



IBM MQSeries Workflow for OS/390

Customization and Administration

Version 3 Release 2



IBM MQSeries Workflow for OS/390

Customization and Administration

Version 3 Release 2

Note!

Before using this information and the product it supports, be sure to read the general information under "Notices" on page xiii.

Second Edition (December 1999)

This edition applies to Version 3, Release 2, Modification 1 of IBM MQSeries Workflow for OS/390 (product number 5655-A96) and to all subsequent releases and modifications until otherwise indicated in new editions.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

IBM welcomes your comments. A form for your comments appears at the back of this publication. If the form has been removed, address your comments to:

IBM Deutschland Entwicklung GmbH
Department 3248
Schoenaicher Strasse 220
D-71032 Boeblingen
Federal Republic of Germany

FAX (Germany): 07031+16-3456
FAX (Other Countries): (+49)+7031-16-3456

IBM Mail Exchange: DEIBMBM9 at IBMMAIL
Internet: s390id@de.ibm.com

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 1998, 1999. All rights reserved.

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

Contents

Figures	vii
--------------------------	------------

Tables	ix
-------------------------	-----------

About this book	xi
----------------------------------	-----------

Who should read this book	xi
How this book is organized	xi
How to get additional information	xiii
How to send your comments	xiii
Notices	xiii
Trademarks	xv
How to read the syntax diagrams	xv

Summary of changes	xix
-------------------------------------	------------

Part 1. Customization 1

Chapter 1. Understanding MQSeries Workflow for OS/390 3

Understanding the MQSeries Workflow for OS/390 architecture	3
Understanding the customization process	4
Customization scenario pre-requirements	5

Chapter 2. Planning your configuration 7

Deciding your MQSeries Workflow for OS/390 identifiers	7
Overview of the customization parameters	7
Installation scope identifiers	9
System group scope identifiers	10
System scope identifiers	11
Flags and high level qualifiers	13
Subsystem identifiers	14
Customization identifiers	14
Evaluate database requirements	15
More detailed database planning (optional)	16

Chapter 3. Before starting customization. 19

Post-installation tasks	19
Create the MMS message catalogs	20
LPA library concatenation	21

Chapter 4. Creating a system group and a primary system 23

Pre-customization	23
Data set allocation	23
Create input files for customization	24
System customization	24
General DB2 customization	25
Workflow DB2 customization	25
Populate the Workflow database	26

Program execution server directory DB2 customization	27
Populate the PES directory database	28
Program execution server mapping DB2 customization	28
MQSeries customization	29
OS/390 trace customization	30
CICS API support customization	30
IMS API support customization	32
Workflow server customization	32
LAN client customization	33
Customize Java-API support	34
System customization verification	36
Verify Workflow client sample application	37
Program execution customization	39
Customize CICS EXCI invocation	39
Customize MQSeries CICS bridge invocation	41
Customize IMS CPIC invocation	44
Customize MQSeries IMS bridge invocation	46
Customize program execution server directory	47
Configure program execution samples	49
Verify program execution samples	50

Chapter 5. Creating additional systems in an existing system group 53

Decide the new system's identifiers	53
Data set allocation	55
Create input files for customizing an additional system in a system group	56
General DB2 customization (DB2 data sharing)	56
Update topology setting in the Workflow database	57
MQSeries customization	58
OS/390 trace customization	58
CICS API support customization	59
IMS API support customization	61
Workflow server customization	61
LAN client customization	62
Customize the MQSeries client connection	62
Customize the MQSeries Workflow client	63

Chapter 6. Adding extra Workflow clients to an existing system 65

Basic client customization	65
Decide the new client's identifiers	65
Data set allocation for client	66
Create input files for customizing a new Client	66
MQSeries customization for a new client	67
Generate MQSeries channel tab file for LAN client	67
CICS API support customization for new client	68
IMS API support customization for new client	70
LAN Client Customization	70
Client request concentrator customization	71

Part 2. System administration . . . 73

Chapter 7. Introduction to system administration . . . 75

Objects you will need to administer or use	75
Administration in an MQSeries Workflow system	78
System administration client/server components	78
The administration server	79
The administration console	80
Overview of administration tasks	80
System and server administration tasks	80
Program and user administration tasks	81

Chapter 8. Administration console tasks . . . 83

Administration server commands	83
Starting the administration server and administration console.	83
Stopping the administration server	84
System commands	84
Starting the system	84
Stopping the system	85
Restarting the system	85
Displaying all server instances in the system	86
Server commands	86
Starting servers	86
Stopping servers.	87
Restarting servers	88
Displaying the number of instances of a server	89
Hold queue commands	89
Displaying number of messages in the hold queue	90
Displaying messages in the hold queue	90
Replaying messages from the hold queue	90
Deleting messages from the hold queue	91

Chapter 9. Buildtime administration tasks . . . 93

Defining process models	93
Defining server properties	93
Defining program properties	96
Defining the connection between a program activity and the PES	100
Uploading process models to the host	101
Importing and exporting process models	101
Using the FDL import/export tool	101

Chapter 10. Program execution. . . 103

Administering the Program Execution Server directory	105
Adding a new service definition and the related user resolution information.	106
Adding a user-defined invocation type	107
Adding a user-defined mapping type	107
Importing the PES directory	107
Caching the PES directory at runtime	108
Administering programs	108
Enabling an OS/390 program to be run as a program activity	108

Enabling an OS/390 program to run as a safe application	109
Disabling a program	109
Authorizing a user to access an OS/390 program	109
Revoking a user's access to OS/390 programs	110
Administering program mapping.	110
Importing a program mapping definition	110
Enabling a program's mapping	112
Disabling a program's mapping	113
Deleting a program mapping definition	113
Enabling a mapping type	113
Disabling a mapping type	114
Administering invocation types	114
Enabling an invocation type	114
Disabling an invocation type	114
Program execution security.	114
Information in the PES directory that is relevant to security	116
Program security	116

Chapter 11. Administering Servlets on the WebSphere Application Server . . 117

Placing servlet class files on the Application Server	117
If necessary, create a new servlet sub-directory	117
Monitoring your servlet, or setting servlet initialization parameters.	118
Placing the HTML files on the Application Server	118
Running a sample servlet, to log on MQSeries Workflow.	118

Chapter 12. Performance tuning . . . 121

Changing the number of running server instances	121
Changing the number of server instances per address space	121
Caching the PES directory	122

Chapter 13. Problem determination 123

Server problems	123
Message catalog not available	123
Servers cannot be started	123
The administration server cannot be started	124
The administration server does not respond to console commands	124
The program execution server cannot be started	124
Server instance terminates unexpectedly without reporting an error	125
One or more program execution server instances terminate, request goes in state error (FMC_ERROR_RETRY_LIMIT_REACHED).	125
Program execution request stays in state running	125
A dump is written before all server instances are started	125
Cannot stop servers	126
Changes made to the machine profile are not activated	126
Changes made to the PES directory are not activated	127

Changes made to the program mapping definition are not activated	127
Hold queue problems (undelivered messages)	127
Resource and performance problems	127
User response times are unacceptably long	127
Invalid password	128
Running out of spool space.	128
The MQSeries Workflow for OS/390 system trace facility.	129
Turning tracing on.	129
Restarting the component and reproducing the problem	130
Viewing the trace	130
Turning tracing off	131
Tracing in CICS	131
What do I do if I get an SVC dump?	131
Create a problem summary.	131

Part 3. Using OS/390 Workload Manager with Workflow 133

Chapter 14. Introduction to WLM 135

What is OS/390 Workload Manager?	135
Overview of WLM	135
WLM queuing model.	135
Service definition	137
MQSeries Workflow and OS/390 Workload Management	138
Workflow administration server	139
OS/390 Workload Manager application environments	141
MQSeries for OS/390 workload management	143
Classification	144

Chapter 15. Setting up WLM for MQSeries Workflow for OS/390. 145

Creating a WLM service definition	145
Service definition	145
Service policy	146
Workload.	147
Service class.	147
Classification rule	149
Application environment	150
Installing and activating a WLM service policy in a Parallel Sysplex environment	152
WLM administration	153
Switching servers between WLM and non-WLM mode by importing an FDL file	154
Starting WLM-managed servers when WLM is in manual mode	155

Chapter 16. WLM problem determination 157

WLM setup problems	157
Unexpected runtime behavior of MQSeries Workflow with WLM.	157

Part 4. Appendixes 159

Appendix A. Program Execution

Server directory	161
PES directory structure	161
Invocation section	162
Mapping section	162
Security section.	162
PES directory template	163
PES directory dependencies on the process model's OS/390 program definitions	164

Appendix B. The PES directory import tool's syntax and semantics 167

Return codes	167
PES directory import examples	167
Importing a PES directory source file	167
Importing a PES directory and writing a log file	168
Deleting the PES directory	168

Appendix C. Program mapping import tool syntax 169

Creating a new program mapping definition	169
Replacing an existing program mapping definition	169
Inserting a program mapping definition	170
Deleting a program mapping definition	170
Listing program mapping definitions	170
Control statement execution	170
Example control statements.	171

Appendix D. Naming and code page restrictions 173

Naming Buildtime objects	173
Restrictions for passwords in CICS	173

Appendix E. FDL code page conversion tool 175

Using the FDL code page conversion tool	175
Options	175
Return codes	176

Appendix F. FDL import/export tool 177

FDL import/export tool's syntax	177
Options for the import / export tool	179
Log file and errors.	179
Return codes	180
Examples.	180
FDL import examples	180
FDL export examples.	181
Translate examples	182

Appendix G. Customization parameter files 183

Customization parameter file for a primary system	183
Customization parameter file for adding a system to a system group	186
Customization parameter file for a client on a queue manager.	190

Appendix H. Machine profiles 193

Server machine profile	193
Client machine profile	195

Appendix I. Environment variable files 197

Server environment variable file	197
Client environment file	197

Appendix J. WLM message classification 199

Message classification namespace.	199
Program Execution Server invocation information	199
Static Workflow message classification	200
Process Template messages	201
Process Template List messages	201
Process messages	201
Process InstList messages	201
Work Item messages	202
Activity messages	202
User Information messages	202
Process Monitor messages	203
WorkList messages	203
PEA/PES Server messages	203
PEA/PES Reply messages	203

Scheduling messages	203
SubProcess messages	204
Internal Server messages	204

Appendix K. Nesting WLM classification information 205

Glossary 207

Bibliography 211

MQSeries Workflow for OS/390 publications	211
MQSeries Workflow publications	211
MQSeries publications	211
Workflow publications	211
Other useful publications	211
Licensed books	211

Index 213

Readers' Comments — We'd Like to Hear from You 219

Figures

1. MQSeries Workflow for OS/390 architecture	3	15. The WLM queuing model used by MQSeries	
2. MQSeries Workflow for OS/390 customization tasks	5	Workflow	143
3. Customization parameters for a Workflow system	8	16. WLM panel: Definition menu	146
4. Customization parameters for DB2	9	17. WLM panel: Create a service policy	146
5. Implementation of the administration component in an MQSeries Workflow system	79	18. WLM panel: Create a workload	147
6. Execution server properties: General page	95	19. WLM panel: Create a service class	148
7. Program properties: Data page	97	20. WLM menu: Choose a goal type	148
8. Program properties: OS/390 page	98	21. WLM panel: Subsystem type selection list for rules	149
9. Program activity properties: Execution page	100	22. WLM panel: Modify rules for the subsystem type.	149
10. OS/390 Program execution server: component structure	104	23. WLM panel: Create an application environment	151
11. Program mapping definition process and components	111	24. WLM menu: Install a service definition	152
12. Server address spaces	136	25. WLM menu: Activate a service definition	153
13. Interactions between MQSeries and WLM to manage execution server address spaces	136	26. WLM panel: Policy selection list	153
14. WLM service policy: response time goals	139	27. FDL settings to switch to WLM mode	154
		28. FDL settings to switch to non-WLM mode	155
		29. Example of nesting WLM classification information	205

Tables

1. Installation scope identifiers	9	48. Customize the MQSeries Workflow client	63
2. System group identifiers	10	49. Identifiers required for each new client	65
3. System scope identifiers	11	50. Data set allocation for client	66
4. Flags and high level qualifiers	13	51. Create input files for client customization	67
5. Subsystem identifiers	14	52. MQSeries customization for new client	67
6. Customization identifiers	14	53. Generate MQSeries Channel tab file for use on LAN Client	68
7. Sample scenario characteristics suitable for the suggested database allocations	15	54. CICS API support customization for new client	68
8. Files that define the databases	16	55. IMS customization	70
9. Suggested buffer pool sizes and allocation	17	56. Enabling a client to use the Workflow API	70
10. Create MMS message catalogs	20	57. Enabling a client to act as a client request concentrator	71
11. Copy LPALIB member	21	58. System and server administration tasks	80
12. Data set allocation	23	59. Program and user administration tasks: tool dependencies	81
13. Create input files for customization	24	60. Server types	86
14. General DB2 customization	25	61. Server properties that can be changed.	93
15. Workflow DB2 customization	25	62. Server properties that should not be changed	94
16. Program execution directory DB2 customization	27	63. Server properties that are ignored on OS/390	94
17. Populate the PES directory	28	64. Program properties: OS/390 page settings	98
18. Program execution server mapping DB2 customization	28	65. Program mapping parser and import tool's return codes	112
19. MQSeries customization	29	66. Meaningful security setting combinations in Buildtime	115
20. OS/390 trace customization	30	67. Extended trace format converter return codes	130
21. CICS API support customization	31	68. Problems importing FDL for WLM	157
22. IMS API support customization	32	69. Problems importing FDL for WLM	158
23. Workflow server customization	32	70. PES directory import tool's options	167
24. Customize the MQSeries client connection	33	71. PES directory import tool's return codes	167
25. Customize the MQSeries Workflow client	34	72. FDL code page conversion tool's return codes	176
26. Customize the MQSeries Workflow client	35	73. FDL import/export tool's return codes	180
27. System customization verification	37	74. Server machine profile settings.	193
28. Verify Workflow client sample application	37	75. Client machine profile settings.	195
29. Customizing program execution invocation types.	39	76. Server environment variable file settings	197
30. Customize CICS EXCI invocation	40	77. Client environment variable file settings	198
31. Customize MQSeries CICS bridge invocation	41	78. MQWIH_ServiceStep field definition	199
32. Customize IMS CPIC invocation	44	79. MQWIH_ServiceName field definition	199
33. Customize MQSeries IMS bridge invocation	46	80. Process Template messages	201
34. Customize program execution server directory	48	81. Process Template List messages	201
35. Configure program execution samples	49	82. Process messages	201
36. Verify program execution samples	50	83. Process InstList messages	201
37. Identifiers required for each new system	53	84. Work Item messages	202
38. Data set allocation	55	85. Activity messages	202
39. Create input files for customization	56	86. User Information messages	202
40. General DB2 customization (DB2 data sharing)	57	87. Process Monitor messages	203
41. Updating topology setting in the Workflow database	57	88. WorkList messages.	203
42. MQSeries customization	58	89. PEA-Server messages	203
43. OS/390 trace customization	58	90. PEA Reply messages	203
44. CICS API support customization	59	91. Scheduling messages	203
45. IMS API support customization	61	92. SubProcess messages	204
46. Workflow server customization	61	93. Internal Server messages.	204
47. Customize the MQSeries client connection	62		

About this book

This book provides information about customization and administration functions and practises within an IBM MQSeries Workflow for OS/390 system. It explains the basic concepts of system administration and describes how to use the MQSeries Workflow administration console to administer and oversee an MQSeries Workflow for OS/390 system or system group. For information about administration of MQSeries Workflow on operating systems other than OS/390, see *IBM MQSeries Workflow: Administration Guide* .

It is assumed that you have read the *IBM MQSeries Workflow: Concepts and Architecture* book and are familiar with the MQSeries Workflow system structure. You should also understand how MQSeries Workflow uses DB2 to store domain, system group, and system properties.

Who should read this book

This book is intended for a system administrator who is the first person defined in an MQSeries Workflow system. A system administrator does the following:

- Installs and customizes MQSeries Workflow for OS/390 and its prerequisite and corequisite products.
- Administrates MQSeries Workflow for OS/390 databases and the day-to-day operation of MQSeries Workflow for OS/390.

This book does not describe installation of MQSeries Workflow products. It assumes that your MQSeries Workflow for OS/390 system has already been set up as described in the *MQSeries Workflow for OS/390: Program Directory*.

