 78
  - IMFQ\_EVENT\_AUXERRORDATASTR2 78
  - IMFQ\_EVENT\_AUXERRORDATASTR3 78
  - IMFQ\_EVENT\_CHANNELNAME 78
  - IMFQ\_EVENT\_CONNECTIONNAME 78
  - IMFQ\_EVENT\_ERRORIDENTIFIER 78
  - IMFQ\_EVENT\_QMGRNAME 78
  - IMFQ\_EVENT\_REASONQUALIFIER 78
  - IMFQ\_EVENT\_XMITQNAME 78
  - instrumentation event 78
- checklist
  - diagnostic 222
- CLOSE (MQI command)
  - Example 164

- CMD
  - installation verification 32
- CMD parameter
  - Power Line 144
- CMD Type(MQS)
  - examples 213
- CMIT (MQI command)
  - Example 167
- coding
  - sample 241
- Command MQ Automation Power Line 143–148
- Command MQ On Ramp
  - color diagnosis 26
  - description 27
  - diagnosing errors 27
  - hyperlinking 23
  - logging off 27
  - menu options 24
  - queue manager connection 25
  - removing non-MVS queue managers 25
  - specifying MVS queue manager 24
  - specifying non-MVS queue manager 24
  - user interface 23
- commands
  - BBI control 37
- communicating with MQSeries objects 156
- completion code
  - returning 156
- Confirm Rule Set Modifications panel 62, 72
- CONN (MQI command)
  - Example 169
- connecting to MQSeries 9
- connection
  - manual 28
  - status color 26
  - using the Command MQ On Ramp 25
- connection handle 156
- connectivity
  - defining 22
  - setting up 22
- COPY MQI command)
  - Example 214
- CQ parameter
  - CMD 211
- CSQ90221 message 211

## D

- .D ACTIVE 37
- .DISPLAY ACTIVE 37
- .D KEYS 37
- .DISPLAY KEYS 37
- .D PRODUCTS 37
- .DISPLAY PRODUCTS 37
- dead letter
  - valid
    - Dead Letter Management 124
- Dead Letter Aging Management
  - enabling 125

- Dead Letter Management
  - enabling Rules 123
  - invalid dead letters 123
  - valid dead letters 124
- dead letter queue 3
- Dead Letter Queue solution 122
  - Dead Letter Management 123
  - implementing 122
  - stopping 125
- dead letters
  - invalid
    - Dead Letter Management 123
- DEBUG
  - Power Line parameter 145
- DEF\_XMIT\_Q\_TYPE\_ERROR
  - IMFQ\_EVENT\_APPLNAME 79
  - IMFQ\_EVENT\_APPLTYPE 79
  - IMFQ\_EVENT\_OBJECTQMGRNAME 79
  - IMFQ\_EVENT\_QMGRNAME 79
  - IMFQ\_EVENT\_QNAME 79
  - IMFQ\_EVENT\_QTYPE 79
  - IMFQ\_EVENT\_XMITQNAME 79
  - instrumentation event 79
- DEF\_XMIT\_Q\_USAGE\_ERROR
  - IMFQ\_EVENT\_APPLNAME 80
  - IMFQ\_EVENT\_APPLTYPE 80
  - IMFQ\_EVENT\_OBJECTQMGRNAME 80
  - IMFQ\_EVENT\_QMGRNAME 80
  - IMFQ\_EVENT\_QNAME 80
  - IMFQ\_EVENT\_XMITQNAME 80
  - instrumentation event 80
- Define Queue for Non-OS/390 Instrumentation Events 20
- defining connectivity 22
- diagnosing errors
  - AutoOPERATOR for MQSeries 219
  - Command MQ On Ramp 27
  - tools 220
- diagnostic checklist 222
- diagnostic rules
  - enabling 19
- DISC (MQI command)
  - Example 171
- DISPLAY MQI 153
- DISPLAY MQI (command)
  - Example 217

## E

- enabling MQSeries events 50
- error messages
  - resources 219
- event message
  - description 48
- events
  - automating 47
  - automation 1
  - channel 5
  - description 3
  - instrumentation 73–96

- listening 29
- not firing 229
- performance 5
- queue manager 5
- tracking automation statistics 114
- types 5
- variables 97
- EXEC
  - times out
    - non-MVS queue manager 237

## F

- frequently asked questions 223

## G

- generating
  - instrumentation events 40
  - MQEVENTs 40
- GET (MQI command)
  - Example 173
- Get(Disabled)
  - queues set to 41
- GET\_INHIBITED
  - IMFQ\_EVENT\_APPLNAME 81
  - IMFQ\_EVENT\_APPLTYPE 81
  - IMFQ\_EVENT\_QMGRNAME 81
  - instrumentation event 81
  - MFQ\_EVENT\_QNAME 81

## H

- HCONN 183, 187
- HELP
  - Power Line parameter 145
- HOBjparameters
  - HOBj 187
- hyperlinking
  - Command MQ On Ramp 23

## I

- IE
  - installation verification 32
- IMFEXEC
  - CMD 209
  - CMD Type(MQS)
    - examples 213
- IMFEXEC CMD
  - CQ parameter 211
  - LM parameter 209
  - MQ command parameter 209
  - PCF parameter 210
  - QM parameter 210
  - Response parameter 210
  - Type parameter 209
  - Wait parameter 210
- IMFEXEC MQI commands (description of)

COPY MQI 153  
 IMFEXEC MQI PUT1 parameters  
     HCONN 197  
 IMFEXEC MQI Variables 154  
 IMFMQI 180  
 IMFMQI\_MD\_ACCOUNTINGTOKEN 180, 193, 204  
 IMFMQI\_MD\_APPLIDENTITYDATA 180, 193, 204  
 IMFMQI\_MD\_APPLORIGINDATA 180, 195  
 IMFMQI\_MD\_BACKOUTCOUNT 179, 193, 203  
 IMFMQI\_MD\_CODEDCHARSETID 178, 191, 202  
 IMFMQI\_MD\_CORRELID 179, 192, 203  
 IMFMQI\_MD\_ENCODING 178, 191, 202  
 IMFMQI\_MD\_EXPIRY 177, 202  
 IMFMQI\_MD\_EXPIRYvariables  
     IMFMQI\_MD\_EXPIRY 191  
 IMFMQI\_MD\_FEEDBACK 178, 191, 202  
 IMFMQI\_MD\_FORMAT 179, 192, 203  
 IMFMQI\_MD\_GROUPID 181, 195  
 IMFMQI\_MD\_MSGFLAGS 181, 195  
 IMFMQI\_MD\_MSGID 179, 192, 203  
 IMFMQI\_MD\_MSGSEQNUMBER 181, 195  
 IMFMQI\_MD\_MSGTYPE 177, 190, 201  
 IMFMQI\_MD\_OFFSET 181, 195  
 IMFMQI\_MD\_ORIGINALLENGTH 181, 195  
 IMFMQI\_MD\_PERSISTENCE 179, 192, 203  
 IMFMQI\_MD\_PRIORITY 179, 192, 203  
 IMFMQI\_MD\_PUTAPPLNAME 180, 194  
 IMFMQI\_MD\_PUTAPPLTYPE 180, 194  
 IMFMQI\_MD\_PUTDATE 180, 194  
 IMFMQI\_MD\_PUTTIME 180, 194  
 IMFMQI\_MD\_REPLYTOQ 179, 193, 203  
 IMFMQI\_MD\_REPLYTOQMGR 180, 193, 204  
 IMFMQI\_MD\_REPORT 177, 190, 201  
 IMFMQI\_MD\_STRUCID 176, 190, 201  
 IMFMQI\_MD\_USERIDENTIFIER 180, 193, 204  
 IMFMQI\_MD\_VERSION 176, 190, 201  
 IMFMQI\_OD\_ALTERNATESECURITYID 186, 201  
 IMFMQI\_OD\_ALTERNATEUSERID 185, 200  
 IMFMQI\_OD\_DYNAMICQNAME 185, 200  
 IMFMQI\_OD\_INVALIDDESTCOUNT 185, 200  
 IMFMQI\_OD\_KNOWNDESTCOUNT 185, 200  
 IMFMQI\_OD\_OBJECTNAME 185, 200  
 IMFMQI\_OD\_OBJECTQMGRNAME 185, 200  
 IMFMQI\_OD\_OBJECTRECOFFSET 185, 200  
 IMFMQI\_OD\_OBJECTRECPTR 185, 201  
 IMFMQI\_OD\_OBJECTTYPE 185, 200  
 IMFMQI\_OD\_RECSPRESENT 185, 200  
 IMFMQI\_OD\_RESOLVEDQMGRNAME 186, 201  
 IMFMQI\_OD\_RESOLVEDQNAME 186, 201  
 IMFMQI\_OD\_RESPONSERECOFFSET 185, 200  
 IMFMQI\_OD\_RESPONSERECPTR 185, 201  
 IMFMQI\_OD\_STRUCID 185, 200  
 IMFMQI\_OD\_UNKNOWNDESTCOUNT 185, 200  
 IMFMQI\_OD\_VERSION 185, 200  
 IMFMQI\_STRUCTURES 176  
 IMFOJOB 98  
 IMFOQMGR 98  
 IMFQ\_EVENT\_APPLNAME  
     ALIAS\_BASE\_Q\_TYPE\_ERROR 74

DEF\_XMIT\_Q\_TYPE\_ERROR 79  
 DEF\_XMIT\_Q\_USAGE\_ERROR 80  
 GET\_INHIBITED 81  
 NOT\_AUTHORIZED 82  
 PUT\_INHIBITED 84  
 Q\_TYPE\_ERROR 88  
 REMOTE\_Q\_NAME\_ERROR 89  
 UNKNOWN\_ALIAS\_BASE\_Q 90  
 UNKNOWN\_DEF\_XMIT\_Q 91  
 UNKNOWN\_OBJECT\_NAME 92  
 UNKNOWN\_REMOTE\_Q\_MGR 93  
 UNKNOWN\_XMIT\_Q 94  
 XMIT\_Q\_USAGE\_ERROR 96  
 IMFQ\_EVENT\_APPLTYPE  
     ALIAS\_BASE\_Q\_TYPE\_ERROR 74  
     DEF\_XMIT\_Q\_TYPE\_ERROR 79  
     DEF\_XMIT\_Q\_USAGE\_ERROR 80  
     GET\_INHIBITED 81  
     NOT\_AUTHORIZED 82  
     PUT\_INHIBITED 84  
     Q\_TYPE\_ERROR 88  
     REMOTE\_Q\_NAME\_ERROR 89  
     UNKNOWN\_ALIAS\_BASE\_Q 90  
     UNKNOWN\_DEF\_XMIT\_Q 91  
     UNKNOWN\_OBJECT\_NAME 92  
     UNKNOWN\_REMOTE\_Q\_MGR 93  
     UNKNOWN\_XMIT\_Q 94  
     XMIT\_Q\_TYPE\_ERROR 95  
     XMIT\_Q\_USAGE\_ERROR 96  
 IMFQ\_EVENT\_AUXERRORDATAINT1  
     CHANNEL\_STOPPED 78  
 IMFQ\_EVENT\_AUXERRORDATAINT2  
     CHANNEL\_STOPPED 78  
 IMFQ\_EVENT\_AUXERRORDATASTR1  
     CHANNEL\_STOPPED 78  
 IMFQ\_EVENT\_AUXERRORDATASTR2  
     CHANNEL\_STOPPED 78  
 IMFQ\_EVENT\_AUXERRORDATASTR3  
     CHANNEL\_STOPPED 78  
 IMFQ\_EVENT\_BASEQNAME  
     ALIAS\_BASE\_Q\_TYPE\_ERROR 74  
     Q\_DEPTH\_HIGH 85  
     Q\_DEPTH\_LOW 85  
     Q\_FULL 86  
     Q\_SERVICE\_INTERVAL\_HIGH 87  
     UNKNOWN\_ALIAS\_BASE\_Q 90  
 IMFQ\_EVENT\_BRIDGENAME  
     BRIDGE\_STARTED 74  
     BRIDGE\_STOPPED 75  
 IMFQ\_EVENT\_BRIDGETYPE  
     BRIDGE\_STARTED 74  
     BRIDGE\_STOPPED 75  
 IMFQ\_EVENT\_CHANNELNAME  
     CHANNEL\_ACTIVATED 75  
     CHANNEL\_CONV\_ERROR 76  
     CHANNEL\_NOT\_ACTIVATED 77  
     CHANNEL\_STARTED 77  
     CHANNEL\_STOPPED 78  
 IMFQ\_EVENT\_COMMAND