How this book is organized

- “Part 1. Customization” on page 1 describes how to customize MQSeries Workflow for OS/390. It contains the following chapters:
 - “Chapter 1. Understanding MQSeries Workflow for OS/390” on page 3 describes the MQSeries Workflow for OS/390 architecture and the customization tasks.
 - “Chapter 2. Planning your configuration” on page 7 provides tables to photocopy and complete for use during customization.
 - “Chapter 3. Before starting customization” on page 19 describes post-installation tasks that must be performed once per installation.
 - “Chapter 4. Creating a system group and a primary system” on page 23 guides you through the process necessary to create a functional MQSeries Workflow for OS/390 system.
 - “Chapter 5. Creating additional systems in an existing system group” on page 53 describes how to add additional Workflow systems to an existing system group.
 - “Chapter 6. Adding extra Workflow clients to an existing system” on page 65 describes how to add various types of client to an existing Workflow system.
- “Part 2. System administration” on page 73 introduces the concepts and components of system administration in an MQSeries Workflow system and

explains how to start and use the MQSeries Workflow for OS/390 administration console. Details regarding problem determination and using the trace facility are also given.

- “Chapter 7. Introduction to system administration” on page 75 gives an overview of the objects and tasks involved in administrating this product.
- “Chapter 8. Administration console tasks” on page 83 describes the command-driven administration interface that is used to start and stop the system and servers.
- “Chapter 9. Buildtime administration tasks” on page 93 describes the administration tasks connected with the Buildtime tool, and process models.
- “Chapter 10. Program execution” on page 103 covers all tasks relating to the program execution server, such as administering programs, users, mappings, and invocation types.
- “Chapter 11. Administering Servlets on the WebSphere Application Server” on page 117 describes how to add a Java Servlet to the WebSphere Application Server for OS/390 that will call the Java-API of MQSeries Workflow. This allows users to invoke MQSeries Workflow actions from their Internet Web browser.
- “Chapter 12. Performance tuning” on page 121 describes some specific ways to improve system performance.
- “Chapter 13. Problem determination” on page 123 describes solutions to specific problems, and describes how to use the tracing facilities.
- “Part 3. Using OS/390 Workload Manager with Workflow” on page 133 describes how to apply OS/390 Workload Management (WLM) system with MQSeries Workflow for OS/390 services.
 - “Chapter 14. Introduction to WLM” on page 135 introduces workload management concepts.
 - “Chapter 15. Setting up WLM for MQSeries Workflow for OS/390” on page 145 describes how to create your service definitions, and how to migrate existing MQSeries Workflow for OS/390 system groups to enable WLM support.
 - “Chapter 16. WLM problem determination” on page 157 provides help solving problems getting WLM working with MQSeries Workflow for OS/390.
- “Part 4” contains the following appendixes:
 - “Appendix A. Program Execution Server directory” on page 161 describes the structure of the program execution server’s configuration directory and its dependencies on values in the OS/390 program definitions made in the process model using MQSeries Workflow Buildtime.
 - “Appendix B. The PES directory import tool’s syntax and semantics” on page 167 provides the syntax, options, and examples of using the tool for importing the program execution server’s directory.
 - “Appendix C. Program mapping import tool syntax” on page 169 describes the database utility language used to modify the program mapper’s database.
 - “Appendix D. Naming and code page restrictions” on page 173 describes the restrictions for naming OS/390 objects in the MQSeries Workflow Buildtime process model.
 - “Appendix E. FDL code page conversion tool” on page 175 contains details about a tool to convert process model information between different code pages. You may need to use this tool if your uploading method corrupts your FDL files.

- “Appendix F. FDL import/export tool” on page 177 provides the syntax, options, and examples of using the tool for importing and exporting process model information in the FDL file format.
- “Appendix G. Customization parameter files” on page 183 describes the template files used during customization.
- “Appendix H. Machine profiles” on page 193 describes contents of the machine profile files that determines the behavior of servers, tools, and clients.
- “Appendix I. Environment variable files” on page 197 describes contents of the environment variable files for servers and clients.
- “Appendix J. WLM message classification” on page 199 describes the classification information for all Workflow messages sent to the execution and program execution servers.
- “Appendix K. Nesting WLM classification information” on page 205 describes how to specify qualifier names in WLM that are more than 8 characters long.
- At the back of the book there is a glossary that defines terms as they are used in this book, a bibliography, and an index.

How to get additional information

Visit the MQSeries Workflow home page at
<http://www.software.ibm.com/ts/mqseries/workflow>

For a list of additional MQSeries Workflow publications, refer to “MQSeries Workflow publications” on page 211.

How to send your comments

Your feedback is important in helping to provide the most accurate and high-quality information. If you have any comments about this book or any other MQSeries Workflow for OS/390 documentation, choose one of the following methods:

- Send your comments by e-mail to: s390id@de.ibm.com. Be sure to include the name of the book, the part number of the book, the version of MQSeries Workflow for OS/390, and, if applicable, the specific location of the text you are commenting on (for example, a page number or table number).
- Fill out one of the forms at the back of this book and return it by mail, by fax, or by giving it to an IBM representative.

Notices

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

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

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
522 South Road
Poughkeepsie New York 12601-5400
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

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

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

IBM accepts no responsibility for the content or use of non-IBM web sites mentioned in this publication or accessed through an IBM web site that is mentioned in this publication.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

AIX	DB2 Universal Database	MVS/VSA
CICS	IBM	OS/390
CICS/ESA	IMS/ESA	VTAM
DB2	MQSeries	

Microsoft, Windows, Windows NT and the Windows logo are registered trademarks of Microsoft Corporation.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

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

How to read the syntax diagrams

In this manual diagrams are used to illustrate programming syntax. To use a diagram, follow a path from left to right, top to bottom, adding elements as you go. In these diagrams, all spaces and other characters are significant.

Each diagram begins with a double right arrowhead and ends with a right and left arrowhead pair.

The following rules apply to the syntax diagrams used in this book:

- The \blacktriangleright — symbol indicates the beginning of a statement.
The — \blacktriangleright symbol indicates that the statement syntax is continued on the next line.
The \blacktriangleright — symbol indicates that a statement is continued from the previous line.
The — \blacktriangleleft symbol indicates the end of a statement.
Diagrams of syntactical units other than complete statements start with the \blacktriangleright — symbol and end with the — \blacktriangleright symbol.
- Required items appear on the horizontal line (the main path).

\blacktriangleright —*required_item*— \blacktriangleleft

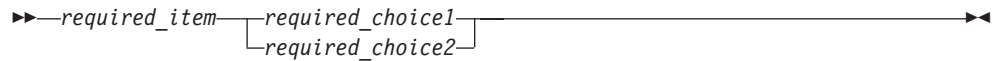
- Optional items normally appear below the main path.

\blacktriangleright —*required_item*— \blacktriangleleft
 └─*optional_item*─┘

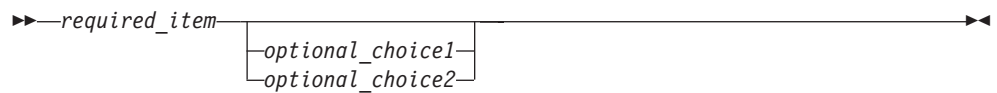
If an optional item appears above the main path, that item has no effect on the execution of the statement and is used only for readability.



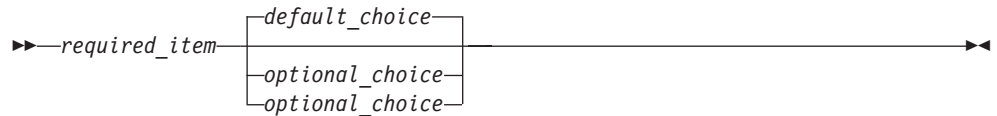
- If you can choose from two or more items, they appear vertically, in a stack. If you *must* choose one of the items, one item of the stack appears on the main path.



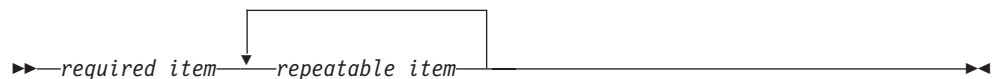
If choosing one of the items is optional, the entire stack appears below the main path.



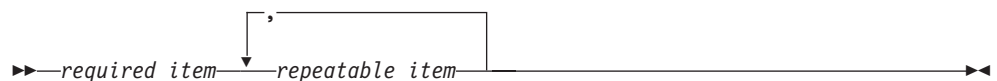
If one of the items is the default, it appears above the main path and the remaining choices are shown below.



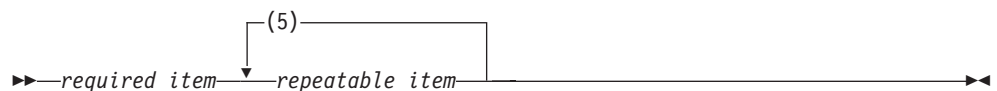
- An arrow returning to the left, above the main line, indicates an item that can be repeated.



If the repeat arrow contains a comma, you must separate repeated items with a comma.



If the repeat arrow contains a number in brackets, the number represents the maximum number of times that item can appear.



A repeat arrow above a stack indicates that you can repeat the items in the stack.

- Keywords appear in uppercase (for example, FROM). Variables appear in all lowercase letters (for example, *column name*). They represent user-supplied names or values.
- If punctuation marks, parentheses, arithmetic operators, or other such symbols are shown, you must enter them as part of the syntax.
- Syntax diagrams may be broken into fragments. A fragment is indicated by vertical bars with the name of the fragment between the bars. The fragment is shown following the main diagram, like so:

|—| A Fragment |—————|

A Fragment:

|—Keyword—*value*—————|

Summary of changes

This book contains information previously presented in *MQSeries Workflow for OS/390: Customization and Administration*, SC33-7030-01. It includes terminology, maintenance, and editorial changes to support MQSeries Workflow for OS/390 Version 3 Release 2.1. The following summarizes the most significant new information and functionality in this release:

- “Chapter 1. Understanding MQSeries Workflow for OS/390” on page 3 describes the architecture and customization tasks.
- “Chapter 2. Planning your configuration” on page 7 describes how MQSeries clusters are used, and includes new identifiers that are required for customization.
- “Chapter 5. Creating additional systems in an existing system group” on page 53 describes how to add additional Workflow systems to an existing system group.
- “Chapter 6. Adding extra Workflow clients to an existing system” on page 65 describes how to add various types of client to an existing Workflow system.
- “Starting the administration server and administration console” on page 83 includes the option to register the administration server with the OS/390 Automatic Restart Manager (ARM).
- “Hold queue commands” on page 89 describes how to display, replay, and delete undelivered messages intended for the execution server.
- “Caching the PES directory at runtime” on page 108 describes how to activate and refresh the PES directory cache.
- “Chapter 11. Administering Servlets on the WebSphere Application Server” on page 117 describes how to enable Java servlets that allow users to invoke calls to the MQSeries Workflow Java API from an Internet Web browser.
- “Part 3. Using OS/390 Workload Manager with Workflow” on page 133 describes how to set up the execution and program execution servers to be managed by the OS/390 Workload manager (WLM) .

Part 1. Customization

Chapter 1. Understanding MQSeries Workflow for OS/390	3
Understanding the MQSeries Workflow for OS/390 architecture	3
Understanding the customization process	4
Customization scenario pre-requirements	5
Scenario 1: Minimum configuration (one system group containing one system)	5
Scenario 2: Multiple queue managers (additional systems or clients)	6
Chapter 2. Planning your configuration	7
Deciding your MQSeries Workflow for OS/390 identifiers	7
Overview of the customization parameters	7
Installation scope identifiers	9
System group scope identifiers	10
System scope identifiers	11
Flags and high level qualifiers	13
Subsystem identifiers	14
Customization identifiers	14
Evaluate database requirements	15
More detailed database planning (optional)	16
Chapter 3. Before starting customization	19
Post-installation tasks	19
Create the MMS message catalogs	20
LPA library concatenation	21
Chapter 4. Creating a system group and a primary system	23
Pre-customization	23
Data set allocation	23
Create input files for customization	24
System customization	24
General DB2 customization	25
Workflow DB2 customization	25
Populate the Workflow database	26
Program execution server directory DB2 customization	27
Populate the PES directory database	28
Program execution server mapping DB2 customization	28
MQSeries customization	29
OS/390 trace customization	30
CICS API support customization	30
IMS API support customization	32
Workflow server customization	32
LAN client customization	33
Customize the MQSeries client connection	33
Customize the MQSeries Workflow client	34
Customize Java-API support	34
System customization verification	36
Verify Workflow client sample application	37
Program execution customization	39
Customize CICS EXCI invocation	39
Customize MQSeries CICS bridge invocation	41
Customize IMS CPIC invocation	44
Customize MQSeries IMS bridge invocation	46
Customize program execution server directory	47
Configure program execution samples	49
Verify program execution samples	50
Chapter 5. Creating additional systems in an existing system group	53
Decide the new system's identifiers	53
Data set allocation	55
Create input files for customizing an additional system in a system group	56
General DB2 customization (DB2 data sharing)	56
Update topology setting in the Workflow database MQSeries customization	57
OS/390 trace customization	58
CICS API support customization	59
IMS API support customization	61
Workflow server customization	61
LAN client customization	62
Customize the MQSeries client connection	62
Customize the MQSeries Workflow client	63
Chapter 6. Adding extra Workflow clients to an existing system	65
Basic client customization	65
Decide the new client's identifiers	65
Data set allocation for client	66
Create input files for customizing a new Client MQSeries customization for a new client	66
Generate MQSeries channel tab file for LAN client	67
CICS API support customization for new client	68
IMS API support customization for new client	70
LAN Client Customization	70
Client request concentrator customization	71

Chapter 1. Understanding MQSeries Workflow for OS/390

This chapter describes the architecture and customization tasks.

- “Understanding the MQSeries Workflow for OS/390 architecture”
- “Understanding the customization process” on page 4

Understanding the MQSeries Workflow for OS/390 architecture

Figure 1 illustrates the MQSeries Workflow for OS/390 architecture.

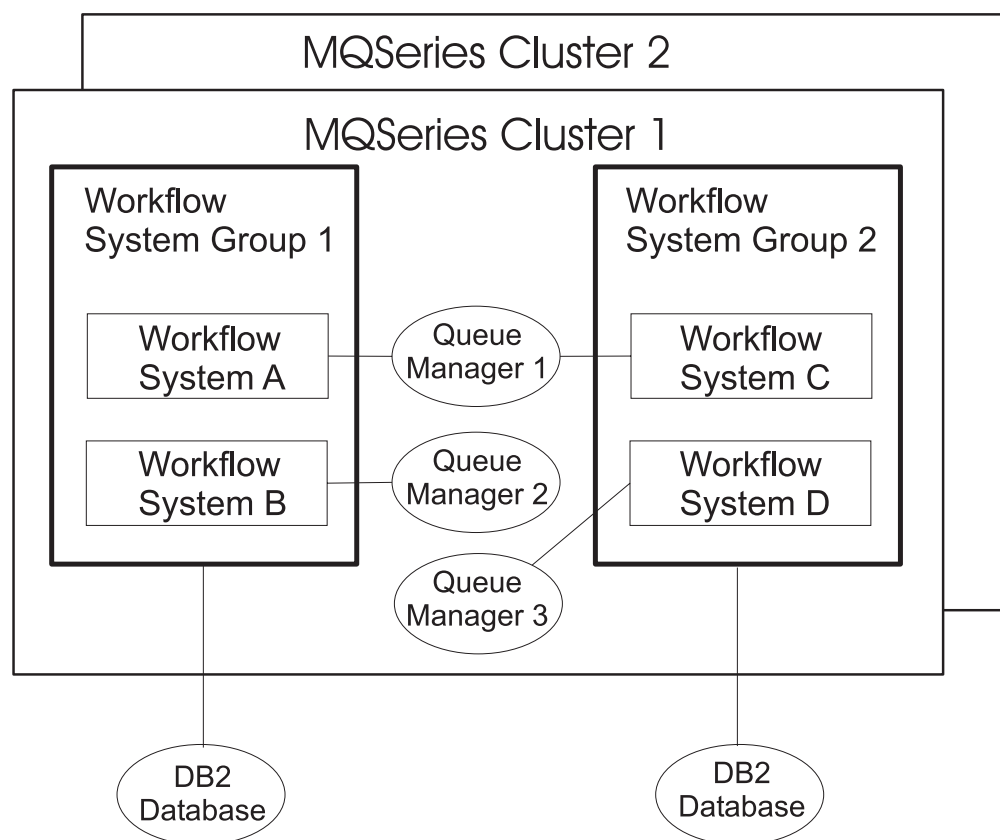


Figure 1. MQSeries Workflow for OS/390 architecture

The architecture consists of the following components:

MQSeries cluster

At least one MQSeries cluster must be defined. Each cluster can contain one or more system groups. By using MQSeries clusters, your MQSeries administration is simplified, and the workflow requests can be distributed evenly between Workflow systems. For more information about clusters, see *MQSeries: Cluster Support*.