NOT\_AUTHORIZED 83  
 IMFQ\_EVENT\_CONNECTIONNAME  
   CHANNEL\_ACTIVATED 75  
   CHANNEL\_CONV\_ERROR 76  
   CHANNEL\_NOT\_ACTIVATED 77  
   CHANNEL\_STARTED 77  
   CHANNEL\_STOPPED 78  
 IMFQ\_EVENT\_ERRORIDENTIFIER  
   BRIDGE\_STOPPED 75  
   CHANNEL\_STOPPED 78  
 IMFQ\_EVENT\_FORMAT  
   CHANNEL\_CONV\_ERROR 76  
 IMFQ\_EVENT\_HIGHQDEPTH  
   Q\_DEPTH\_HIGH 85  
   Q\_DEPTH\_LOW 85  
   Q\_FULL 86  
   Q\_SERVICE\_INTERVAL\_HIGH 87  
 IMFQ\_EVENT\_MSGDEQCOUNT  
   Q\_DEPTH\_HIGH 85  
   Q\_DEPTH\_LOW 85  
   Q\_FULL 86  
   Q\_SERVICE\_INTERVAL\_HIGH 87  
 IMFQ\_EVENT\_MSGENQCOUNT  
   Q\_DEPTH\_HIGH 85  
   Q\_DEPTH\_LOW 85  
   Q\_FULL 86  
   Q\_SERVICE\_INTERVAL\_HIGH 87  
 IMFQ\_EVENT\_OBJECTQMGRNAME  
   DEF\_XMIT\_Q\_TYPE\_ERROR 79  
   DEF\_XMIT\_Q\_USAGE\_ERROR 80  
   NOT\_AUTHORIZED 83  
   PUT\_INHIBITED 84  
   Q\_TYPE\_ERROR 88  
   REMOTE\_Q\_NAME\_ERROR 89  
   UNKNOWN\_ALIAS\_BASE\_Q 90  
   UNKNOWN\_DEF\_XMIT\_Q 91  
   UNKNOWN\_OBJECT\_NAME 92  
   UNKNOWN\_REMOTE\_Q\_MGR 93  
   UNKNOWN\_XMIT\_Q 94  
   XMIT\_Q\_TYPE\_ERROR 95  
   XMIT\_Q\_USAGE\_ERROR 96  
 IMFQ\_EVENT\_OPTIONS  
   NOT\_AUTHORIZED 82  
 IMFQ\_EVENT\_PROCESSNAME  
   NOT\_AUTHORIZED 83  
   UNKNOWN\_OBJECT\_NAME 92  
 IMFQ\_EVENT\_QMGRNAME  
   ALIAS\_BASE\_Q\_TYPE\_ERROR 74  
   BRIDGE\_STOPPED 75  
   BRIDGE\_STARTED 74  
   CHANNEL\_ACTIVATED 75  
   CHANNEL\_CONV\_ERROR 76  
   CHANNEL\_NOT\_ACTIVATED 77  
   CHANNEL\_STARTED 77  
   CHANNEL\_STOPPED 78  
   DEF\_XMIT\_Q\_TYPE\_ERROR 79  
   DEF\_XMIT\_Q\_USAGE\_ERROR 80  
   GET\_INHIBITED 81  
   NOT\_AUTHORIZED 82  
   PUT\_INHIBITED 84  
   Q\_DEPTH\_HIGH 85  
   Q\_DEPTH\_LOW 85  
   Q\_FULL 86  
   Q\_MGR\_ACTIVE 86  
   Q\_MGR\_NOT\_ACTIVE 86  
   Q\_SERVICE\_INTERVAL\_HIGH 87  
   Q\_TYPE\_ERROR 88  
   REMOTE\_Q\_NAME\_ERROR 89  
   UNKNOWN\_ALIAS\_BASE\_Q 90  
   UNKNOWN\_DEF\_XMIT\_Q 91  
   UNKNOWN\_OBJECT\_NAME 92  
   UNKNOWN\_REMOTE\_Q\_MGR 93  
   UNKNOWN\_XMIT\_Q 94  
   XMIT\_Q\_TYPE\_ERROR 95  
   XMIT\_Q\_USAGE\_ERROR 96  
 IMFQ\_EVENT\_QNAME  
   ALIAS\_BASE\_Q\_TYPE\_ERROR 74  
   DEF\_XMIT\_Q\_TYPE\_ERROR 79  
   DEF\_XMIT\_Q\_USAGE\_ERROR 80  
   NOT\_AUTHORIZED 82  
   PUT\_INHIBITED 84  
   Q\_TYPE\_ERROR 88  
   UNKNOWN\_ALIAS\_BASE\_Q 90  
   UNKNOWN\_DEF\_XMIT\_Q 91  
   UNKNOWN\_OBJECT\_NAME 92  
   UNKNOWN\_REMOTE\_Q\_MGR 93  
   UNKNOWN\_XMIT\_Q 94  
   XMIT\_Q\_TYPE\_ERROR 95  
   XMIT\_Q\_USAGE\_ERROR 96  
 IMFQ\_EVENT\_QTYPE  
   ALIAS\_BASE\_Q\_TYPE\_ERROR 74  
   DEF\_XMIT\_Q\_TYPE\_ERROR 79  
   XMIT\_Q\_TYPE\_ERROR 95  
 IMFQ\_EVENT\_REASONQUALIFIER  
   BRIDGE\_STOPPED 75  
   CHANNEL\_CONV\_ERROR 76  
   CHANNEL\_STOPPED 78  
   NOT\_AUTHORIZED 82  
   Q\_MGR\_NOT\_ACTIVE 86  
 IMFQ\_EVENT\_TIMESINCERESET  
   Q\_DEPTH\_HIGH 85  
   Q\_DEPTH\_LOW 85  
   Q\_FULL 86  
   Q\_SERVICE\_INTERVAL\_HIGH 87  
 IMFQ\_EVENT\_USERIDENTIFIER  
   NOT\_AUTHORIZED 83  
 IMFQ\_EVENT\_XMITQNAME  
   CHANNEL\_ACTIVATED 75  
   CHANNEL\_CONV\_ERROR 76  
   CHANNEL\_NOT\_ACTIVATED 77  
   CHANNEL\_STARTED 77  
   CHANNEL\_STOPPED 78  
   DEF\_XMIT\_Q\_TYPE\_ERROR 79  
   DEF\_XMIT\_Q\_USAGE\_ERROR 80  
   UNKNOWN\_DEF\_XMIT\_Q 91  
   UNKNOWN\_XMIT\_Q 94  
   XMIT\_Q\_TYPE\_ERROR 95  
   XMIT\_Q\_USAGE\_ERROR 96

IMFQ\_MD\_ACCOUNTINGTOKEN 101  
 IMFQ\_MD\_APPLIDENTITYDATA 101  
 IMFQ\_MD\_APPLORIGINDATA 102  
 IMFQ\_MD\_BACKOUTCOUNT 101  
 IMFQ\_MD\_CODEDCHARSETID 100  
 IMFQ\_MD\_CORRELID 101  
 IMFQ\_MD\_ENCODING 100  
 IMFQ\_MD\_EXPIRY 99  
 IMFQ\_MD\_FEEDBACK 100  
 IMFQ\_MD\_FORMAT 101  
 IMFQ\_MD\_GROUPID 102  
 IMFQ\_MD\_MSGFLAGS 103  
 IMFQ\_MD\_MSGID 101  
 IMFQ\_MD\_MSGSEQNUMBER 102  
 IMFQ\_MD\_MSGTYPE 99  
 IMFQ\_MD\_OFFSET 102  
 IMFQ\_MD\_ORIGINALLENGTH 103  
 IMFQ\_MD\_PERSISTENCE 101  
 IMFQ\_MD\_PRIORITY 101  
 IMFQ\_MD\_PUTAPPLNAME 102  
 IMFQ\_MD\_PUTAPPLTYPE 102  
 IMFQ\_MD\_PUTDATE 102  
 IMFQ\_MD\_PUTTIME 102  
 IMFQ\_MD\_REPLYTOQ 101  
 IMFQ\_MD\_REPLYTOQMGR 101  
 IMFQ\_MD\_REPORT 99  
 IMFQ\_MD\_STRUCID 98  
 IMFQ\_MD\_USERIDENTIFIER 101  
 IMFQ\_MD\_VERSION 99  
 IMFQ\_QAO\_CATCH\_UP\_MSG 98  
 IMFQ\_QATTR\_BACKOUTREQUEUEQNAME 103  
 IMFQ\_QATTR\_BACKOUTTHRESHOLD 103  
 IMFQ\_QATTR\_CREATIONDATE 103  
 IMFQ\_QATTR\_CREATIONTIME 103  
 IMFQ\_QATTR\_CURRENTQDEPTH 103  
 IMFQ\_QATTR\_DEFINITIONTYPE 103  
 IMFQ\_QATTR\_DEFINPUTOPENOPTION 103  
 IMFQ\_QATTR\_DEFPERSISTENCE 103  
 IMFQ\_QATTR\_DEFPPRIORITY 103  
 IMFQ\_QATTR\_HARDENGETBACKOUT 103  
 IMFQ\_QATTR\_INHIBITGET 103  
 IMFQ\_QATTR\_INHIBITPUT 104  
 IMFQ\_QATTR\_INITIATIONQNAME 103  
 IMFQ\_QATTR\_MAXMSGLENGTH 104  
 IMFQ\_QATTR\_MAXQDEPTH 104  
 IMFQ\_QATTR\_MSGDELIVERYSEQUENCE 104  
 IMFQ\_QATTR\_OPENINPUTCOUNT 104  
 IMFQ\_QATTR\_OPENOUTPUTCOUNT 104  
 IMFQ\_QATTR\_PROCESSNAME 104  
 IMFQ\_QATTR\_QDESC 103  
 IMFQ\_QATTR\_RETENTIONINTERVAL 104  
 IMFQ\_QATTR\_SHAREABILITY 104  
 IMFQ\_QATTR\_STORAGECLASS 104  
 IMFQ\_QATTR\_TRIGGERCONTROL 104  
 IMFQ\_QATTR\_TRIGGERDAT 104  
 IMFQ\_QATTR\_TRIGGERDEPTH 104  
 IMFQ\_QATTR\_TRIGGERMSGPRIORITY 104  
 IMFQ\_QATTR\_TRIGGERTYPE 104  
 IMFQ\_QATTR\_USAGE 104

IMFQ\_STRUCTURES 98  
 IMFQAPLN  
     PUT\_INHIBITED 74, 79, 80, 81, 82, 84, 88, 89, 90, 91,  
     92, 93, 94, 95, 96  
     XMIT\_Q\_TYPE\_ERROR 95  
 IMFQCPF 98  
 IMFQRMT 98  
 IMFRC  
     APL variable 146  
 IMFTEXT  
     message size limitation 63  
 Inhibit  
     event type 5  
 INHIBTEV  
     GET\_INHIBITED 81  
     PUT\_INHIBITED 84  
     queue manager attribute 5  
 installation  
     troubleshooting 223  
 installation verification procedure  
     description 32  
     examples 33  
     required values 32  
 instrumentation events  
     categories 5  
     description 48, 73–96  
     enabling  
         MVS 21  
         non-MVS 21  
     generating 40  
     types 5, 21  
 interface  
     MQSeries EXEC 154  
 invalid dead letters  
     Dead Letter Management 123  
 issuing commands  
     set up 30

**K**

keys  
     specifying 15

**L**

libraries  
     adding 17  
 listening for events  
     set up 29  
 LM parameter  
     CMD 209  
 Local  
     event type 5  
 local queue manager 2  
 LOCALEV  
     ALIAS\_BASE\_Q\_TYPE\_ERROR 74  
     queue manager attribute 5  
     UNKNOWN\_ALIAS\_BASE\_Q 90  
     UNKNOWN\_OBJECT\_NAME 92

logging off  
Command MQ On Ramp 27

## M

message

CSQ9022I 211  
description 6, 48  
format 6, 48  
non-event 48

message channel 3

Message Queue Interface (MQI) 2

messaging 1–2

MFQ\_EVENT\_QNAME

GET\_INHIBITED 81

REMOTE\_Q\_NAME\_ERROR 89

migration considerations 11

MQ command parameter

CMD 209

MQ PUT

non-zero code return 231

MQCOR000 Rule 138

MQCOR001 Rule 138

MQCOR006 Rule 138

MQCOR007 Rule 138

MQCOR008 Rule 138

MQCOR009 Rule 138

MQCOR010 Rule 139

MQCOR011 Rule 139

MQCOR012 Rule 139

MQCOR013 Rule 139

MQCOR014 Rule 139

MQCOR015 Rule 139

MQCOR016 Rule 139

MQCOR017 Rule 139

MQCOR018 Rule 139

MQCOR019 Rule 139

MQCORMOD Rule 138

MQDEDQ01 Rule

enabling 123

MQDEDQ02 Rule

enabling 125

MQDEDQ03 Rule

enabling 124

MQDIA001

enabling 19

MQDIA002

enabling 19

MQDIA003

enabling 19

MQEV

parameters 39

MQEVENT

creating Rules 52

description 48

generating 40

MQFMT\_EVENT message 51

MQGINHIB 41

parameters 39

MQI BACK 153

MQI CLOSE 153

MQI COMMIT 153

MQI Commands 154

MQI commands (description of)

COPY MQI 153

MQI CONN 153

MQI DISC 153

MQI EXEC interface

description 156

MQI GET 153

MQI OPEN 153

MQI PUT 153

MQI PUT1 153

MQI sample coding 241

MQIEXEC Interface 156

MQINT001 Rule

enabling 130

MQINT002 Rule

enabling 130

mqmd\_UserIdentifier 211

MQNSHARE 41

parameters 40

MQP00001 134

MQP2023C 134

MQP2023D 134

MQP20636 134

MQP20832 134

MQP2083C 134

MQP54500 134

MQPRMT01 Rule 134

MQQDP001

enabling 132

MQQDP002

enabling 132

MQS events description 48

MQS Selection Criteria 105

MQSeries

advantages 2

description 1

interaction with AutoOPERATOR 7–8

terms and concepts 2

workstation 115

MQSeries Event Types panel 55, 57

MQSeries events

automation 1, 47

description 3

enabling 50

tracking automation statistics 114

variables 97

MQSeries EXEC interface 154

MQSeries Instrumentation

events 74

MQSeries messages

format 48

MQSeries Objects 156

MQSeries OCD

queues managed 120

MQTCK001 134

## N

NCMD  
  installation verification 32  
NIE  
  installation verification 32  
NODE  
  Power Line parameter 144  
Non-MQFMT\_EVENT Message  
  Creating a rule for 63  
Non-MQFMT\_EVENT Message Rule MQSeries Message  
  creating an example 64  
non-MVS queue manager  
  removing from Command MQ On Ramp 25  
  specifying in Command MQ On Ramp 24  
NOSHARE  
  queues set as 41  
NOT\_AUTHORIZED  
  IMFQ\_EVENT\_APPLNAME 82  
  IMFQ\_EVENT\_APPLTYPE 82  
  IMFQ\_EVENT\_COMMAND 83  
  IMFQ\_EVENT\_OBJECTQMGRNAME 83  
  IMFQ\_EVENT\_OPTIONS 82  
  IMFQ\_EVENT\_PROCESSNAME 83  
  IMFQ\_EVENT\_QMGRNAME 82  
  IMFQ\_EVENT\_QNAME 82  
  IMFQ\_EVENT\_REASONQUALIFIER 82  
  IMFQ\_EVENT\_USERIDENTIFIER 83  
  instrumentation event 82

## O

object handle 156  
OPEN (MQI command)  
  Example 183

## P

panel  
  Action Specification - MQS 59, 68, 69, 110  
    description, Action Specification for MQS Panel 110  
  Alert Action(s) I 60, 69  
  Alert Action(s) II 60, 70  
  Automation Control 52, 64  
  Confirm Rule Set Modifications 62, 72  
  MQSeries Event Types 55, 57  
  Rule Processor Detail Control 53–54, 60–61, 65–66, 70  
  Rule Set Overview 53, 65, 71  
  Selection Criteria - MQS 54, 58, 66–67, 105  
    description 105  
  Variable Dependencies - MQS 58, 68  
parameters  
  BUFFER 189  
  BUFFLEN 189  
  HCONN 183, 187  
password keys  
  specifying 15  
PCF parameter  
  CMD 210  
PERFMEV

  Q\_DEPTH\_LOW 85  
  Q\_DEPTHI\_HIGH 85  
  Q\_FULL 86  
  Q\_SERVICE\_INTERVAL\_HIGH 87  
  queue manager attribute 5  
performance events  
  description 5  
performance statistics 118  
  field descriptions 118  
PFC Variables 269  
POPTS 188  
PORT  
  Power Line parameter 144  
Power Line 143–148  
  APLCC variables 146  
  APLRC variables 146  
  CMD parameter 144  
  DEBUG parameter 145  
  examples 148  
  HELP parameter 145  
  IMFRC variables 146  
  NODE parameter 144  
  PORT parameter 144  
  RM parameter 144  
  WAIT parameter 145  
primary commands  
  workstation panel 116  
PUT (MQI command)  
  Example 187  
PUT\_INHIBITED  
  IMFQ\_EVENT\_APPLNAME 84  
  IMFQ\_EVENT\_APPLTYPE 84  
  IMFQ\_EVENT\_OBJECTQMGRNAME 84  
  IMFQ\_EVENT\_QMGRNAME 84  
  IMFQ\_EVENT\_QNAME 84  
  IMFQAPLN 74, 79–82, 84, 88,–96  
  instrumentation event 84  
PUTI (MQI command)  
  Example 197

## Q

Q\_DEPTH\_HIGH  
  IMFQ\_EVENT\_BASEQNAME 85  
  IMFQ\_EVENT\_HIGHQDEPTH 85  
  IMFQ\_EVENT\_MSGDEQCOUNT 85  
  IMFQ\_EVENT\_MSGGENQCOUNT 85  
  IMFQ\_EVENT\_QMGRNAME 85  
  IMFQ\_EVENT\_TIMESINCERESET 85  
  instrumentation event 85  
Q\_DEPTH\_LOW  
  IMFQ\_EVENT\_BASEQNAME 85  
  IMFQ\_EVENT\_HIGHQDEPTH 85  
  IMFQ\_EVENT\_MSGDEQCOUNT 85  
  IMFQ\_EVENT\_MSGGENQCOUNT 85  
  IMFQ\_EVENT\_QMGRNAME 85  
  IMFQ\_EVENT\_TIMESINCERESET 85  
  instrumentation event 85  
Q\_FULL