System group

A Workflow system group can contain one or more Workflow systems. Each system group is a member of exactly one cluster. Multiple system groups can share the same DB2 subsystem.

System

A Workflow system consists of servers for administration, program execution, execution, scheduling, and cleanup. Systems within the same system group cannot use the same queue manager, but systems that are in different system groups can use the same queue manager. The systems in a system group can reside on different OS/390 images, but they must use the same database using DB2 data sharing

MQSeries queue manager

At least one MQSeries queue manager must exist. Every system in a system group must use a different queue manager.

DB2 MQSeries Workflow for OS/390 requires at least one DB2 subsystem. The DB2 database used by the systems in a system group can either be in the same subsystem, or in a data sharing group.

The first time that you customize MQSeries Workflow for OS/390, you will create one Workflow system group containing one Workflow system. To create further system groups, you will have to repeat the customization procedure using unique customization parameters. Before you will be able to create a system group, you must ensure that you have the following minimum environment for an MQSeries Workflow for OS/390 system group:

- One MQSeries cluster
- One MQSeries queue manager
- One DB2 subsystem

After creating a system group containing one Workflow system, you can add additional systems to the system group as described in “Chapter 5. Creating additional systems in an existing system group” on page 53.

Understanding the customization process

The tasks required to customize MQSeries Workflow for OS/390 depend on the way that you intend to structure your configuration. The possible dimensions of a system are shown in Figure 2 on page 5.

1. The installation procedure is described in *MQSeries Workflow for OS/390: Program Directory*.
2. After each installation, you must perform “Post-installation tasks” on page 19 once for every OS/390 image that you want to customize. This defines OS/390 resources that have scope over an OS/390 system. These resources are:
 - The MQSeries Workflow for OS/390 message catalog.
 - Modules which have to be loaded into the LPA.
3. To create each system group and a primary system, you must perform:
 - a. “Deciding your MQSeries Workflow for OS/390 identifiers” on page 7
 - b. “Chapter 4. Creating a system group and a primary system” on page 23. This defines all resources that have scope over a Workflow system:
 - MQSeries Workflow for OS/390 system prefix.
 - MQSeries Workflow for OS/390 databases.
 - MQSeries cluster name.
 - MQSeries queue manager.

After successfully performing this step, your Workflow system has the same capabilities as for the MQSeries Workflow for OS/390 Version 3.1. Only now

can you proceed to perform steps 4 and 5 in any order, and as often as necessary to create the required configuration.

4. You can then add systems to a system group as described in “Chapter 5. Creating additional systems in an existing system group” on page 53. This allows you to scale an existing system over a parallel sysplex. With this additional Workflow system, the following resources are defined:
 - An additional MQSeries queue manager, which must be different to the one used by the primary system, but they must exist in the same MQSeries cluster.
 - Additional execution server and program execution server.
5. You can add Workflow clients to primary or additional systems as described in “Chapter 6. Adding extra Workflow clients to an existing system” on page 65. This step sets up an additional queue manager in the same MQSeries cluster as the primary system. There are many different types of client which can be setup.

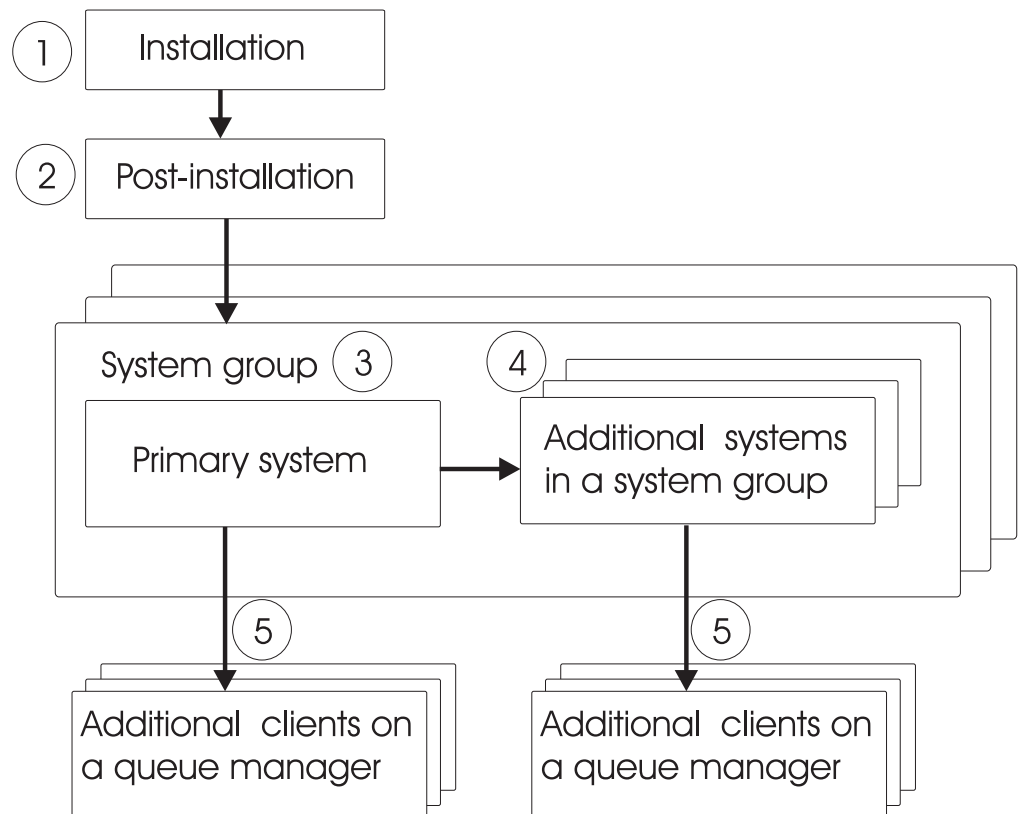


Figure 2. MQSeries Workflow for OS/390 customization tasks

Customization scenario pre-requirements

The following sections describe the pre-requirements for two scenarios:

Scenario 1: Minimum configuration (one system group containing one system)

The minimum requirements before starting customization for an MQSeries Workflow for OS/390 system group with a primary system are:

1. It is assumed that there is a single queue manager.
2. The queue manager is the repository for the cluster.

3. A cluster receiver channel is defined.
4. You have verified that the cluster works correctly.

Scenario 2: Multiple queue managers (additional systems or clients)

The minimum requirements before starting customization for an MQSeries Workflow for OS/390 with multiple queue managers in the cluster (either for additional systems or additional clients on a queue manager), the minimum requirements are:

1. It is assumed that all relevant queue managers are in a single cluster.
2. At least two queue managers are defined as repository queue managers for the cluster.
3. Each queue manager has a cluster receiver channel.
4. Each queue manager has a cluster sender channel to a repository queue manager.
5. You have verified that the cluster works correctly.

Chapter 2. Planning your configuration

Before starting to customize your MQSeries Workflow for OS/390 system, you should complete the following:

1. "Deciding your MQSeries Workflow for OS/390 identifiers"
2. "Evaluate database requirements" on page 15

Deciding your MQSeries Workflow for OS/390 identifiers

You install the product image from the tape to the location that is specified by the MQSeries Workflow for OS/390 installation high level qualifier *InstHLQ*. Each time that you want to create a new MQSeries Workflow for OS/390 system you must specify a new customization high level qualifier *CustHLQ*. It determines where the new system files are copied and customized.

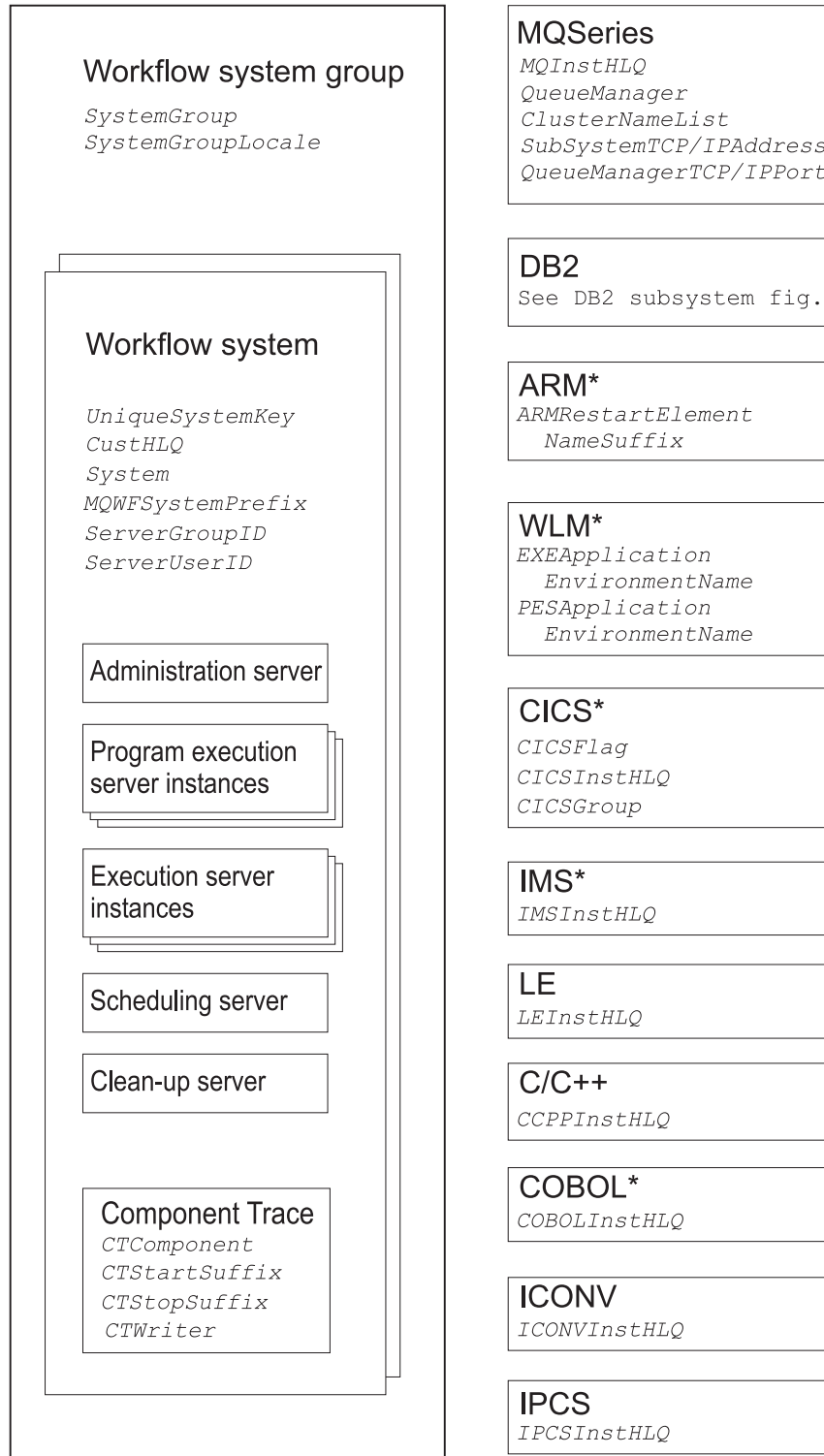
We recommend that you copy and complete the following tables for each MQSeries Workflow for OS/390 system you want to plan. The identifiers decided here will be entered into the customization parameter file during "Pre-customization" on page 23. During "Create input files for customization" on page 24, these parameters are automatically substituted in the customization jobs. The customization parameter file is listed in "Customization parameter file for a primary system" on page 183.

It may be necessary for the information to be agreed and exchanged between the following people:

- OS/390 system administrator
- CICS administrator
- IMS administrator
- DB2 administrator
- RACF administrator
- MQSeries administrator
- MQSeries Workflow local area network (LAN) administrator
- MQSeries Workflow for OS/390 administrator

Overview of the customization parameters

The main system components and associated customization parameters are illustrated in Figure 3 on page 8 and Figure 4 on page 9.



* Using these features is optional.

Figure 3. Customization parameters for a Workflow system

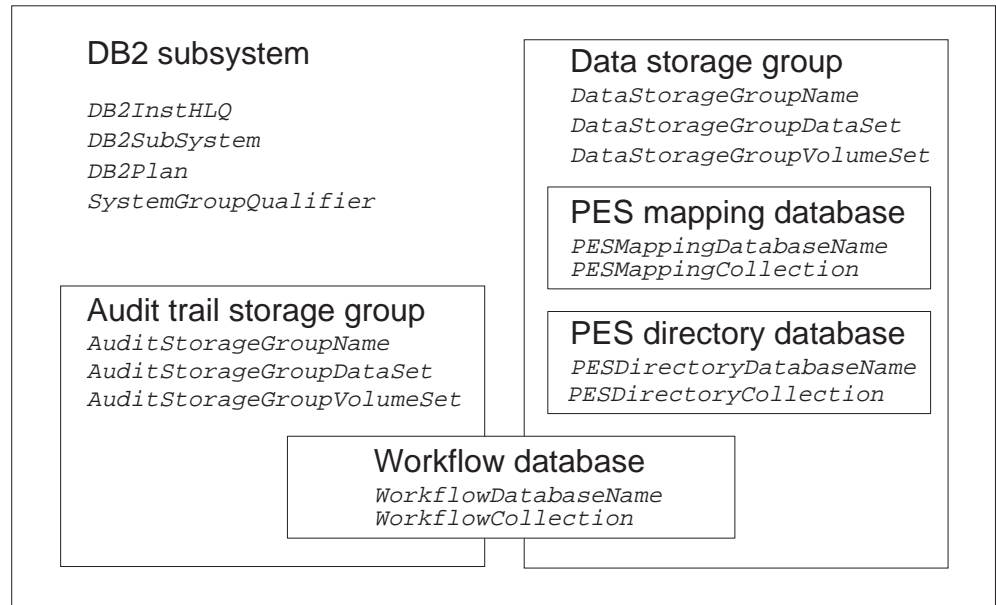


Figure 4. Customization parameters for DB2

Installation scope identifiers

The following identifiers have scope over a Workflow installation.

Table 1. Installation scope identifiers

Parameter	Your value	Name in customization parameter file	Description
<i>InstHLQ</i>		MQWFIHLQ	MQSeries Workflow for OS/390 installation high level qualifier. This qualifier is determined when the product image is installed from tape, this is described in <i>MQSeries Workflow for OS/390: Program Directory</i> .
<i>CustHLQ</i>		MQWFCHLQ	The high level qualifier for the MQSeries Workflow for OS/390 system you want to customize.
<i>ClientHLQ</i>		CLNTCHLQ	Client customization high level qualifier for MQSeries Workflow.

System group scope identifiers

The following identifiers have scope over a Workflow system group.

Table 2. System group identifiers

Parameter	Your value	Name in customization parameter file	Description
<i>SystemGroup</i>		MQWFSGNM	MQSeries Workflow for OS/390 system group name. This name must be unique within your MQSeries Workflow domain.
<i>MQWFSystemPrefix</i>		MQWFSYSP	MQSeries Workflow for OS/390 MQSeries queue prefix is used to prefix MQSeries object names, for example queue names. This identifier may be up to 8 uppercase characters long.
<i>MQClusterName</i>		MQWFCLST	The MQSeries name of a cluster where the system group resides. This name will be used to define all MQSeries objects which have cluster scope. This cluster name must be unique in your entire MQSeries network to avoid collisions and to allow future extensions.
<i>DB2SampleLoad Library</i>		DB2SMPRL	The load library where the DB2 sample DSNTEP2 is located for example, <DB2InstHLQ>.RUNLIB.LOAD
<i>DB2SampleDBRM Library</i>		DB2SMPDL	The library where the DSNTEP2 DBRM is located, for example, <DB2InstHLQ>.DBRMLIB.DATA
<i>SystemGroupLocale</i>		MQWFSGLC	MQSeries Workflow for OS/390 system group locale setting. This is used to set the locale for all servers and tools. It selects the correct code page conversion. Use 'C' for the default on your machine or a specific locale setting, for example, 'De_DE.IBM-273' for German. The MQSeries Workflow for OS/390 servers and utilities load the active locale from the C environment variable LC_ALL to determine to which local code page MQSeries Workflow messages from remote clients should be converted. The LC_ALL environment variable is set and propagated to the MQSeries Workflow programs with the LE runtime option ENVAR.
<i>SystemGroupPrefix</i>		DB2SGPRE	DB2 system group object qualifier. This is used to prefix all DB2 objects created for this <i>SystemGroup</i> .
<i>DataStorageGroup Name</i>		DB2STGNW	DB2 storage group name where your MQSeries Workflow for OS/390 data will be stored.
<i>DataStorageGroup DataSetPrefix</i>		DB2STGPW	DB2 storage group data set prefix for runtime data.
<i>DataStorageGroup VolumeSet</i>		DB2STGVW	DB2 storage group volume set for runtime data. Specify the volume name or '*' for SMS managed volumes.
<i>AuditStorageGroup Name</i>		DB2STGNA	DB2 storage group name for the audit trail data.
<i>AuditStorageGroup DataSetPrefix</i>		DB2STGPA	DB2 storage group data set prefix for audit trail data.
<i>AuditStorageGroup VolumeSet</i>		DB2STGVA	DB2 storage group volume set for the audit trail data. Specify the volume name or '*' for SMS managed volumes. Note: For performance reasons this should not be the same volume used for <i>DataStorageGroupVolumeSet</i> .
<i>WorkflowDatabaseName</i>		DB2DBNAM	DB2 Workflow database name.