- IMFQ\_EVENT\_BASEQNAME 86
- IMFQ\_EVENT\_HIGHQDEPTH 86
- IMFQ\_EVENT\_MSGDEQCOUNT 86
- IMFQ\_EVENT\_MSGENQCOUNT 86
- IMFQ\_EVENT\_QMGRNAME 86
- IMFQ\_EVENT\_TIMESINCERESET 86
- instrumentation event 86
- Q\_MGR\_ACTIVE
  - IMFQ\_EVENT\_QMGRNAME 86
  - instrumentation event 86
- Q\_MGR\_NOT\_ACTIVE
  - IMFQ\_EVENT\_QMGRNAME 86
  - IMFQ\_EVENT\_REASONQUALIFIER 86
  - instrumentation event 86
- Q\_SERVICE\_INTERVAL\_HIGH
  - IMFQ\_EVENT\_BASEQNAME 87
  - IMFQ\_EVENT\_HIGHQDEPTH 87
  - IMFQ\_EVENT\_MSGDEQCOUNT 87
  - IMFQ\_EVENT\_MSGENQCOUNT 87
  - IMFQ\_EVENT\_QMGRNAME 87
  - IMFQ\_EVENT\_TIMESINCERESET 87
  - instrumentation event 87
- Q\_TYPE\_ERROR
  - IMFQ\_EVENT\_APPLNAME 88
  - IMFQ\_EVENT\_APPLTYPE 88
  - IMFQ\_EVENT\_OBJECTQMGRNAME 88
  - IMFQ\_EVENT\_QMGRNAME 88
  - IMFQ\_EVENT\_QNAME 88
  - instrumentation event 88
- QM parameter
  - CMD 210
- QMQCORCF EXEC 138
- QMDEDQ1 EXEC
  - invoking 123
- QMQNB001
  - sample coding 241
- QMQNB002
  - sample coding 241
- QMQNB003
  - sample coding 241
- QMQNB004
  - sample coding 242
- QMQNB005
  - sample coding 242
- QMQNB006
  - sample coding 242
- QMQNB007
  - sample coding 242
- QMQNB008
  - sample coding 242
- QMQNB009
  - sample coding 242
- QMQUULDQ 121
- queue
  - dead letter 3
- queue depth management solution
  - enabling Rules 132
  - implementing 132
  - stopping 133

- queue management
  - viewing
    - workstation panel 120
- queue manager
  - communicating 3
  - connecting manually 28
  - connecting with Command MQ On Ramp 25
  - description 2
  - events 5
  - local 2
  - not connecting 225
  - remote 3
  - setting up connectivity 22
  - specifying in Command MQ On Ramp 24
- queue manager attribute
  - AUTHOREV 5
  - CHADEV 5
  - INHIBTEV 5
  - LOCALEV 5
  - PERFMEV 5
  - REMOTEV 5
  - STRSTPEV 5
- queues
  - specifying in AAOMQL00 42
    - examples 44
- queuing 1, 2

## R

- reason code
  - returning 156
- refresh
  - list of queues 18
- Remote
  - event type 5
- remote queue manager 3
- REMOTE\_Q\_NAME\_ERROR
  - IMFQ\_EVENT\_APPLNAME 89
  - IMFQ\_EVENT\_APPLTYPE 89
  - IMFQ\_EVENT\_OBJECTQMGRNAME 89
  - IMFQ\_EVENT\_QMGRNAME 89
  - instrumentation event 89
- REMOTE\_Q\_NAME\_ERROR\_MFQ\_EVENT\_QNAME 89
- REMOTEEV
  - DEF\_XMIT\_Q\_TYPE\_ERROR 79
  - DEF\_XMIT\_Q\_USAGE\_ERROR 80
  - Q\_TYPE\_ERROR 88
  - queue manager attribute 5
  - REMOTE\_Q\_NAME\_ERROR 89
  - UNKNOWN\_DEF\_XMIT\_Q 91
  - UNKNOWN\_REMOTE\_Q\_MGR 93
  - UNKNOWN\_XMIT\_Q 94
  - XMIT\_Q\_TYPE\_ERROR 95
  - XMIT\_Q\_USAGE\_ERROR 96
- .RESET MQ 37
  - refresh list of queues 18
- .RESET QM 37
- RESPONSE parameter
  - CMD 210

- RM
  - Power Line parameter 144
- Rule
  - description 47
  - not enabling non-MVS queue 239
- Rule Processor Detail Control Panel 65–66
- Rule Processor Detail Control panel 53–54, 60–61, 65–66, 70
- Rule Set Modifications Panel 62, 72
- Rule Set Overview Panel 61, 65
- Rule Set Overview panel 53, 65, 71
- Rules
  - creating
    - event messages and non-event messages 51
    - examples 52
    - for a Non-MQMFT\_EVENT Message 63
    - how to save 72
    - MQMFT\_EVENT
      - example 52
  - Rules and Formatting Header Variables 267
  - Rules and Formatting Header Variables –(Rules and Rule-invoked EXECs) 267, 294
  - Rules and Formatting Header Variables Version 2 268
  - Rules and Formatting Header Variables Version 2 – (Rules and Rule-invoked EXE 268, 295

## S

- sample coding 241
- security
  - AutoOPERATOR for MQSeries 151
- security considerations
  - defining SSIDs 19
  - defining user IDs 19
- Selection Criteria - MQS
  - field description
    - CorrelId 108
    - Event Type 107
    - Format 106
    - Manager(s) 106
    - Msg Buffer 109
    - Msgid 107
    - Queue ID 106
  - panel 54, 58, 66–67, 105
- Selection Criteria - MQS Panel 66
- service interval performance solution
  - enabling Rules 130
  - implementing 130
  - stopping 131
- Setting Up Non-OS/390 Queue Manager Connectivity 28
- Setting Up OS/390 Queue Manager Connectivity 28
- SHARED Variables 97
- skeleton tailoring
  - defined 309
- software requirements 10
- solution
  - basic intercommunication
    - implementing 140
    - stopping 141
    - using 137

- channel availability 134
  - implementing 135
  - stopping 136
- Dead Letter Queue 122
  - Dead Letter Management 123
  - implementing 122
  - not moving messages 233
  - stopping 125
- implementation considerations 121
- queue depth management
  - enabling Rules 132
  - implementing 132
  - stopping 133
- service interval performance
  - enabling Rules 130
  - implementing 130
  - stopping 131
- system queue archival
  - enabling Rules 127
  - implementing 126
  - stopping 129
- Start and Stop
  - event type 5
- statistics
  - tracking automation 114
- STEPLIB DD Card
  - adding libraries 17
- STRSTPEV
  - Q\_MGR\_ACTIVE 86
  - Q\_MGR\_NOT\_ACTIVE 86
  - queue manager attribute 5
- system queue archival 121
- system queue archival solution
  - enabling Rules 127
  - implementing 126
  - stopping 129

## T

- tools
  - diagnosing errors 220
- tracking automation statistics 114
- transmission queue 3
- TYPE parameter
  - CMD 209

## U

- UNKNOWN\_ALIAS\_BASE\_Q
  - IMFQ\_EVENT\_APPLNAME 90
  - IMFQ\_EVENT\_APPLTYPE 90
  - IMFQ\_EVENT\_BASEQNAME 90
  - IMFQ\_EVENT\_OBJECTQMGRNAME 90
  - IMFQ\_EVENT\_QMGRNAME 90
  - IMFQ\_EVENT\_QNAME 90
  - instrumentation event 90
- UNKNOWN\_DEF\_XMIT\_Q
  - IMFQ\_EVENT\_APPLNAME 91
  - IMFQ\_EVENT\_APPLTYPE 91

IMFQ\_EVENT\_OBJECTQMGRNAME 91  
 IMFQ\_EVENT\_QMGRNAME 91  
 IMFQ\_EVENT\_QNAME 91  
 IMFQ\_EVENT\_XMITQNAME 91  
 instrumentation event 91  
 UNKNOWN\_OBJECT\_NAME  
 IMFQ\_EVENT\_APPLNAME 92  
 IMFQ\_EVENT\_APPLTYPE 92  
 IMFQ\_EVENT\_OBJECTQMGRNAME 92  
 IMFQ\_EVENT\_PROCESSNAME 92  
 IMFQ\_EVENT\_QMGRNAME 92  
 IMFQ\_EVENT\_QNAME 92  
 instrumentation event 92  
 UNKNOWN\_REMOTE\_Q\_MGR  
 IMFQ\_EVENT\_APPLNAME 93  
 IMFQ\_EVENT\_APPLTYPE 93  
 IMFQ\_EVENT\_OBJECTQMGRNAME 93  
 IMFQ\_EVENT\_QMGRNAME 93  
 IMFQ\_EVENT\_QNAME 93  
 instrumentation event 93  
 UNKNOWN\_XMIT\_Q  
 IMFQ\_EVENT\_APPLNAME 94  
 IMFQ\_EVENT\_APPLTYPE 94  
 IMFQ\_EVENT\_OBJECTQMGRNAME 94  
 IMFQ\_EVENT\_QMGRNAME 94  
 IMFQ\_EVENT\_QNAME 94  
 IMFQ\_EVENT\_XMITQNAME 94  
 instrumentation event 94

## V

valid dead letter  
   Dead Letter Management 124  
 variable  
   APL  
     APLCC 146  
     APLLN1 147  
     APLLNxx 147  
     APLNOL 147  
     APLRC 146  
     IMFRC 146  
   AutoOPERATOR-supplied  
     IMFOJOB 98  
     IMFOQMGR 98  
     IMFQ\_MD\_ACCOUNTINGTOKEN 101  
     IMFQ\_MD\_APPLIDENTITYDATA 101  
     IMFQ\_MD\_APPLORIGINDATA 102  
     IMFQ\_MD\_BACKOUTCOUNT 101  
     IMFQ\_MD\_CODEDCHARSETID 100  
     IMFQ\_MD\_CORRELID 101  
     IMFQ\_MD\_ENCODING 100  
     IMFQ\_MD\_EXPIRY 99  
     IMFQ\_MD\_FEEDBACK 100  
     IMFQ\_MD\_FORMAT 101  
     IMFQ\_MD\_GROUPID 102  
     IMFQ\_MD\_MSGFLAGS 103  
     IMFQ\_MD\_MSGID 101  
     IMFQ\_MD\_MSGSEQUENCE 102  
     IMFQ\_MD\_MSGTYPE 99

IMFQ\_MD\_OFFSET 102  
 IMFQ\_MD\_ORIGINALLENGTH 103  
 IMFQ\_MD\_PERSISTENCE 101  
 IMFQ\_MD\_PRIORITY 101  
 IMFQ\_MD\_PUTAPPLNAME 102  
 IMFQ\_MD\_PUTAPPLTYPE 102  
 IMFQ\_MD\_PUTDATE 102  
 IMFQ\_MD\_PUTTIME 102  
 IMFQ\_MD\_REPLYTOQ 101  
 IMFQ\_MD\_REPLYTOQMGR 101  
 IMFQ\_MD\_REPORT 99  
 IMFQ\_MD\_STRUCID 98  
 IMFQ\_MD\_USERIDENTIFIER 101  
 IMFQ\_MD\_VERSION 99  
 IMFQ\_QAO\_CATCH\_UP\_MSG 98  
 IMFQ\_QATTR\_BACKOUTREQUEUEQNAME  
   103  
 IMFQ\_QATTR\_BACKOUTTHRESHOLD 103  
 IMFQ\_QATTR\_CREATIONDATE 103  
 IMFQ\_QATTR\_CREATIONTIME 103  
 IMFQ\_QATTR\_CURRENTQDEPTH 103  
 IMFQ\_QATTR\_DEFINITIONTYPE 103  
 IMFQ\_QATTR\_DEFINPUTOPENOPTION 103  
 IMFQ\_QATTR\_DEFPERSISTENCE 103  
 IMFQ\_QATTR\_DEFPRIORITY 103  
 IMFQ\_QATTR\_HARDENGETBACKOUT 103  
 IMFQ\_QATTR\_INHIBITGET 103  
 IMFQ\_QATTR\_INHIBITPUT 104  
 IMFQ\_QATTR\_INITIATIONQNAME 103  
 IMFQ\_QATTR\_MAXMSGLENGTH 104  
 IMFQ\_QATTR\_MAXQDEPTH 104  
 IMFQ\_QATTR\_MSGDELIVERYSEQUENCE 104  
 IMFQ\_QATTR\_OPENINPUTCOUNT 104  
 IMFQ\_QATTR\_OPENOUTPUTCOUNT 104  
 IMFQ\_QATTR\_PROCESSNAME 104  
 IMFQ\_QATTR\_QDESC 103  
 IMFQ\_QATTR\_RETENTIONINTERVAL 104  
 IMFQ\_QATTR\_SHAREABILITY 104  
 IMFQ\_QATTR\_STORAGECLASS 104  
 IMFQ\_QATTR\_TRIGGERCONTROL 104  
 IMFQ\_QATTR\_TRIGGERDAT 104  
 IMFQ\_QATTR\_TRIGGERDEPTH 104  
 IMFQ\_QATTR\_TRIGGERMSGPRIORITY 104  
 IMFQ\_QATTR\_TRIGGERTYPE 104  
 IMFQ\_QATTR\_USAGE 104  
 IMFQ\_STRUCTURES 98  
 IMFQCPF 98  
 IMFQMT 98  
 IMFMQI\_MD\_ACCOUNTINGTOKEN 204  
 MQSeries events 97  
 SHARED 97  
 TSO  
   IMFNOL 211  
   MQLN1 211  
   MQLN2 211  
   MQLNxxxx 211  
 Variable Dependencies panel 58, 68  
 variable  
   IMFMQI\_MD\_GROUPID 195

## Variables

- CICS Information Header
    - list of 250, 277
  - Dead Letter Header
    - list of 247, 274
  - General Purpose
    - list of 271
  - IMS Information Header
    - list of 254, 281
  - Message Descriptor
    - list of 245, 272
  - Message Descriptor Extension
    - list 288
    - list of 261
  - Options and Miscellaneous
    - list of 271
  - Reference Message Header
    - list of 264, 291
  - Transmission Queue Header
    - list of 256, 283
  - Trigger Message
    - list of 249, 276
  - Trigger Message 2 (Character format)
    - list of 266, 293
  - Work Information Header
    - list of 263, 290
- variables
- AutoOPERATOR-supplied
    - for USER messages 98
  - IMFEXEC MQI 154
  - IMFMQI\_MD\_ACCOUNTINGTOKEN 180, 193
  - IMFMQI\_MD\_APPLIDENTITYDATA 193, 204
  - IMFMQI\_MD\_APPLORIGINDATA 180, 195
  - IMFMQI\_MD\_BACKOUTCOUNT 179, 193, 203
  - IMFMQI\_MD\_CODEDCHARSETID 178, 191, 202
  - IMFMQI\_MD\_CORRELID 179, 203
  - IMFMQI\_MD\_ENCODING 178, 191, 202
  - IMFMQI\_MD\_EXPIRY 177, 202
  - IMFMQI\_MD\_FEEDBACK 178, 191, 202
  - IMFMQI\_MD\_FORMAT 179, 192, 203
  - IMFMQI\_MD\_GROUPID 181
  - IMFMQI\_MD\_MSGFLAGS 181, 195
  - IMFMQI\_MD\_MSGID 179, 192, 203
  - IMFMQI\_MD\_MSGSEQNUMBER 181, 195
  - IMFMQI\_MD\_MSGTYPE 177, 190, 201
  - IMFMQI\_MD\_OFFSET 181, 195
  - IMFMQI\_MD\_ORIGINALLENGTH 181, 195
  - IMFMQI\_MD\_PERSISTENCE 179, 192, 203
  - IMFMQI\_MD\_PRIORITY 179, 192, 203
  - IMFMQI\_MD\_PUTAPPLNAME 180, 194
  - IMFMQI\_MD\_PUTAPPLTYPE 180, 194
  - IMFMQI\_MD\_PUTDATE 180, 194
  - IMFMQI\_MD\_PUTTIME 180, 194
  - IMFMQI\_MD\_REPLYTOQ 179, 193, 203
  - IMFMQI\_MD\_REPLYTOQMGR 180, 193, 204
  - IMFMQI\_MD\_REPORT 177, 190, 201
  - IMFMQI\_MD\_STRUCID 176, 190, 201
  - IMFMQI\_MD\_USERIDENTIFIER 180, 193, 204
  - IMFMQI\_MD\_VERSION 176, 190, 201