Table 2. System group identifiers (continued)

Parameter	Your value	Name in customization parameter file	Description
<i>PESMappingDatabaseName</i>		DB2MDBNM	DB2 program execution server (PES) mapping database name.
<i>PESDirectoryDatabaseName</i>		DB2PDBNM	DB2 PES directory database name.
<i>WorkflowCollection</i>		DB2DBC0L	DB2 Workflow database collection name.
<i>PESMappingCollection</i>		DB2MDC0L	DB2 PES mapping database collection name.
<i>PESDirectoryCollection</i>		DB2PDC0L	DB2 PES directory database collection name.
<i>DB2Plan</i>		DB2PLANN	DB2 database plan name.

System scope identifiers

The following identifiers have scope over a Workflow system. To create multiple systems within a system group, you will have to define unique system scope identifiers for each system, as described in “Chapter 5. Creating additional systems in an existing system group” on page 53.

Table 3. System scope identifiers

Parameter	Your value	Name in customization parameter file	Description
<i>UniqueSystemKey</i>		MQWFKUKEY	Unique key for an MQSeries Workflow for OS/390 system, may be up to eight uppercase characters long. This is the name given to the Workflow server start job, and must be unique within SYS1.PROCLIB. This key is used in the START command to start an administration server on the <i>System</i> associated with this key.
<i>System</i>		MQWFSYSN	MQSeries Workflow for OS/390 system name. This name must be unique within the system group. This is the system where the administration server is started when the start administration server command is issued: START <i>UniqueSystemKey.AdminServerID</i> .
<i>ServerUserID</i>		STTSKUID	The server started task RACF user ID used by all MQSeries Workflow for OS/390 servers. This is the default user ID that OS/390 programs will be run under, by the PES, as a result of MQSeries Workflow process activity requests for OS/390 program invocations. This user ID requires EXECUTE rights on <i>DB2Plan</i>
<i>ServerGroupID</i>		STTSKGRP	The server started task RACF group ID for all MQSeries Workflow for OS/390 servers.
<i>CTComponent</i>		CTRCNAME	CTRACE component name.
<i>CTStartSuffix</i>		CTRCPMS1	CTRACE PARMLIB member suffix (start writer). This value may be any two alphanumeric characters.
<i>CTStopSuffix</i>		CTRCPMS2	CTRACE PARMLIB member suffix (stop writer). This value may be any two alphanumeric characters.
<i>CTWriter</i>		CTRCWPRC	CTRACE writer procedure name. This must not be more than seven characters long.
<i>ARMRestartPolicy</i>		ARMPOLNM	The name of the ARM restart policy.

Table 3. System scope identifiers (continued)

Parameter	Your value	Name in customization parameter file	Description
<i>ARMRestartElementNameSuffix</i>		ARMRESFX	The 8 character suffix of the 16 character ARM restart element name. This value must be specified as part of the start administration server command to register the administration server with ARM. This suffix is concatenated to the constant prefix SYSMQWF_. For example, if you set ARMRESFX=MQWFS1, then the name of the ARM restart element that is used in the ARM restart policy will be SYSMQWF_MQWFS1.
EXE <i>ApplicationEnvironmentName</i>		WLMAEEXE	Application Environment for the MQSeries Workflow for OS/390 execution server . It must be unique in the parallel sysplex environment.
PES <i>ApplicationEnvironmentName</i>		WLMAEPES	Application Environment for the MQSeries Workflow for OS/390 program execution server . It must be unique in the parallel sysplex environment.
<i>ClusterNameList</i>		MQCLNAME	This is the name of a MQSeries Namelist object which holds the MQSeries cluster name that you defined in Table 2 on page 10. The name must conform to the MQSeries naming rules for Object names. It is used later in all MQSeries object definitions which have cluster scope.
<i>QueueManager</i>		MQQMNAME	Name of the MQSeries queue manager that is to be used by MQSeries Workflow for OS/390. The queue manager name must be unique for the complete MQSeries network. Note: If you want to run CICS applications that use the MQSeries Workflow for OS/390 application program interface (API), this must either be the same queue manager that is used by CICS, or the queue managers must be members of the same MQSeries cluster.
<i>SubsystemTCP/IPAddress</i>		STCPADDR	This is the TCP/IP address of the OS/390 Subsystem where the MQSeries Queue Manager resides.
<i>QueueManagerTCP/IPPort</i>		STCPPORT	This is the TCP/IP port of the listener of the Queue Manager. The MQSeries default is 1414. All Queue Managers on the OS/390 image must have different ports for their listeners.
<i>MQWFConfigurationKey</i>		MQWF CFGK	This key must be unique for all MQSeries Workflow for OS/390 systems that you configure. It can be up to 8 uppercase characters long. This key is used inside the profiles and identifies a configuration for a system.

Flags and high level qualifiers

The following flag and high level qualifiers are used during customization.

Table 4. Flags and high level qualifiers

Parameter	Your value	Name in customization parameter file	Description
<i>CICSFlag</i>	Use one of the values provided in the customization parameter file.	CICSFL	This parameter determines whether a CICS installation library is included. The default setting in the customization parameter file assumes that CICS is installed. If you do not have CICS installed, later when you reach step 2 of “Create input files for customization” on page 24, you will only have to comment out the default line and remove the comment symbol from the front of the alternative setting. For more details see the comment sections of the listing in “Customization parameter file for a primary system” on page 183.
<i>CICSInstHLQ</i>	*	CICSLPFX	CICS installation high level qualifier.
<i>DB2InstHLQ</i>		DB2INHLQ	DB2 installation high level qualifier.
<i>MQInstHLQ</i>		MQPREFIX	MQSeries installation high level qualifier.
<i>LEInstHLQ</i>		LELIBPFX	Language Environment installation high level qualifier.
<i>CCPPInstHLQ</i>		CLIBRPFX	C/C++ installation high level qualifier.
<i>COBOLInstHLQ</i>	*	CBLIBPFX	COBOL installation high level qualifier
<i>IMSInstHLQ</i>	*	IMSLIBPX	IMS installation high level qualifier.
<i>ICONVInstHLQ</i>		ICONVPFX	ICONV installation high level qualifier. This should point to the unicode converter data sets mentioned below, it is normally the same as the Language Environment high level qualifier (<i>LEInstHLQ</i>). The value is substituted in the environment variable file, see “Appendix I. Environment variable files” on page 197. In the following example, <i>ICONVInstHLQ</i> should be set to SYS1: SYS1.SCEEUCS2 SYS1.SCEEUCS2.UCMAP SYS1.SCEEUCS2.UCONVTBL
<i>IPCSInstHLQ</i>		IPCSPRFX	IPCS installation high level qualifier.

* CICS, IMS, and COBOL are optional.

Subsystem identifiers

The following subsystem identifiers are required for customization.

Table 5. Subsystem identifiers

Parameter	Your value	Name in customization parameter file	Description
<i>DB2SubSystem</i>		DB2SSYSN	Name of the DB2 subsystem that is to be used by MQSeries Workflow for OS/390.
<i>CICSGroup</i>		CICSGRPN	CICS group name used for program execution server invocations.

Customization identifiers

These identifiers are not present in the customization parameter file. Many of these parameters are optional, depending on which invocation types you intend to use.

Table 6. Customization identifiers

Parameter	Your value	Description
<i>DB2AdminUserID</i>		The user ID of the DB2 administrator. This user ID requires SYSADM rights to be able to perform the customization process. This can be granted with the command <code>GRANT SYSADM TO DB2AdminUserID</code>
<i>MQWFAdminUserID</i>		The user ID of the MQSeries Workflow for OS/390 administrator. This user ID requires EXECUTE rights on <i>DB2Plan</i> to be able to execute the tools that update the runtime databases.
<i>MQHostName</i>		The TCP/IP host name of the OS/390 where the queue manager is installed. This value is required during "Customize the MQSeries client connection" on page 33.
<i>applid</i>		This value is required during "Customize CICS EXCI invocation" on page 39 and "Customize program execution server directory" on page 47.
<i>netid</i>		This value is required during "Customize IMS CPIC invocation" on page 44 and "Customize program execution server directory" on page 47.
<i>luname</i>		This value is required during "Customize IMS CPIC invocation" on page 44 and "Customize program execution server directory" on page 47.
<i>CICSBridgeInputQueue</i>		This value is required during "Customize MQSeries CICS bridge invocation" on page 41 and "Customize program execution server directory" on page 47.
<i>IMSBridgeInputQueue</i>		This value is required during "Customize MQSeries IMS bridge invocation" on page 46 and "Customize program execution server directory" on page 47.
<i>XCFGroupName</i>		This value is required during "Customize MQSeries IMS bridge invocation" on page 46. The MQSeries instance and the target IMS system must belong to the same XCF group.
<i>XCFMemberIMS</i>		This value is required during "Customize MQSeries IMS bridge invocation" on page 46. It represents the IMS system as a member in the XCF group <i>XCFGroupName</i> .

Table 6. Customization identifiers (continued)

Parameter	Your value	Description
<i>XCFMemberMQ</i>		This value is required during “Customize MQSeries IMS bridge invocation” on page 46. It represents the MQSeries instance as a member in the XCF group <i>XCFGroupName</i> .
<i>PESDirectory</i> <i>SourceFile</i>		Your PES directory is based on a skeleton. After customizing the source file, any new changes will have to be made to your source file. This value is required during “Customize program execution server directory” on page 47.

Evaluate database requirements

If you want to use two storage groups, ten buffer pools, approximately 600MB of primary allocation for Workflow data, and about 160MB of primary allocation for audit trail data; you can use the suggested database allocations in Table 9 on page 17, and no further planning is necessary. Otherwise, more detailed database planning is required.

The suggested database allocations are suitable for the operational Workflow scenario described in Table 7.

Table 7. Sample scenario characteristics suitable for the suggested database allocations

Parameter	Value
Program activities per process	10
Process lifetime	1 day
Finished processes kept	7 days
Created processes	100 per day
Work items per program activity	10
Audit trail	condensed
Audit trail cleanup every	7 days
Average small container size	512 bytes
Average large container size	4096 bytes

More detailed database planning (optional)

If the suggested values are not acceptable for your requirements, this step helps you to determine the size and organization of your database. Later, you will use the values that are decided here to customize the jobs that create the DB2 objects for MQSeries Workflow for OS/390.

This is a planning phase. You should not modify any of the files mentioned here, copies of these files will be generated during “Pre-customization” on page 23.

The values you decide on must be consistent with your values that you have already planned, especially those in Table 2 on page 10.

1. Print a copy of *InstHLQ.SFMCDB2(FMCHDDBP)* to help you decide how many buffer pools and which buffer pool sizes you want.
2. Print a copy of *InstHLQ.SFMCDB2(FMCHDDST)* to help you decide how many storage groups you want to use, and whether you want to use volume names or SMS managed volumes in the storage group definitions.
3. To help you decide which buffer pool or storage group you want to use for each database, table space, or index, and estimate the required sizes for table spaces and indexes, print a copy of the files listed in Table 8.

Table 8. Files that define the databases

Database name (see Table 2 on page 10)	Database definitions	Table space definitions	Table and index definitions
<i>Workflow DatabaseName</i>	<i>InstHLQ.SFMCDB2 (FMCHDDDB)</i>	<i>InstHLQ.SFMCDB2 (FMCHDDTS)</i>	<i>InstHLQ.SFMCDB2 (FMCHDDTB)</i>
<i>PESDirectory DatabaseName</i>	<i>InstHLQ.SFMCDB2 (FMCHDDPD)</i>	<i>InstHLQ.SFMCDB2 (FMCHDDPS)</i>	<i>InstHLQ.SFMCDB2 (FMCHDDPT)</i>
<i>PESMapping DatabaseName</i>	<i>InstHLQ.SFMCDB2 (FMCHDDMD)</i>	<i>InstHLQ.SFMCDB2 (FMCHDDMS)</i>	<i>InstHLQ.SFMCDB2 (FMCHDDMT)</i>

Table 9 on page 17 provides a summary of the suggested table space sizes, buffer pool sizes, and buffer pool allocations. It is recommended that you use a copy of this table for your detailed planning.

Table 9. Suggested buffer pool sizes and allocation

Database or Table space	Suggested table space size in 1000 KB (PRIQTY)	Storage group	Buffer pool IDs										
			BP 32K	BP0	BP1	BP2	BP3	BP4	BP5	BP6	BP7	BP8	
			Suggested buffer pool size (x1000 pages)										
			2	4	4	4	4	2	2	2	4	4	
Database	-			•									
PESDIRTS	1												•
MAPPING	1												•
TEMPLT32	12		•										
TEMPLT04	12												•
CONTAINR	120										•		
PROCESS	12												•
WORKITEM	72				•								
NTFYITEM	12								•				
ACTWI	20							•					
PROCACT	12									•			
PROGACT	12									•			
BLOCKACT	12									•			
BLOCK	12									•			
STAFF04	16												•
STAFF32	16		•										
MPOOL	1		•										
TOPLGY32	20		•										
TEST	12												•
ADMIN	12												•
ADMIN01	12												•
MODEL	20		•										
LIST	12		•										
Indexes	-				•		•						
ADTTRAIL *	160	*											•

Note: * The audit trail table space ADTTRAIL should be on a separate volume for performance reasons. By default it is allocated to the volume *AuditStorageGroupVolumeSet* and all other tables and databases are allocated to the volume *DataStorageGroupVolumeSet*. See your values in Table 2 on page 10.

Chapter 3. Before starting customization

Before starting customization, you should check the following:

1. You have MQSeries for OS/390 Version 2.1 installed, and one queue manager with cluster setup is available for MQSeries Workflow for OS/390.
2. You have DB2 for OS/390 Version 5.1 installed, and one subsystem is available for MQSeries Workflow for OS/390.
3. To perform customization, you must have DB2 SYSADM rights.
4. IBM Resource Access Control Facility (RACF) authority to alter the MQSeries Workflow for OS/390 installation data sets, and the right to create MQSeries objects.

Note: This manual assumes that you are using RACF for your security. If you are using a different security system, you must apply the equivalent security access controls for your system.

5. RACF authority to alter PROCLIB and PARMLIB.
6. The load library *InstHLQ*.SFMCLINK must be Advanced Program Facility (APF) authorized.
7. You should have configured the Resource Recovery Service (RRS) as described in *OS/390 MVS Programming: Resource Recovery*.
8. Then you are ready to perform "Post-installation tasks" followed by
9. "Pre-customization" on page 23

Post-installation tasks

After performing the installation as described in *MQSeries Workflow for OS/390: Program Directory*, you are ready to update the MVS Message Services (MMS) message catalog, and copy the LPALIB member.

Note: Before submitting each JCL, be sure to insert your own job card.