- IMFMQI\_OD\_ALTERNATESECURITYID 186, 201
  - IMFMQI\_OD\_ALTERNATEUSERID 185, 200
  - IMFMQI\_OD\_DYNAMICQNAME 185, 200
  - IMFMQI\_OD\_INVALIDDESTCOUNT 185, 200
  - IMFMQI\_OD\_KNOWNDESTCOUNT 185, 200
  - IMFMQI\_OD\_OBJECTNAME 185, 200
  - IMFMQI\_OD\_OBJECTQMGRNAME 185, 200
  - IMFMQI\_OD\_OBJECTRECOFFSET 185, 200
  - IMFMQI\_OD\_OBJECTRECPT 185, 201
  - IMFMQI\_OD\_OBJECTTYPE 185, 200
  - IMFMQI\_OD\_RECSPRESENT 185, 200
  - IMFMQI\_OD\_RESOLVEDQMGRNAME 186, 201
  - IMFMQI\_OD\_RESOLVEDQNAME 186, 201
  - IMFMQI\_OD\_RESPONSERECOFFSET 185, 200
  - IMFMQI\_OD\_RESPONSERECPT 185, 201
  - IMFMQI\_OD\_STRUCID 185, 200
  - IMFMQI\_OD\_UNKNOWNDESTCOUNT 185, 200
  - IMFMQI\_OD\_VERSION 185, 200
    - list of 97
  - POPTS 188
- Variables for all MQSeries messages
- AutoOPERATOR-supplied 98
- variables
- IMFMQI\_MD\_APPLIDENTITYDATA 180
  - IMFMQI\_MD\_CORRELID 192
- viewing automation statistics 115

## W

- WAIT
  - Power Line parameter 145
- WAIT parameter
  - CMD 210
- warning
  - confirm rule set modification 62
- What Non-Event Messages Are 49
- Work Information Header Variables 263
- workstation panel 115, 118
  - fields 118
  - primary commands 116
  - queue management 120
    - field descriptions 120

## X

- XMIT\_Q\_TYPE\_ERROR
  - IMFQ\_EVENT\_APPLTYPE 95
  - IMFQ\_EVENT\_OBJECTQMGRNAME 95
  - IMFQ\_EVENT\_QMGRNAME 95
  - IMFQ\_EVENT\_QNAME 95
  - IMFQ\_EVENT\_QTYPE 95
  - IMFQ\_EVENT\_XMITQNAME 95
  - IMFQAPLN 95
  - instrumentation event 95

XMIT\_Q\_USAGE\_ERROR  
IMFQ\_EVENT\_APPLNAME 96  
IMFQ\_EVENT\_APPLTYPE 96  
IMFQ\_EVENT\_OBJECTQMGRNAME 96  
IMFQ\_EVENT\_QMGRNAME 96  
IMFQ\_EVENT\_QNAME 96  
IMFQ\_EVENT\_XMITQNAME 96  
instrumentation event 96

# STOP!

## IMPORTANT INFORMATION - DO NOT INSTALL THIS PRODUCT UNLESS YOU HAVE READ ALL OF THE FOLLOWING MATERIAL

By clicking the YES or ACCEPT button below (when applicable), or by installing and using this Product or by having it installed and used on your behalf, You are taking affirmative action to signify that You are entering into a legal agreement and are agreeing to be bound by its terms, EVEN WITHOUT YOUR SIGNATURE. BMC is willing to license this Product to You ONLY if You are willing to accept all of these terms. CAREFULLY READ THIS AGREEMENT. If You DO NOT AGREE with its terms, DO NOT install or use this Product; press the NO or REJECT button below (when applicable) or promptly contact BMC or your BMC reseller and your money will be refunded if by such time You have already purchased a full-use License.

### SOFTWARE LICENSE AGREEMENT FOR BMC PRODUCTS

**SCOPE.** This is a legally binding Software License Agreement ("**License**") between You (either an individual or an entity) and BMC pertaining to the original computer files (including all computer programs and data stored in such files) contained in the enclosed Media (as defined below) or made accessible to You for electronic delivery, if as a prerequisite to such accessibility You are required to indicate your acceptance of the terms of this License, and all whole or partial copies thereof, including modified copies and portions merged into other programs (collectively, the "**Software**"). "**Documentation**" means the related hard-copy or electronically reproducible technical documents furnished in association with the Software, "**Media**" means the original BMC-supplied physical materials (if any) containing the Software and/or Documentation, "**Product**" means collectively the Media, Software, and Documentation, and all Product updates subsequently provided to You, and "**You**" means the owner or lessee of the hardware on which the Software is installed and/or used. "**BMC**" means BMC Software Distribution, Inc. unless You are located in one of the following regions, in which case "BMC" refers to the following indicated BMC Software, Inc. subsidiary: (i) Europe, Middle East or Africa --BMC Software Distribution, B.V., (ii) Asia/Pacific -- BMC Software Asia Pacific Pte Ltd., (iii) Brazil -- BMC Software do Brazil, or (iv) Japan -- BMC Software K.K. If You enter into a separate, written software license agreement signed by both You and BMC or your authorized BMC reseller granting to you the rights to install and use this Product, then the terms of that separate, signed agreement will apply and this License is void.

**FULL-USE LICENSE.** Subject to these terms and payment of the applicable license fees, BMC grants You this non-exclusive License to install and use one copy of the Software for your internal use on the number(s) and type(s) of servers or workstations for which You have paid or agreed to pay to BMC or your BMC reseller the appropriate license fee. If your license fee entitles You only to a License having a limited term, then the duration of this License is limited to that term; otherwise this License is perpetual, subject to the termination provisions below.

**TRIAL LICENSE.** If You have not paid or agreed to pay to BMC or your BMC Reseller the appropriate license fees for a full use license, then, **NOTWITHSTANDING ANYTHING TO THE CONTRARY CONTAINED IN THIS LICENSE:** (i) this License consists of a non-exclusive evaluation license ("Trial License") to use the Product for a limited time ("Trial Period") only for evaluation; (ii) during the Trial Period, You may not use the Software for development, commercial, production, database management or other purposes than those expressly permitted in clause (i) immediately above; and (iii) your use of the Product is on an **AS IS** basis, and **BMC, ITS RESELLERS AND LICENSORS GRANT NO WARRANTIES OR CONDITIONS (INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE) TO YOU AND ACCEPT NO LIABILITY WHATSOEVER RESULTING FROM THE USE OF THIS PRODUCT UNDER THIS TRIAL LICENSE.** If You use this Product for other than evaluation purposes or wish to continue using it after the Trial Period, you must purchase a full-use license. When the Trial Period ends, your right to use this Product automatically expires, though in certain cases You may be able to extend the term of the Trial Period by request. Contact BMC or your BMC reseller for details.