Create the MMS message catalogs

To add the MQSeries Workflow for OS/390 messages to MMS, you must do the following:

Table 10. Create MMS message catalogs

Step number	Required or optional	Description	Action	Verification
1	Required	Update parameters in JCLs.	<ol style="list-style-type: none"> 1. Edit <i>InstHLQ.SFMCCNTL</i> (FMCHJMM1) <ol style="list-style-type: none"> a. Replace <MQWFIHLQ> with your MQSeries Workflow for OS/390 installation high level qualifier, see <i>InstHLQ</i> in Table 1 on page 9. b. Replace <MMSVOL> with the volume name where the VSAM cluster for the MQSeries Workflow for OS/390 message catalogs should reside. c. Replace <STORCLAS> with the storage class of the volume where the VSAM cluster for the MQSeries Workflow for OS/390 message catalogs should reside. 2. Edit <i>InstHLQ.SFMCCNTL</i> (FMCHJMM2) <ol style="list-style-type: none"> a. Replace <MQWFIHLQ> with your MQSeries Workflow for OS/390 installation high level qualifier. 3. Edit <i>InstHLQ.SFMCPARM</i> (FMCHYMMS) <ol style="list-style-type: none"> a. Replace <MQWFIHLQ> with your MQSeries Workflow for OS/390 installation high level qualifier. 	
2	Required	Create the VSAM clusters.	Submit JCL <i>InstHLQ.SFMCCNTL</i> (FMCHJMM1)	rc = 0
3	Required	Load the input files <i>InstHLQ.SFMMSG</i> (FMCHMxxx) to the VSAM clusters.	Submit JCL <i>InstHLQ.SFMCCNTL</i> (FMCHJMM2)	rc = 0
4	Required	Copy MMS PARMLIB member.	Being careful not to overwrite an existing PARMLIB member: Copy <i>InstHLQ.SFMCPARM</i> (FMCHYMMS) to your system PARMLIB.	
5	Required	Rename the MMS PARMLIB member.	Being careful not to overwrite an existing PARMLIB member, rename the MMS PARMLIB member that you copied in step 4 to MMSLSTxx.	
6	Required	Provide RACF profile.	Give the system address space MMS read access to the VSAM clusters created in step 2.	
7	Required	Access the PARMLIB member.	Either <ul style="list-style-type: none"> • Specify MMS(xx) on the INIT statement in SYS1.PARMLIB(CONSOLnn), or • Issue the operator command SET MMS=xx 	

LPA library concatenation

To be able to access the MQSeries Workflow for OS/390 trace writer FMCHXTRC correctly, and to improve the performance when using the MQSeries Workflow trace facility, you must perform the following action:

Table 11. Copy LPALIB member

Step number	Optional or required	Description	Action
1	Required	Add MQSeries Workflow for OS/390 LPA library to your LPA library concatenation list.	Concatenate the MQSeries Workflow for OS/390 LPA library <i>InstHLQ.SFMCLPA</i> to your LPA library concatenation list.

Chapter 4. Creating a system group and a primary system

This chapter will guide you through the customization tasks necessary to make MQSeries Workflow for OS/390 functional within your system. This procedure consists of the following stages:

- “Pre-customization” is required.
- “System customization” on page 24 is required.
- “Verify Workflow client sample application” on page 37 is optional.
- “Program execution customization” on page 39 is optional.

Pre-customization

Each time that you want to create a new MQSeries Workflow for OS/390 system, you must perform a customization. Before starting customization, you must perform the following pre-customization task. This creates the libraries and copies files from the installation image (*InstHLQ*) to the location for the new system that is to be customized (*CustHLQ*). The information you enter during this task is used to generate customization files.

Data set allocation

This step creates the data sets that are required for customization.

Table 12. Data set allocation

Step number	Required or optional	Description	Action	Verification
1	Required	Copy allocation job.	Copy the JCL <i>InstHLQ</i> .SFMCCNTL(FMCHJACD) to a private partitioned data set.	
2	Required	Customize allocation job.	Edit your copy of FMCHJACD, and make the changes described in the comment header of the file (replace <MQWFCHLQ> with your MQSeries Workflow for OS/390 customization high level qualifier, see <i>CustHLQ</i> in Table 1 on page 9).	
3	Required	Allocate customization data sets.	Submit your copy of FMCHJACD.	rc=0 indicates that the following libraries have been created: <ol style="list-style-type: none">1. <i>CustHLQ</i>.SFMCCNTL2. <i>CustHLQ</i>.SFMCDATA3. <i>CustHLQ</i>.SFMADB24. <i>CustHLQ</i>.SFMCMQS5. <i>CustHLQ</i>.SFMCPARM6. <i>CustHLQ</i>.SFMCPROC7. <i>CustHLQ</i>.SFMCREXX8. <i>CustHLQ</i>.GENPROC9. <i>CustHLQ</i>.GENPARM <p>Note: The last two libraries are for the generated PROCLIB and PARMLIB members.</p>

Create input files for customization

In this task you specify all the identifiers that the customization process requires, and generate customization files from the values you have entered. If you later realize that the identifiers were not correct, you must repeat this task before repeating the customization process.

Table 13. Create input files for customization

Step number	Required or optional	Description	Action	Verification
1	Required	Copy customization templates.	<ol style="list-style-type: none"> 1. Copy the JCL <i>InstHLQ.SFMCCNTL(FMCHJCCT)</i> to a private partitioned data set. 2. Edit your copy of FMCHJCCT as described in the comment header. 3. Submit your copy of FMCHJCCT 	rc=0 for the copy step. The JCL also deletes any members in the libraries for the generated PROCLIB and PARMLIB members <i>CustHLQ.GENPROC</i> and <i>CustHLQ.GENPARM</i> . rc=8 can be accepted for the delete step, it indicates that there was nothing to delete.
2	Required	Edit the customization parameter file.	<p>Edit the customization parameter template member <i>CustHLQ.SFMCDATA(FMCHCIF)</i>, and enter your values from the tables in “Chapter 2. Planning your configuration” on page 7, as described in the comment sections of the file.</p> <p>Note: This file is described in “Customization parameter file for a primary system” on page 183. From now on, this member will contain your customization parameters. This member is used as an input file for the generation process in step 3.</p>	
3	Required	Generate all the JCLs necessary to customize this product.	<ol style="list-style-type: none"> 1. Copy the JCL <i>CustHLQ.SFMCCNTL(FMCHJCUS)</i> to a private partitioned data set. 2. Edit your copy of FMCHJCUS as described in the comment header. 3. Submit your copy of FMCHJCUS. 	This requires rc=0. The program performs some syntax checking on the length and value of the variables you specified in the file <i>CustHLQ.SFMCDATA(FMCHCIF)</i> . The program then substitutes your values for variables in the customization template files. Some PROCLIB and PARMLIB members are also copied with new names to the library <i>CustHLQ.GENPROC</i> and <i>CustHLQ.GENPARM</i> .

When you have completed this stage, the JCL files that are required in the following chapters will contain all the customization parameters that you determined in “Chapter 2. Planning your configuration” on page 7.

System customization

To customize the MQSeries Workflow for OS/390 system, you must perform the following tasks in the given sequence:

1. “General DB2 customization” on page 25
2. “Workflow DB2 customization” on page 25
3. “Program execution server directory DB2 customization” on page 27
4. “Program execution server mapping DB2 customization” on page 28
5. “MQSeries customization” on page 29
6. “OS/390 trace customization” on page 30

7. "CICS API support customization" on page 30
8. "IMS API support customization" on page 32
9. "Workflow server customization" on page 32
10. "LAN client customization" on page 33
11. "Customize Java-API support" on page 34
12. "System customization verification" on page 36

After completing the above tasks, you will be able to connect a MQSeries Workflow client to MQSeries Workflow for OS/390.

General DB2 customization

Before performing this customization you should ensure that you have DB2 SYSADM grants. This can be granted with the command:

```
GRANT SYSADM TO DB2AdminUserID
```

Before submitting each JCL, be sure to insert your own job card.

Table 14. General DB2 customization

Step number	Required or optional	Description	Action	Verification
1	Required	Bind the plan for the DB2 sample application DSNTEP2.	Submit JCL <i>CustHLQ</i> .SFMCCNTL (FMCHJBTE)	rc=0
2	Optional	If you want to change the buffer pool names and sizes:	Edit <i>CustHLQ</i> .SFMCDDB2 (FMCHDDBP), and change the buffer pool definitions.	
	Required	Define buffer pools.	Submit JCL <i>CustHLQ</i> .SFMCCNTL (FMCHJDBP)	rc=0
4	Optional	If you want the storage groups to use more than one volume name, or SMS managed volumes:	Edit <i>CustHLQ</i> .SFMCDDB2 (FMCHDDST), and change the VOLUMES parameter as necessary.	
	Required	Create storage groups.	Submit JCL <i>CustHLQ</i> .SFMCCNTL (FMCHJDST)	rc=0

Workflow DB2 customization

To create the Workflow database, you must perform the following steps:

Table 15. Workflow DB2 customization

Step number	Required or optional	Description	Action	Verification
1	Optional	If you want to change the default buffer pool name or the storage group for the database:	Edit <i>CustHLQ</i> .SFMCDDB2 (FMCHDDDB), and change the buffer pool name and storage group.	
	Required	Create Workflow database.	Submit JCL <i>CustHLQ</i> .SFMCCNTL (FMCHJDDB)	rc=0

Table 15. Workflow DB2 customization (continued)

Step number	Required or optional	Description	Action	Verification
2	Optional	If you want to change the buffer pool names to be used for the table spaces, or the value for the primary space allocation:	Edit <i>CustHLQ.SFMCDB2</i> (FMCHDDTS), and change the buffer pool names. You can also change the value for the primary space allocation PRIQTY to the required size (in KB).	
	Required	Create Workflow table spaces.	Submit JCL <i>CustHLQ.SFMCCNTL</i> (FMCHJDTS)	rc=0. If you get a non-zero return code, you can roll back the complete action by dropping the Workflow database using the job FMCHJEDB. After this step you have to start again with <i>Step number 1: Create Workflow database</i> .
3	Optional	If you want to change the buffer pools, or the value for the primary space allocation for the indexes:	Edit <i>CustHLQ.SFMCDB2</i> (FMCHDDTB), and change the buffer pool names. You can also change the value for the primary space allocation PRIQTY to the required size (in KB).	
	Required	Create Workflow tables.	Submit JCL <i>CustHLQ.SFMCCNTL</i> (FMCHJDTB)	rc=0. If you get a non-zero return code, you can roll back the complete action by dropping the Workflow table spaces using the job FMCHJETS. After this step you have to start again with <i>Step number 2: Create Workflow table spaces</i> .
4	Required	Bind the Workflow packages and add the Workflow Collection to the Workflow plan.	Submit JCL <i>CustHLQ.SFMCCNTL</i> (FMCHJBDB)	rc=4 can be accepted.
5	Required	Be sure that RRS is active.	If RRS is not active, you can activate it by issuing the command: START RRS	

Populate the Workflow database

To populate and verify the Workflow database, you must perform the following steps:

Step number	Required or optional	Description	Action	Verification
1	Required	Populate the Workflow database with initial settings.	Submit JCL <i>CustHLQ.SFMCCNTL</i> (FMCHJRBS)	rc=0. If you get a non-zero return code, you can rollback the action by deleting the contents of the Workflow tables using the job FMCHJEDC. If you perform a rollback, you have to repeat <i>Step 1</i> .
2	Required	Populate the Workflow database with initial topology settings.	Submit JCL <i>CustHLQ.SFMCCNTL</i> (FMCHJRIB)	rc=0

Step number	Required or optional	Description	Action	Verification
3	Required	Run the DB2 utility RUNSTATS for the Workflow database.	Submit JCL <i>CustHLQ.SFMCCNTL(FMCHJRST)</i>	rc=4 can be accepted.
4	Required	Rebind the Workflow packages.	Submit JCL <i>CustHLQ.SFMCCNTL(FMCHJBDB)</i>	rc=4 can be accepted.
5	Required	Verify the Workflow database configuration.	Submit JCL <i>CustHLQ.SFMCCNTL(FMCHJCCH)</i>	Verify that each SELECT statement returns at least one row of data.

Program execution server directory DB2 customization

The program execution server (PES) directory contains the information about services and invocations that enables the PES to invoke CICS and IMS programs.

Table 16. Program execution directory DB2 customization

Step number	Required or optional	Description	Action	Verification
1	Optional	If you want to change the buffer pool names or storage group for the database:	Edit <i>CustHLQ.SFMCDB2(FMCHDDPD)</i> , and change the buffer pool names. You can also change the storage group.	
	Required	Create the PES directory database.	Submit JCL <i>CustHLQ.SFMCCNTL(FMCHJDPD)</i>	rc=0
2	Optional	If you want to change the buffer pool names for the table space, or if you want to change the primary space allocation:	Edit <i>CustHLQ.SFMCDB2(FMCHDDPS)</i> , and change the buffer pool names to be used for the table space. You can also set the value for the primary space allocation (PRIQTY) to the required size (in KB).	
	Required	Create the PES directory table space.	Submit JCL <i>CustHLQ.SFMCCNTL(FMCHJDPS)</i>	rc=0. If you get a non-zero return code, you can roll back the complete action by dropping the PES directory database using the job FMCHJEPD. After this step you have to start again with Step number 1: Create the PES directory database.
3	Optional	If you want to change the buffer pools, or if you want to change the primary space allocation for the indexes:	Edit <i>CustHLQ.SFMCDB2(FMCHDDPT)</i> , and change the buffer pool name for the index definition. You can also set the value for the primary space allocation (PRIQTY) to the required size (in KB).	
	Required	Create the PES directory table.	Submit JCL <i>CustHLQ.SFMCCNTL(FMCHJDPT)</i>	rc=0. If you get a non-zero return code, you can roll back the complete action by dropping the PES directory table space using the job FMCHJEPS. After this step you have to start again with Step number 2: Create the PES directory table space.

Table 16. Program execution directory DB2 customization (continued)

Step number	Required or optional	Description	Action	Verification
4	Required	Bind the PES directory packages and add the PES directory collection to the Workflow plan.	Submit JCL <i>CustHLQ</i> .SFMCCNTL (FMCHJBPD)	rc=0

Populate the PES directory database

You must populate the program execution server (PES) directory with the initial definition.

Table 17. Populate the PES directory

Step number	Required or optional	Description	Action	Verification
1	Required	Import the PES directory.	Submit JCL <i>CustHLQ</i> .SFMCCNTL (FMCHJPIC) Note: For subsequent executions of this step, use FMCHJPIC	rc=0
2	Required	Run the DB2 utility RUNSTATS.	Submit JCL <i>CustHLQ</i> .SFMCCNTL (FMCHJSPD)	rc=0
3	Required	Rebind the PES directory packages.	Submit JCL <i>CustHLQ</i> .SFMCCNTL (FMCHJBPD)	rc=0
4	Required	Verify the PES directory database configuration.	Submit JCL <i>CustHLQ</i> .SFMCCNTL (FMCHJCPD)	Verify that the SELECT statement returns at least one row of data.

Program execution server mapping DB2 customization

This customization creates the PES mapping database that is used by the default program mapper. If you do not want to invoke any legacy applications that would require program mapping, you can skip this, and continue customization at “MQSeries customization” on page 29.

Table 18. Program execution server mapping DB2 customization

Step number	Required or optional	Description	Action	Verification
1	Optional	If you want to change the default buffer pool name, or storage group for the database:	Edit <i>CustHLQ</i> .SFMCCDB2 (FMCHDDMD), and change the buffer pool names.	
	Required	Create the PES mapping database.	Submit JCL <i>CustHLQ</i> .SFMCCNTL (FMCHJDMD)	rc=0

Table 18. Program execution server mapping DB2 customization (continued)

Step number	Required or optional	Description	Action	Verification
2	Optional	If you want to change the buffer pool names to be used for the table space, or if you want to change the value for the primary space allocation:	Edit <i>CustHLQ.SFMCDB2</i> (FMCHDDMS), and change the buffer pool names to be used for the table space. You can also set the value for the primary space allocation (PRIQTY) to the required size (in KB).	
	Required	Create the PES mapping table space.	Submit JCL <i>CustHLQ.SFMCNTL</i> (FMCHJDMS)	rc=0. If you get a non-zero return code, you can roll back the complete action by dropping the PES mapping database using the job FMCHJEMD. After this step you have to start again with <i>Step number 1: Create the PES mapping database</i> .
3	Optional	If you want to change the buffer pools, or the primary space allocation for the index:	Edit <i>CustHLQ.SFMCDB2</i> (FMCHDDMT), and change the buffer pool name for the index definition. You can also set the value for the primary space allocation (PRIQTY) to the required size (in KB).	
	Required	Create the PES mapping tables.	Submit JCL <i>CustHLQ.SFMCNTL</i> (FMCHJDMT)	rc=0. If you get a non-zero return code, you can roll back the complete action by dropping the PES mapping table spaces using the job FMCHJEMS. After this step you have to start again with <i>Step number 2: Create the PES mapping space</i> .
4	Required	Bind the PES mapping packages and add the PES mapping collection to the Workflow plan.	Submit JCL <i>CustHLQ.SFMCNTL</i> (FMCHJBMA)	rc=0

MQSeries customization

This defines all the MQSeries resources required by MQSeries Workflow for OS/390. Before you perform this customization, make sure that your queue manager is started.

Table 19. MQSeries customization

Step number	Required or optional	Description	Action	Verification
1	Required	Define the MQSeries resources (except for program execution).	Submit JCL <i>CustHLQ.SFMCNTL</i> (FMCHJDMQ)	rc=0
2	Required	Define the MQSeries resources required by MQSeries Workflow for OS/390 program execution.	Submit JCL <i>CustHLQ.SFMCNTL</i> (FMCHJPMQ)	rc=0

OS/390 trace customization

One Workflow Server Trace exists for each Workflow system. As this trace uses OS/390 system components (Component Trace or CTRACE for short), you must also update system resources. You are provided with two templates for members of SYS1.PARMLIB which control the tracing of Workflow servers. You are provided with one template for a JCL procedure member of SYS1.PROCLIB which controls a CTRACE external writer. Finally, you must provide and specify in the JCL procedure, the trace output data sets for the external writer.

Table 20. OS/390 trace customization

Step number	Required or optional	Description	Action
1	Required	Look up variable values.	<p>Check Table 3 on page 11, and note the values that you planned for the following identifiers:</p> <ol style="list-style-type: none"> 1. The two alphanumeric characters <i>nm</i> for <i>CTStartSuffix</i>. 2. The two alphanumeric characters <i>mm</i> for <i>CTStopSuffix</i>. 3. The value for <i>CTWriter</i>. <p>Note: These are the values that you should have assigned to the variables <CTRCPMS1>, <CTRCPMS2>, and <CTRCWPRC> respectively, in file <i>CustHLQ.SFMCDATA(FMCHCECIF)</i></p>
2	Required	Check system libraries.	<p>Make sure that the system (or the sysplex) does not already contain the members:</p> <ol style="list-style-type: none"> 1. SYS1.PARMLIB(CTIFMC<i>nn</i>). 2. SYS1.PARMLIB(CTIFMC<i>mm</i>). 3. SYS1.PROCLIB(CT<i>Writer</i>). <p>Where <i>nm</i>, <i>mm</i>, and <i>CTWriter</i> are the values from step 1.</p>
3	Required	Copy PARMLIB members.	<ol style="list-style-type: none"> 1. Copy the Component Trace Start PARMLIB member <i>CustHLQ.GENPARM(CTIFMC<i>nn</i>)</i> to SYS1.PARMLIB 2. Copy the Component Trace Stop PARMLIB member <i>CustHLQ.GENPARM(CTIFMC<i>mm</i>)</i> to SYS1.PARMLIB
4	Required	Copy PROCLIB member	<ol style="list-style-type: none"> 1. Copy the trace writer <i>CustHLQ.GENPROC(CT<i>Writer</i>)</i> to SYS1.PROCLIB
5	Required	Create the extended trace output data sets.	<ol style="list-style-type: none"> 1. Edit data set <i>CustHLQ.SFMCCNTL(FMCHJCTR)</i> 2. Submit JCL <i>CustHLQ.SFMCCNTL(FMCHJCTR)</i>
6	Required	Provide RACF profiles.	<p>Give update access (to the trace data sets created in step 5) to the user ID that is assigned to the trace writer.</p> <p>Note: If no explicit user ID assignment is made, the trace writer runs under the user ID STCUSER and group ID STCGROUP.</p>

CICS API support customization

If you want to use the MQSeries Workflow for OS/390 API and trace in CICS, then you must perform this customization. If you only want to use CICS legacy applications, or if you do not want to use CICS at all you can skip this customization, and continue at “IMS API support customization” on page 32.

Table 21. CICS API support customization

Step number	Required or optional	Description	Action	Verification
1	Required	Enable LE and C/C++ features in CICS.	<p>If they are not already enabled:</p> <ol style="list-style-type: none"> 1. Enable LE in CICS. Note: The CSD definitions necessary to accomplish this task are located in <i>LEInstHLQ.SCEESAMP(CEECCSD)</i> 2. Enable the C/C++ feature in CICS. Note: A sample that may help you with this task is located in <i>CCPPIInstHLQ.SCLBSAM(CLB3YCSD)</i> <p>where <i>LEInstHLQ</i> and <i>CCPPIInstHLQ</i>, are your values from Table 4 on page 13.</p>	
2	Required	Specify the location of the Workflow executables, and start-up parameters.	<ol style="list-style-type: none"> 1. Edit your CICS start-up job. 2. Find the DFHRPL entry. 3. Add the MQSeries Workflow for OS/390 library called <i>InstHLQ.SFMCLOAD</i> to the DFHRPL entry. 4. Specify an EDSALIM value of at least 200M and a CICS region size that will accommodate your EDSALIM setting. For example, specify the CICS parameter <i>EDSALIM=200M</i> and <i>REGION=220M</i> in your CICS start-up job. 	
3	Required	Create user profile, machine profile, and environment data in VSAM format.	<ol style="list-style-type: none"> 1. Edit <i>CustHLQ.SFMCCNTL(FMCHJCPR)</i> 2. Change the CICSVOL value to the name of the volume where you want the profiles to be located. 3. Submit JCL <i>CustHLQ.SFMCCNTL(FMCHJCPR)</i> 	rc=0
4	Optional	If you do not want to use the value for <i>CICSGroup</i> that you specified in Table 5 on page 14:	<p>Change the group in the CSD file:</p> <ol style="list-style-type: none"> 1. Edit the CSD file <i>CustHLQ.SFMCDATA(FMCHEPRO)</i> 2. Change the GROUP values to the one(s) you intended for the Workflow executable, profiles, etc.. 	
5	Required	Update CICS CSD with file definitions for C++ and MQSeries Workflow for OS/390.	<ol style="list-style-type: none"> 1. Edit <i>CustHLQ.SFMCCNTL(FMCHJCUP)</i> 2. Change the CICSNAME value to the name of the CICS system that you are customizing. 3. Submit JCL <i>CustHLQ.SFMCCNTL(FMCHJCUP)</i> 	rc=0
6	Required	Restart CICS.	Restart CICS.	
7	Required	Make MQSeries CICS stubs available in CICS.	Make the MQSeries CICS Stubs <i>IMQB23IC</i> and <i>IMQS23IC</i> from <i>MQInstHLQ.SCSQLOAD</i> available to CICS.	
8	Required	Make the C/C++ group <i>CLB</i> and the MQSeries Workflow for OS/390 group available in CICS.	<ol style="list-style-type: none"> 1. Make the C/C++ group <i>CLB</i> available in CICS, with the command: <i>CEDA ADD G(CLB) LIST(xxx)</i> 2. Make the MQSeries Workflow for OS/390 group (<i>CICSGroup</i>, unless you changed it in step 4) available in CICS, with the command: <i>CEDA ADD G(yyy) LIST(yyy)</i> <p>where <i>xxx</i> is a LIST used at CICS start-up, and <i>yyy</i> is the MQSeries Workflow for OS/390 group.</p>	

Table 21. CICS API support customization (continued)

Step number	Required or optional	Description	Action	Verification
9	Required	Stop and restart CICS.	Stop and restart CICS.	
10	Required	Verify profile access.	<ol style="list-style-type: none"> 1. Logon to CICS. 2. Perform: CEMT I FI(FMCHEUPR) 3. One file should be displayed. Try to open the file by typing "OPE" over "CLO" (and pressing enter). If this works without resulting in an error message, the profile access has been established. If you get an error message, retry the previous steps for enabling CICS API support. If this does not help, contact your IBM representative. 4. You can now close the file again by typing "CLO" over "OPE" Since CICS will then disable the file, type "ENA" over "UNE" (UNEnabled). 	

IMS API support customization

This makes MQSeries Workflow for OS/390 DLLs available to IMS so that programs using the MQSeries Workflow for OS/390 container API can be executed in IMS. If you only want to use IMS legacy applications, or if you do not want to use IMS at all you can skip this customization, and continue at "Workflow server customization".

Table 22. IMS API support customization

Step number	Required or optional	Description	Action
1	Required	Provide load modules for IMS	Add all members with the prefix "FMCH3" from the library <i>InstHLQ.SFMCLoad</i> library to your IMS PGMLIB library.

Workflow server customization

To enable a Workflow server, a JCL procedure has to be provided in SYS1.PROCLIB. For each Workflow system, you must copy a template into SYS1.PROCLIB, and then customize it. This is necessary to start the MQSeries Workflow for OS/390 servers as a started task.

Table 23. Workflow server customization

Step number	Required or optional	Description	Action	Verification
1	Required	Copy definitions for Workflow servers into a procedure library.	Copy the JCL procedure <i>CustHLQ.GENPROC(UniqueSystemKey)</i> to SYS1.PROCLIB, where <i>UniqueSystemKey</i> is your value in Table 3 on page 11.	
2	Required	Assign RACF user ID and group to the Workflow server started task. Note: This is the <i>ServerUserID</i> , see Table 3 on page 11	Submit JCL <i>CustHLQ.SFMCCNTL(FMCHJDSC)</i>	rc=0
3	Required	Provide RACF profile.	Give the <i>ServerUserID</i> assigned in Step 2 read access to the data set <i>CustHLQ.SFMCDATA</i> .	

Table 23. Workflow server customization (continued)

Step number	Required or optional	Description	Action	Verification
4	Required if you want more than eight server instances per address space, or if you want console messages to be in uppercase, otherwise optional	Modify the Workflow server start job definitions.	<ol style="list-style-type: none"> 1. Edit SYS1.PROCLIB(<i>UniqueSystemKey</i>). 2. If you want, you can extend the definitions to allow more than eight server instances per address space. 3. If you want console messages from the server address space to be in uppercase, change the value for LANGC to ENP (the default value is ENU). 4. If you want, you can modify the DD statements for the stdout and stderr output, and the simple trace output. 5. If you want, you can modify the sysout class for stdout, stderr, and simple trace output. 	
5	Required only if you want servers and tools to give MMS messages in uppercase	Modify the language setting in the machine profile.	Edit the machine profile <i>CustHLQ.SFMCDATA(FMCHEMPLR)</i> , and change the Language setting to ENP for uppercase U.S. English. The default value is ENU (mixed-case U.S. English). Note: These messages are generally routed to SYSOUT data sets	
6	Required	Grant the server user ID execute access to the database plan.	Issue the command: GRANT EXECUTE ON PLAN <i>DB2Plan</i> TO <i>ServerUserID</i> using your values for <i>DB2Plan</i> in Table 2 on page 10, and <i>ServerUserID</i> in Table 3 on page 11.	

LAN client customization

This task describes how to configure a MQSeries Workflow LAN client to connect to a MQSeries Workflow for OS/390 server. This task consists of two parts:

1. "Customize the MQSeries client connection"
2. "Customize the MQSeries Workflow client" on page 34

Note: It is very important that you check the files called *Readme.1st* and *Readme.xxx* (where *xxx* is your language code) on the *MQSeries Workflow Version 3.2.1 CD*.

Customize the MQSeries client connection

To set up an MQSeries client connection you must do the following:

Table 24. Customize the MQSeries client connection

Step number	Required or optional	Description	Action
1	Required	Install MQSeries client.	Install an MQSeries client from the MQSeries CD as described in the MQSeries Workflow product documentation.

Table 24. Customize the MQSeries client connection (continued)

Step number	Required or optional	Description	Action
2	Required	Generate a channel tab file.	<ol style="list-style-type: none"> 1. Edit <i>CustHLQ.SFMCCNTL(FMCHJCCC)</i> and replace the <volume> parameter with the DASD name where you want to have your generated channel tab stored. 2. Decide which client connection channel definitions you want to have to be created within the channel tab: (a) Only for the current Workflow system or (b) Also for further systems you want to connect to using the channel tab which will be created within this step. <ol style="list-style-type: none"> a. For the current Workflow system: Submit the JCL <i>CustHLQ.SFMCCNTL(FMCHJCCC)</i>, and expect rc=0. b. For multiple Workflow systems: <ol style="list-style-type: none"> 1) Edit the JCL <i>CustHLQ.SFMCCNTL(FMCHJCCC)</i> and concatenate all client connection definitions to the CSQUCMD DD statement in step FMCHJCC1. One way to do this is by inserting all required definitions into the already included file <i>CustHLQ.SFMCMQS(FMCHNCCC)</i>, by the concatenation of FMCHNCCC files of multiple systems (if you have them all on shared DASD). Another way to do this is to concatenate a file of your own to the collected client channel definitions. 2) Submit the JCL <i>CustHLQ.SFMCCNTL(FMCHJCCC)</i>, and expect rc=0.

For more information about MQSeries client connection, see the MQSeries documentation *MQSeries Clients*. Now your MQSeries client connection is defined; you are ready to customize the MQSeries Workflow client.

Customize the MQSeries Workflow client

To set up an MQSeries Workflow client you must do the following:

Table 25. Customize the MQSeries Workflow client

Step number	Required or optional	Description	Action
1	Required	Install an MQSeries Workflow client.	Install an MQSeries Workflowclient from the <i>MQSeries Workflow Version 3.2.1 CD</i> as described in <i>IBM MQSeries Workflow: Installation Guide</i> .
2	Required	Download channel tab file.	Download the file <i>CustHLQ.mqwfchl.tab</i> file binary from OS/390 to a directory on the workstation where you want to run your client.
3	Required	Configure MQSeries Workflow client.	Configure the client using the MQSeries Workflow customization tool, as described in the documentation.

Now you have customized the MQSeries Workflow client.

Customize Java-API support

This customization step is optional, if you do not do it now, but realize later that you need Java support, you can perform this customization later. If you do not want MQSeries Workflow tasks to be able to invoke the Java virtual machine

(JavaVM) residing on OS/390 Unix System Services (USS), you can skip this customization, and continue at “System customization verification” on page 36.

To set up an MQSeries Workflow client you must do the following:

Table 26. Customize the MQSeries Workflow client

Step number	Required or optional	Description	Action
1	Required	Edit the copy job	<p>Edit the copy job <i>InstHLQ</i>.SFMCCNTL(FMCHJHFS)</p> <ol style="list-style-type: none"> 1. If you want to use the Java API, then change: <ol style="list-style-type: none"> a. <HFSJAR> to the Java directory for storing jar files otherwise comment the line out with '/*'. 2. If you want to copy all classes, Java, and html files that you will need later for servlet customization (described in “Chapter 11. Administering Servlets on the WebSphere Application Server” on page 117) then change: <ol style="list-style-type: none"> a. <HFSCLASS> to the Java directory for storing class files b. <HFSHTML> to the Java directory for storing html and Java files otherwise comment these lines out with '/*'.
2	Required	Copy the Java-Agent	<p>Submit the edited copy job <i>InstHLQ</i>.SFMCCNTL(FMCHJHFS). This copies the Java-Agent jar file <i>fmchgagt.jar</i> from your SFMCJIF – partitioned data set to the Hierarchical File System (HFS) directories specified in step 1a.</p>
3	Required	Create two external links in your HFS Java directory.	<p>To create two links that point to the OS/390 partitioned data set which includes the two DLLs for the JNI-layer, issue the following commands:</p> <pre>ln -e FMCH1LOC libfmcojloc.so ln -e FMCH1PRF libfmcojprf.so</pre>
4	Required	Set the environment variables.	<ol style="list-style-type: none"> 1. In your HFS .profile file: <ol style="list-style-type: none"> a. Set the environment variable CLASSPATH to your HFS directory where you copied the Java agent jar file (see <HFSJAR> in step 1a). b. Set the environment variable LIBPATH to the directory where you put in the external links in step 3. c. Add the statements from the MQSeries Workflow for OS/390 environment variables file <i>CustHLQ</i>.SFMCDATA(FMCHEENV). 2. Run your profile with the command: <pre>./profile</pre>
5	Required	Decide whether to call JavaVM via LNKLST or STEPLIB definitions.	<p>If you want to call JavaVM via a LNKLST then perform step 6a, otherwise perform step 6b.</p>

Table 26. Customize the MQSeries Workflow client (continued)

Step number	Required or optional	Description	Action
6a	One of 6a and 6b is required	Add JNI-Layer and MQSeries data sets to LNKLST.	<p>Add data sets to the linklist LNKLST:</p> <ol style="list-style-type: none"> 1. for the JNI-Layer data sets: <ol style="list-style-type: none"> a. <i>CustHLQ</i>.SFMCLOAD(FMCOJLOC) b. <i>CustHLQ</i>.SFMCLOAD(FMCOJPRF) 2. and for the data sets where MQSeries is located: <ol style="list-style-type: none"> a. <i>MQInstHLQ</i>.SCSQLOAD b. <i>MQInstHLQ</i>.SCSQAUTH <p>where <i>CustHLQ</i> and <i>MQInstHLQ</i> are your values from “Installation scope identifiers” on page 9 and “Flags and high level qualifiers” on page 13 respectively.</p> <p>Note: If you want to change the link list dynamically using the set command, you can create a member PROGxy in parmlib, which includes the following statements:</p> <pre>LNKLST DEFINE NAME(FMCJNI1) COPYFROM(CURRENT) LNKLST ADD NAME(FMCJNI1) DSNAME(<i>CustHLQ</i>.SFMCLOAD) LNKLST ADD NAME(FMCJNI1) DSNAME(<i>MQInstHLQ</i>.SCSQLOAD) LNKLST ADD NAME(FMCJNI1) DSNAME(<i>MQInstHLQ</i>.SCSQAUTH) LNKLST ACTIVATE NAME(FMCJNI1)</pre> <p>then activate this member with the console command:</p> <pre>/set prog=<xy></pre>
6b		Add JNI-Layer and MQSeries data sets to STEPLIB.	<p>Set the environment variable STEPLIB to the JNI-Layer and MQSeries data sets using the command:</p> <pre>EXPORT STEPLIB=<i>CustHLQ</i>.SFMCLOAD:<i>MQInstHLQ</i>.SCSQLOAD: <i>MQInstHLQ</i>.SCSQAUTH</pre> <p>where <i>CustHLQ</i> and <i>MQInstHLQ</i> are your values from “Installation scope identifiers” on page 9 and “Flags and high level qualifiers” on page 13 respectively.</p>
7	Required	Copy the user and a machine profiles to enable Workflow log on.	<p>Copy the two profiles <i>CustHLQ</i>.SFMCDATA(FMCHECPR) and <i>CustHLQ</i>.SFMCDATA(FMCHEUPR) to the HFS-directory from where you call the JAVA-program. The names for these profiles have to be:</p> <pre>DD:FMCEMPR DD:FMCEUPR</pre>

Now you have customized the Java-API support.