**TERM AND TERMINATION.** This License takes effect on the first to occur of the date of shipment or accessibility to You for electronic delivery, as applicable (the "**Product Effective Date**"). You may terminate this License at any time for any reason by written notice to BMC or your BMC reseller. This License and your right to use the Product will terminate automatically with or without notice by BMC if You fail to comply with any material term of this License. Upon termination, You must erase or destroy all components of the Product including all copies of the Software, and stop using or accessing the Software. Provisions concerning Title and Copyright, Restrictions (or Restricted Rights, if You are a U.S. Government entity) or limiting BMC's liability or responsibility shall survive any such termination.

**TITLE AND COPYRIGHT; RESTRICTIONS.** All title and copyrights in and to the Product, including but not limited to all modifications thereto, are owned by BMC and/or its affiliates and licensors, and are protected by both United States copyright law and applicable international copyright treaties. You will not claim or assert title to or ownership of the Product. To the extent expressly permitted by applicable law or treaty notwithstanding this limitation, You may copy the Software only for backup or archival purposes, or as an essential step in utilizing the Software, but for no other purpose. You will not remove or alter any copyright or proprietary notice from copies of the Product. You acknowledge that the Product contains valuable trade secrets of BMC and/or its affiliates and licensors. Except in accordance with the terms of this License, You agree (a) not to decompile, disassemble, reverse engineer or otherwise attempt to derive the Software's source code from object code except to the extent expressly permitted by applicable law or treaty despite this limitation; (b) not to sell, rent, lease, license, sublicense, display, modify, time share, outsource or otherwise transfer the Product to, or permit the use of this Product by, any third party; and (c) to use reasonable care and protection to prevent the unauthorized use, copying, publication or dissemination of the Product and BMC confidential information learned from your use of the Product. **You will not export or re-export any Product without both the written consent of BMC and the appropriate U.S. and/or foreign government license(s) or license exception(s).** Any programs, utilities, modules or other software or documentation created, developed, modified or enhanced by or for You using this Product shall likewise be subject to these restrictions. BMC has the right to obtain injunctive relief against any actual or threatened violation of these restrictions, in addition to any other available remedies. Additional restrictions may apply to certain files, programs or data supplied by third parties and embedded in the Product; consult the Product installation instructions or Release Notes for details.

**LIMITED WARRANTY AND CONDITION.** If You have purchased a Full-Use License, BMC warrants that (i) the Media will be, under normal use, free from physical defects, and (ii) for a period of ninety (90) days from the Product Effective Date, the Product will perform in substantial accordance with the operating specifications contained in the Documentation that is most current at the Product Effective Date. BMC's entire liability and your exclusive remedy under this provision will be for BMC to use reasonable best efforts to remedy defects covered by this warranty

and condition within a reasonable period of time or, at BMC's option, either to replace the defective Product or to refund the amount paid by You to license the use of the Product. BMC and its suppliers do not warrant that the Product will satisfy your requirements, that the operation of the Product will be uninterrupted or error free, or that all software defects can be corrected. This warranty and condition shall not apply if: (i) the Product is not used in accordance with BMC's instructions, (ii) a Product defect has been caused by any of your or a third party's malfunctioning equipment, (iii) any other cause within your control causes the Product to malfunction, or (iv) You have made modifications to the Product not expressly authorized in writing by BMC. No employee, agent or representative of BMC has authority to bind BMC to any oral representations, warranties or conditions concerning the Product. **THIS WARRANTY AND CONDITION IS IN LIEU OF ALL OTHER WARRANTIES AND CONDITIONS. THERE ARE NO OTHER EXPRESS OR IMPLIED WARRANTIES OR CONDITIONS, INCLUDING THOSE OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, REGARDING THIS LICENSE OR ANY PRODUCT LICENSED HEREUNDER. THIS PARAGRAPH SHALL NOT APPLY TO A TRIAL LICENSE.** Additional support and maintenance may be available for an additional charge; contact BMC or your BMC reseller for details.

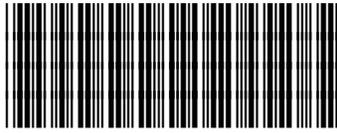
**LIMITATION OF LIABILITY.** Except as stated in the next succeeding paragraph, BMC's and your BMC reseller's total liability for all damages in connection with this License is limited to the price paid for the License. **IN NO EVENT SHALL BMC BE LIABLE FOR ANY CONSEQUENTIAL, SPECIAL, INCIDENTAL, PUNITIVE OR INDIRECT DAMAGES OF ANY KIND ARISING OUT OF THE USE OF THIS PRODUCT (SUCH AS LOSS OF PROFITS, GOODWILL, BUSINESS, DATA OR COMPUTER TIME, OR THE COSTS OF RECREATING LOST DATA), EVEN IF BMC HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.** Some jurisdictions do not permit the limitation of consequential damages so the above limitation may not apply.

**INDEMNIFICATION FOR INFRINGEMENT.** BMC will defend or settle, at its own expense, any claim against You by a third party asserting that your use of the Product within the scope of this License violates such third party's patent, copyright, trademark, trade secret or other proprietary rights, and will indemnify You against any damages finally awarded against You arising out of such claim. However, You must promptly notify BMC in writing after first receiving notice of any such claim, and BMC will have sole control of the defense of any action and all negotiations for its settlement or compromise, with your reasonable assistance. BMC will not be liable for any costs or expenditures incurred by You without BMC's prior written consent. If an order is obtained against your use of the Product by reason of any claimed infringement, or if in BMC's opinion the Product is likely to become the subject of such a claim, BMC will at its option and expense either (i) procure for You the right to continue using the product, or (ii) modify or replace the Product with a compatible, functionally equivalent, non-infringing Product, or (iii) if neither (i) nor (ii) is practicable, issue to You a pro-rata refund of your paid license fee(s) proportionate to the number of months remaining in the 36 month period following the Product Effective Date. This paragraph sets forth your only remedies and the total liability to You of BMC, its resellers and licensors arising out of such claims.

**GENERAL.** This License is the entire understanding between You and BMC concerning this License and may be modified only in a mutually signed writing between You and BMC. If any part of it is invalid or unenforceable, that part will be construed, limited, modified, or severed so as to eliminate its invalidity or unenforceability. This License will be governed by and interpreted under the laws of the jurisdiction named below, without regard to conflicts of law principles, depending on which BMC Software, Inc. subsidiary is the party to this License: (i) BMC Software Distribution, Inc. - the State of Texas, U.S.A., (ii) BMC Software Distribution, B.V. - The Netherlands, (iii) BMC Software Asia Pacific Pte Ltd. -- Singapore (iv) BMC Software do Brazil -- Brazil, or (v) BMC Software K.K. -- Japan. Any person who accepts or signs changes to the terms of this License promises that they have read and understood these terms, that they have the authority to accept on your behalf and legally obligate You to this License. Under local law and treaties, the restrictions and limitations of this License may not apply to You; You may have other rights and remedies, and be subject to other restrictions and limitations.

**U.S. GOVERNMENT RESTRICTED RIGHTS.** UNPUBLISHED -- RIGHTS RESERVED UNDER THE COPYRIGHT LAWS OF THE UNITED STATES. Use, duplication, or disclosure by the U.S. Government is subject to restrictions set forth in FAR Section 52.227-14 Alt. III (g)(3), FAR Section 52.227-19, DFARS 252.227-7014 (b) or DFARS 227.7202, as amended from time to time. Contractor/Manufacturer is BMC Software, Inc., 2101 CityWest Blvd., Houston, TX 77042-2827, USA. Any contract notices should be sent to this address.

# Notes



\*100042383\*