System customization verification

This verification tests if the client system can connect to MQSeries Workflow for OS/390 by logging on as ADMIN, a predefined, and always available user ID.

Table 27. System customization verification

Step number	Required or optional	Description	Action	Verification
1	Required	Start the OS/390 administration and execution servers.	On the OS/390 system console, issue the commands: <code>START UniqueSystemKey.AdminServerID</code> <code>MODIFY AdminServerID,START EXE</code> where <i>UniqueSystemKey</i> is your value specified in Table 3 on page 11, and <i>AdminServerID</i> is a made-up name used to identify the administration server for the system identified by <i>UniqueSystemKey</i> .	You should get the system console messages FMC19027I Administration server successfully started. and FMC10200I Execution server for system <i>SystemName</i> started.
2	Required	Start the runtime client.	Double-click on the runtime client icon.	You are prompted for a user ID and password.
3	Required	Logon.	Logon using the user ID ADMIN, and the password "password".	If no error message is displayed, the verification is complete.
4	Required	Logoff.	Logoff the runtime client.	
5	Required	Stop the OS/390 execution and administration servers.	On the OS/390 system console, issue the command <code>MODIFY AdminServerID,STOP EXE</code> <code>MODIFY AdminServerID,STOP ADM</code> where <i>AdminServerID</i> is the name you used when starting the server.	

Verify Workflow client sample application

This customization verification stage is optional, if you wish, you can skip to "Program execution customization" on page 39.

This uses a sample application to verify that the workstation client can work with MQSeries Workflow for OS/390 on the host system. Using the client, you should be able to query templates, instances, and work lists. In addition, you should be able to instantiate templates, start instances and start work items.

Table 28. Verify Workflow client sample application

Step number	Required or optional	Description	Action	Verification
1	Required	Start the OS/390 administration server.	On the OS/390 system console, issue the command <code>START UniqueSystemKey.AdminServerID</code> where <i>UniqueSystemKey</i> is your value specified in Table 3 on page 11, and <i>AdminServerID</i> is a made-up name used to identify the administration server.	You should get the system console message FMC19027I Administration server successfully started.
2	Required	Import the Workflow sample process model.	Submit JCL <code>CustHLQ.SFMCCNTL (FMCHJFDL)</code>	rc=0

Table 28. Verify Workflow client sample application (continued)

Step number	Required or optional	Description	Action	Verification
3	Required	Start the OS/390 execution server.	On the OS/390 system console, issue the command: MODIFY <i>AdminServerID</i> ,START EXE where <i>AdminServerID</i> is the name you used in step 1, when starting the server.	You should get the system console message FMC10200I Execution server for system <i>SystemName</i> started.
4	Required	Start the runtime client.	Double-click on the runtime client icon.	You are prompted for the client's user ID and password.
5	Required	Logon.	Logon using the user ID ADMIN, and the password "password".	
6	Required	Start a new process instance.	1. Look up your process templates. 2. Create an instance of the process template named Edit. 3. Refresh the instance window. Note: For more information about using the runtime client, refer to <i>IBM MQSeries Workflow: Getting Started with Runtime</i> .	
7	Required	Start a new workitem.	1. Start the process instance. 2. Refresh the workitem window.	An editor should be displayed. A new workitem named Edit_Activity should appear in the workitem window.
8	Required	Terminate the workitem.	1. Close the editor. 2. Refresh the instance and workitem windows.	The workitem named Edit_Activity is finished, and the instance has terminated.
9	Required	Logoff.	Log off the runtime client.	
10	Required	Stop the execution server.	On the OS/390 system console, issue the command: MODIFY <i>AdminServerID</i> ,STOP EXE	
11	Required	Stop the OS/390 administration server.	On the OS/390 system console, issue the command: MODIFY <i>AdminServerID</i> ,STOP ADM	

Program execution customization

This customization is optional, depending on your program execution requirements:

Table 29. Customizing program execution invocation types

Program type	Invocation type	Customization required
CICS	EXCI	"Customize CICS EXCI invocation"
	MQSeries CICS Bridge	"Customize MQSeries CICS bridge invocation" on page 41
IMS	CPIC	"Customize IMS CPIC invocation" on page 44
	MQSeries IMS Bridge	"Customize MQSeries IMS bridge invocation" on page 46

If you want to be able to invoke CICS and/or IMS programs, then it is recommended that you also perform:

1. "Customize program execution server directory" on page 47
2. "Configure program execution samples" on page 49
3. "Verify program execution samples" on page 50

The following manuals may help you customize program execution:

- *OS/390 MVS Programming: Resource Recovery*
- *CICS for OS/390: CICS Resource Definition guide*
- *CICS for OS/390: CICS RACF Security Guide*
- *CICS for OS/390: CICS Internet and External Interfaces Guide*
- *CICS for MVS/ESA External CICS Interfaces*
- *MQSeries for OS/390 Version 2.1 System Management Guide*

Customize CICS EXCI invocation

The program execution server supports EXCI invocations of OS/390 programs as part of a process activity. You only need to perform this customization if you want the program execution server to be able to invoke CICS programs using the EXCI invocation:

Table 30. Customize CICS EXCI invocation

Step number	Required or optional	Description	Action	Verification
1	Required	Define group resources CONNECTION and SESSIONS.	<p>Decide or identify the name of the group (<i>GroupName</i>) that will contain the resource definition <i>ConnectionName</i> for the connection and sessions. In that group:</p> <ol style="list-style-type: none"> Create a generic EXCI CONNECTION resource definition with the following values: <ol style="list-style-type: none"> ACCESSMETHOD=IRC PROTOCOL=EXCI CONNTYPE=GENERIC If user security checking is required (user IDs are checked, but no passwords are required), then specify ATTACHSEC=IDENTIFY. Note: This option makes step 3 and step 4 required. If no user security checking is required specify ATTACHSEC=LOCAL so that the invoked applications will be run under the default user ID of the target CICS. Note: This option makes step 3 and step 4 unnecessary. Define a SESSIONS resource definition with the following values <ol style="list-style-type: none"> CONNECTION=<i>ConnectionName</i> PROTOCOL=EXCI RECEIVEPFX=RC RECEIVECOUNT=4 	
2	Required	Make sure IRC is started.	If IRC is not already started, issue the CICS command: CEMT SET IRC OPEN	Verify that the IRC status is OPEN with the CICS command: CEMT I IRC
	Optional	Add IRC start to CICS start job.	You can add the CICS parameter IRCSTRT=YES in the CICS start job so that IRC is opened when CICS is started. If you do not add this parameter, you will have to open IRC on the CICS system manually each time CICS is restarted. This is done using the CICS command: CEMT SET IRC OPEN.	
3	Required only if security checks are required	Define the RACF profiles required to give the program execution server authority to run CICS applications using EXCI calls.	<ol style="list-style-type: none"> Identify the RACF user ID of the PES (see <i>ServerUserID</i> in Table 3 on page 11), the ID of the of the CICS server region (see <i>applid</i> in Table 6 on page 14), and the names of the RACF profiles for resources used by CICS application server programs that are to be executed by the PES using EXCI. Define RACF FACILITY class profile DFHAPPL.<i>applid</i> with universal access NONE Give the <i>ServerUserID</i> and all users READ access to the RACF FACILITY class profile DFHAPPL.<i>applid</i>. Give <i>ServerUserID</i> and all users READ access to the RACF profiles for the transaction CSMI on the target CICS. Note: EXCI uses the CICS transaction CSMI to run requested CICS programs. 	
4	Required only if security checks are required	Enable user IDs.	If you set ATTACHSEC =IDENTIFY in step number 1.1.d above then the invoked applications will be run using the RACF user ID <i>userid</i> of the MQSeries Workflow user making the request. It is therefore necessary to give these <i>userids</i> the appropriate authority to RACF profiles of all resources accessed by CICS application server programs that the users can cause to be invoked by the program execution server. If no <i>userid</i> is passed, the invoked application will run under the PES user ID. In this case you must give the corresponding authorization for the CICS resources to the <i>ServerUserID</i> in Table 3 on page 11.	

Customize MQSeries CICS bridge invocation

The program execution server supports MQSeries CICS bridge invocations of OS/390 programs as part of a process activity. For more information, see *MQSeries for MVS/ESA System Management Guide - Customize the CICS bridge*. You only need to perform the following customization if you want the program execution server to be able to invoke CICS programs using MQSeries CICS bridge invocation:

Table 31. Customize MQSeries CICS bridge invocation

Step number	Required or optional	Description	Action
1	Required	Prepare CICS to run the CICS bridge.	<ol style="list-style-type: none"> 1. Make sure that the MQSeries CICS adapter is set up and customized on CICS. For more information see <i>MQSeries for OS/390 Version 2.1 System Management Guide — MQSeries CICS adapter</i>. 2. Define CICS bridge transactions and programs by running the resource definition utility DFHCSDUP on the CICS system using the sample <i>MQInstHLQ.SCSQPROC(CSQ4CKBC)</i>. 3. Add group CSQCKB to startup group list on the CICS.
2	Required	Define the MQSeries queue for the request messages to the CICS bridge.	<ol style="list-style-type: none"> 1. Define a local MQSeries queue <i>CICSBridgeInputQueue</i> (see your value in Table 6 on page 14) with attributes: <ol style="list-style-type: none"> a. SHARE b. MSGDLVSQ(FIFO) c. DEFPERSIST(YES) d. HARDENBO
3	Required only if security checks are required.	Define user ID under which the CICS bridge (monitor) is started as a surrogate user ID of all RACF user IDs for which program execution requests to the CICS bridge should be issued	<ol style="list-style-type: none"> 1. For each MQSeries Workflow user ID <i>mqwf_uid</i> define a profile named <i>mqwf_uid.DFHSTART</i> in the RACF SURROGATE class without any general access rights using the RACF command: RDEFINE SURROGAT <i>mqwf_uid.DFHSTART</i> UACC(NONE) OWNER(<i>mqwf_uid</i>) 2. Give READ access to surrogate_id to all of the above define profiles by issuing for each <i>mqwf_uid</i> using the RACF command: PERMIT <i>mqwf_uid.DFHSTART</i> CLASS(SURROGAT) ID(<i>surrogate_id</i>) ACCESS(READ) <p>where</p> <p><i>surrogate_id</i> Name of the user ID to be defined as surrogate user ID of all RACF user IDs to be allowed to run CICS bridge invocations. This must be the user ID under which the CICS bridge (monitor task) is started.</p> <p><i>mqwf_uid</i> RACF user IDs of all MQSeries Workflow users to be allowed to run CICS bridge invocations.</p> <p>Note: Set the CICS startup parameter XUSER=YES to enable surrogate user checking.</p>

Table 31. Customize MQSeries CICS bridge invocation (continued)

Step number	Required or optional	Description	Action
4	Required only if security checks are required.	Give access rights to request queue and reply queues used by the CICS bridge and the dead-letter queue.	<ol style="list-style-type: none"> 1. Give READ access to the CICS bridge request queue to the user ID of the CICS bridge monitor (the user ID under which the CICS bridge is started) and to all RACF user IDs for which CICS bridge request should be issued (RACF user IDs corresponding to MQSeries Workflow user IDs). 2. Give WRITE access to the CICS bridge request queue to the <i>ServerUserID</i> (see your value in Table 3 on page 11). 3. Give READ access to the CICS bridge reply to queue(s) to the <i>ServerUserID</i>. 4. Give WRITE access to each reply queue to those RACF user IDs for which the bridge should put reply messages on that queue. 5. Give WRITE access to all reply to queues to the user ID of the CICS bridge monitor. 6. Give WRITE access to the dead-letter queue to all RACF user IDs for which requests should be issued and to the user ID of the CICS bridge monitor.
5	Required only if the target CICS requires security checks for RACF user IDs of MQSeries Workflow users.	Define RACF authority access to CICS programs to RACF user IDs of MQSeries Workflow users.	<ol style="list-style-type: none"> 1. Identify the RACF user IDs <i>mqwf_userid</i> of MQSeries Workflow users who should be allowed to run applications on the target CICS using MQSeries bridge invocation. 2. Identify the names of RACF profiles for resources that are used by CICS application server programs using the MQSeries CICS bridge invocation. 3. For each <i>mqwf_userid</i> define appropriate authority to RACF profiles for all resources accessed by the CICS application server programs that the program execution request should process invocation requests.
6	Required only if the CICS bridge should run with an authentication level of LOCAL.	Define RACF authority access to CICS programs to CICS default user id.	<ol style="list-style-type: none"> 1. For the CICS default user ID (CICS DFLTUSER) define appropriate authority to RACF profiles of all resources accessed by the CICS application server programs for which program execution request should be processed.

Table 31. Customize MQSeries CICS bridge invocation (continued)

Step number	Required or optional	Description	Action
7	Required	Start the CICS bridge transaction CKBR on the CICS with authority LOCAL (the default) to run with authority associated to the CICS default user id (CICS DFLTUSER) or with authority IDENTIFY if the RACF user IDs but no passwords should be checked.	<ol style="list-style-type: none"> 1. Start the CICS bridge with the CICS command: CKBR Q=<i>InputQueue</i> AUTH=LOCAL <li style="text-align: center;">or CKBR Q=<i>InputQueue</i> AUTH=IDENTIFY <p>where <i>InputQueue</i> is your value for <i>CICSBridgeInputQueue</i> in Table 6 on page 14.</p> <ol style="list-style-type: none"> 2. See <i>MQSeries for OS/390 System Management Guide - Starting the CICS bridge</i> for further information and other ways how to start the CICS bridge. The user ID under which the CICS bridge is started is the user ID of the CICS bridge monitor. <p>Note: Since MQSeries Workflow for OS/390 does not support passwords the authority levels, VERIFY_UOW and VERIFY_ALL are not supported by CICS bridge invocations.</p>

The CICS bridge is now ready to process request messages. If the authorization level is LOCAL the CICS bridge is running with the authority of the CICS default user ID. If the authorization level is IDENTIFY a corresponding CICS program started by the bridge will be run with the user ID as specified in the MQMD header of the request message. There is no password checking.

Customize IMS CPIC invocation

The program execution server supports IMS invocations of OS/390 programs as part of a process activity. You only need to perform this customization if you want the program execution server to be able to invoke IMS programs using the CPIC invocation. For more information about APPC, refer to *OS/390 MVS Planning: APPC/MVS Management*.

Table 32. Customize IMS CPIC invocation

Step number	Required or optional	Description	Action
1	Required	Define a system base LU for CPIC requests to use for APPC conversations.	<p>Since the CPIC call cannot specify a dedicated LU from which a conversation should be allocated, it is necessary to use a system LU as the default local LU. This step is performed as follows:</p> <ol style="list-style-type: none"> 1. If the APPCPMxx PARMLIB member contains LUADD statements with BASE and NOSCHED, the last one of these defines the system base LU. If a new LU has to be defined as base LU another LUADD statement has to be added to the end of the APPCPMxx member. 2. If there are no LUADD statements with parameter NOSCHED, but some with parameter BASE, the last one of these LUADDs defines the system base LU as long as it is associated with the APPC/MVS transaction scheduler explicitly with SCHED(ASCH) or implicitly, without a SCHED parameter. Add a new LUADD statement defining the new base LU to the end of the APPCPMxx member, if you do not want to use the current one. 3. If there is no base LU defined at all, define one by adding a LUADD statement with parameters BASE and optionally, with NOSCHED or SCHED(ASCH).
2	Required	If the system base LU defined in step 1 is associated with the APPC/MVS transaction scheduler (ASCH):	<p>Make sure that there is a PARMLIB member ASCHPMxx available defining the scheduling characteristics of the ASCH, as described below:</p> <ol style="list-style-type: none"> 1. If there is already an ASCHPMxx member in the SYS1.PARMLIB this can be used to run the ASCH address space. 2. If there is no ASCHPMxx PARMLIB member you must create one. For details on how to create one, see <i>OS/390 MVS Planning: APPC/MVS Management</i>, "Defining Scheduling Characteristics with ASCHPMxx." <p>There are no CPIC invocation specific definitions needed in ASCHPMxx.</p>
3	Required	Define an APPC LU associated with the target IMS system (service).	<p>This is the partner LU for a CPIC invocation issuing a request to be performed on that target IMS. The IMS ID is passed as scheduler name for this LU.</p> <ol style="list-style-type: none"> 1. Put LUADD statement to APPCPMxx PARMLIB member with following parameters: <ol style="list-style-type: none"> a. ACBNAME(<ims_lu>) — where <i>ims_lu</i> is the name of the APPC LU to be associated with the target IMS. b. BASE c. SCHED(<ims_id>) — where <i>ims_id</i> is the (1-4 character) ID of the target IMS as defined in the IMSCTRL installation macro. <p>Now the target IMS system is defined as APPC component LU and CPIC invocations can issue requests to the IMS using the LU defined here as the partner LU.</p>

Table 32. Customize IMS CPIC invocation (continued)

Step number	Required or optional	Description	Action
4	Required	Define a system base APPC LU for VTAM that is enabled for protected conversation support.	<p>1. In a member of SYS1.VTAMLST define an APPL statement for the above defined system base LU with:</p> <ul style="list-style-type: none"> a. ACBNAME=<base_lu> — where <i>base_lu</i> is the name of the system base LU. b. APPC=YES c. ATNLOSS=ALL d. SYNCLVL=SYNCPT e. VERIFY=NONE or VERIFY=OPTIONAL <p>Note: For more information about VTAM definitions, see <i>VTAM Resource Definition Guide</i>.</p> <p>Now the system base LU is defined in VTAM and supports distributed syncpoint conversations.</p>
5	Required	Define APPC LU of target IMS to VTAM enabled for protected conversation support.	<p>1. In a member of SYS1.VTAMLST define an APPL statement for the target IMS LU defined in step 3 with:</p> <ul style="list-style-type: none"> a. ACBNAME=<ims_lu> — where <i>ims_lu</i> is the name of the APPC LU to be associated with the target IMS. b. APPC=YES c. ATNLOSS=ALL d. SYNCLVL=SYNCPT e. If no security checks are required SECACPT=NONE <p>Note: If no security checks are required this LU will not accept conversations with any security information, that means the security information from a CPIC allocate request is not passed to this LU.</p> <ul style="list-style-type: none"> f. If security checks are required — SECACPT=ALREADYV or SECACPT=AVPV <p>Note: If security checks are required this LU will accept conversations with a user ID that is indicated as having already been verified, since no password is passed on the CPIC invocation.</p> <ul style="list-style-type: none"> g. VERIFY=NONE or VERIFY=OPTIONAL <p>The target IMS LU has now been defined in VTAM and supports distributed syncpoint conversations.</p>
6	Required	If security checks are required, you may decide to prohibit general access to the LU of the target IMS, and grant access to the RACF user IDs representing MQSeries Workflow users who are to be allowed to run transactions on the target IMS:	<ul style="list-style-type: none"> 1. Prohibit general access to LU of target IMS by defining a RACF profile using the command RDEFINE APPL <ims_lu> UACC(NONE) — where <i>ims_lu</i> is the name of the APPC LU to be associated with the target IMS. 2. Give READ access to MQSeries Workflow for OS/390 RACF user IDs by issuing the RACF command PERMIT <ims_lu> CLASS(APPL) ID(<user_id>) ACCESS(READ) repeatedly for each user ID or a RACF group ID. 3. Activate the above made definitions by issuing the RACF command SETROPTS CLASSACT(APPL) RACLIST(APPL) 4. To activate the definitions, issue the RACF command SETROPTS RACLIST(APPL) REFRESH <p>Note: For more information about using RACF, see <i>OS/390 Security Server (RACF) Command Language Reference</i>. The target IMS LU (the target IMS via APPC) is now only accessible for MQSeries Workflow users who should be allowed to run transactions on that system.</p>

Table 32. Customize IMS CPIC invocation (continued)

Step number	Required or optional	Description	Action
7	Required	Make APPC/MVS ready to work with the previously defined APPC LUs.	<ol style="list-style-type: none"> 1. If APPC/MVS is not running, start the APPC/MVS address space by issuing the command <code>START APPC,SUB=MSTR,APPC=xx</code> — where <i>xx</i> is the suffix in the name of the PARMLIB member <code>APPCPMxx</code> containing the definitions of the APPC LUs made in step number 3. 2. If the system base LU is associated to the APPC/MVS transaction scheduler (ASCH) start the ASCH address space by issuing the command <code>START ASCH,SUB=MSTR,ASCH=xx</code> where <i>xx</i> is the suffix in the name of the PARMLIB member <code>ASCHPMxx</code> containing the configuration of the ASCH. 3. If APPC/MVS is already running and the changes in <code>APPCPMxx</code> must be activated, issue the command <code>SET APPC=xx</code>. For more information, see <i>OS/390 MVS Planning: APPC/MVS Management</i>, "Starting the APPC and ASCH Address Spaces." <p>The APPC/MVS is now ready to send out-bound requests from a CPIC invocation to the target IMS system as specified by the connection parameters for CPIC invocations.</p>
8	Required	Make APPC/IMS LU ready to receive inbound requests from CPIC invocations.	<ol style="list-style-type: none"> 1. Start APPC/IMS on the target IMS by issuing the command <code>/START APPC</code> 2. If security checking is required, change the APPC/IMS security level to allow user ID checking by issuing the command <code>/SECURE APPC CHECK</code> — then the user ID from an inbound request will be checked using the RACF resource class TIMS. 3. If no security checking is required on IMS for inbound requests, switch off APPC/IMS security checking by issuing the command <code>/SECURE APPC NONE</code> <p>The target IMS is now ready to call transactions requested by inbound requests from CPIC invocations via its APPC LU.</p>

Customize MQSeries IMS bridge invocation

The program execution server supports IMS invocations of OS/390 programs as part of a process activity. You only need to perform this customization if you want the program execution server to be able to invoke IMS programs using the MQSeries IMS bridge.

Table 33. Customize MQSeries IMS bridge invocation

Step number	Required or optional	Description	Action
1	Required	Define parameters for MQSeries.	<ol style="list-style-type: none"> 1. Define <code>XCFGroupName</code> and <code>XCFMemberMQ</code> (see your values in Table 6 on page 14) where the member name represents the MQSeries instance by OTMACON keyword of the CSQ6SYSP macro. Note: The MQSeries instance and the target IMS system must belong to the same XCF group. <p>For more information, see the <i>MQSeries for OS/390 System Management Guide</i>.</p>

Table 33. Customize MQSeries IMS bridge invocation (continued)

Step number	Required or optional	Description	Action
2	Required	Define parameters for IMS.	<p>In the IMS parameter list</p> <ol style="list-style-type: none"> 1. Define the <i>XCFGroupName</i> using the GRNAME parameter. Note: The MQSeries instance and the target IMS system must belong to the same XCF group. 2. Define the <i>XCFMemberIMS</i> of the IMS system using the USERVAR parameter. 3. Define OTMA = Y in the IMS parameter list so that OTMA (and the IMS bridge) are started automatically when IMS is started. <p>For more information, see the <i>MQSeries for OS/390 System Management Guide</i>.</p>
3	Required	Tell MQSeries the XCF group and member name of the IMS system.	<ol style="list-style-type: none"> 1. Define an MQSeries storage class with the XCF group that MQSeries and the target IMS belong to (see <i>XCFGroupName</i> in Table 6 on page 14), and with the XCF member name of the IMS (see <i>XCFMemberIMS</i> in Table 6 on page 14).
4	Required	Define the MQSeries queue for the request messages to the IMS bridge.	<ol style="list-style-type: none"> 1. Define a local MQSeries queue <i>IMSBridgeInputQueue</i> (see your value in Table 6 on page 14) with the storage class defined in step 3, and the attributes: <ol style="list-style-type: none"> a. MSGDLVSQ(FIFO) b. DEFPERSIST(YES) c. HARDENBO
5	Required only if security checking is required.	Set up security levels for the MQ IMS bridge.	<p>So that user IDs will be checked, but no passwords are required:</p> <ol style="list-style-type: none"> 1. Identify the MQSeries subsystem user ID, unless the access levels for both RACF profiles are defined by the universal access fields. 2. Define RACF profile <i>IMSXCF.XCFGroupName.XCFMemberMQ</i> in the RACF FACILITY class, giving an access level of READ to the MQSeries subsystem user ID.
6	Required only if security checking is required.	Set up security levels for OTMA.	<p>So that user IDs will be checked, but no passwords are required:</p> <ol style="list-style-type: none"> 1. Define an OTMA security level of CHECK by issuing the IMS command: /SECURE OTMA CHECK 2. Define RACF profile <i>IMSXCF.XCFGroupName.XCFMemberIMS</i> in the RACF FACILITY class, giving an access level of UPDATE to the MQSeries subsystem user ID.
7	Required only if no security should be active.	Switch off OTMA security.	<ol style="list-style-type: none"> 1. Issue the command: /SECURE OTMA NONE

Customize program execution server directory

This prepares the connection information in the PES directory, and imports it into the PES directory database.

Table 34. Customize program execution server directory

Step number	Required or optional	Description	Action	Verification
1	Required	Add EXCI, CPIC, and MQSeries bridge connection parameters to the PES directory.	<p>Edit <i>CustHLQ</i>.SFMCDATA(FMCHEDTP), and substitute your values for the following connection parameters for CICS and/or IMS.</p> <ol style="list-style-type: none"> 1. If you intend to use CICS EXCI invocation, customize the EXCI invocation section in the following way: <ol style="list-style-type: none"> a. Change <i><applid></i> to the application identifier of the CICS system you want to use. 2. If you intend to use MQSeries CICS bridge invocation, customize the MQSeries CICS bridge invocation section in the following way: <ol style="list-style-type: none"> a. Change <i><queuename></i> to the name of the MQSeries CICS bridge input queue, see your value for <i>CICSBridgeInputQueue</i> in Table 6 on page 14. b. If the queue manager of the CICS system belongs to the same MQSeries cluster as your Workflow system, then remove the text <i>QUEUEMANAGER=<queuemanager></i>; 3. If you intend to use IMS CPIC invocation, customize the CPIC invocation section in the following way: <ol style="list-style-type: none"> a. Change <i><netid></i> to the APPC network identifier of the IMS system you want to use b. Change <i><luname></i> to the LU Name of the IMS system you want to use. c. If the mode name of your IMS LU is not #INTER, then change #INTER to your value. 4. If you intend to use MQSeries IMS bridge invocation, customize the MQSeries IMS bridge invocation section in the following way: <ol style="list-style-type: none"> a. Change <i><queuename></i> to the name of the MQSeries IMS bridge input queue, see your value for <i>IMSBridgeInputQueue</i> in Table 6 on page 14. b. If the queue manager of the IMS system belongs to the same MQSeries cluster as your Workflow system, then remove the text <i>QUEUEMANAGER=<queuemanager></i>; 	
2	Required	Import the PES directory.	Submit JCL <i>CustHLQ</i> .SFMCCNTL (FMCHJPIC)	rc=0. A problem at this stage may be caused by errors in FMCHEDTP, or may require that you repeat "Program execution server directory DB2 customization" on page 27.

Now you are ready to perform "Configure program execution samples" on page 49.

Configure program execution samples

This prepares the sample programs that are provided with MQSeries Workflow for OS/390. They will be used to verify program execution.

Table 35. Configure program execution samples

Step number	Required or optional	Description	Action
1	Required	Import mapping sample definitions for the legacy sample programs into the program execution mapping database.	Submit JCL <i>CustHLQ</i> .SFMCNTL (FMCHJMPR)
2	Required	Copy sample programs.	Copy the following files to your IMS PGMLIB: 1. <i>InstHLQ</i> .SFMCLOAD (FMCH3ICS) 2. <i>InstHLQ</i> .SFMCLOAD (FMCH3IMS)
3	Required	Make sample programs known to CICS/IMS.	1. Define the programs FMCH2CMT and FMCH2CCT to your CICS system using LANG(LE370) and DATALOCATION(ANY). 2. Define the programs FMCH3IMS and FMCH3ICS to your IMS system. 3. Define the transactions FMCH3IMT and FMCH3ICT to your IMS system with type TP.
4	Required	Import sample process model.	Submit JCL <i>CustHLQ</i> .SFMCNTL (FMCHJPDJ)
5	Required	Enable PES to execute sample programs.	If security is active for either CICS or IMS, then you must enable the <i>ServerUserID</i> defined in Table 3 on page 11 to execute the sample programs, and all resources required by them.

Now you are ready to perform “Verify program execution samples” on page 50.

Verify program execution samples

This is the final verification that you have configured your MQSeries Workflow for OS/390 system correctly for program execution. This task allows you to runs the sample CICS an IMS legacy programs using different invocation types. The four sample processes are:

1. **CICSMapping** starts the CICS legacy sample program FMCH2CMT using the MQSeries CICS bridge invocation type. This CICS program uses the default mapper.
2. **CICSContainer** starts the CICS sample program FMCH2CCT using the EXCI invocation type. This program uses the MQSeries Workflow container API.
3. **IMSMapping** starts the IMS legacy sample program FMCH3IMS using the CPIC invocation type. This program uses the default mapper.
4. **IMSContainer** starts the IMS sample program FMCH3ICS using the MQSeries IMS bridge invocation type. This program uses the MQSeries Workflow container API.

Table 36. Verify program execution samples

Step number	Required or optional	Description	Action	Verification
1	Required	Ensure that the necessary subsystems are running.	If necessary, start DB2, MQSeries <i>QueueManager</i> , CICS, or IMS.	
2	Required	Start the OS/390 administration server.	On the OS/390 system console, issue the command <code>START <i>UniqueSystemKey.AdminServerID</i></code> where <i>UniqueSystemKey</i> is your value specified in Table 3 on page 11, and <i>AdminServerID</i> is a made-up name used to identify the administration server.	You should get the system console message FMC19027I Administration server successfully started.
3	Required	Start the MQSeries Workflow for OS/390 system.	On the OS/390 system console, issue the command: <code>MODIFY <i>AdminServerID</i>,START</code>	
4	Required	Start the runtime client.	Double-click on the runtime client icon.	You are prompted for the client's user ID and password.
5	Required	Logon.	Logon using the user ID ADMIN, and the password "password". Note: For more information on using the runtime client, refer to <i>IBM MQSeries Workflow: Getting Started with Runtime</i> .	You will see the basic tree view with the icons labeled: <ul style="list-style-type: none"> • Process template lists • Process instance lists • Worklists If this is the first time that the client has been run, you will not see any elements.
6	Required	Create new process template list.	Create process template lists for the sample processes.	You will see the sample items: <ol style="list-style-type: none"> 1. CICSMapping 2. CICSContainer 3. IMSMapping 4. IMSContainer

Table 36. Verify program execution samples (continued)

Step number	Required or optional	Description	Action	Verification
7	Required	Create process instances.	<ol style="list-style-type: none"> 1. Create process instances for the processes you want to test. 2. Create process instance list items for the sample processes. 	<p>You will see the sample items:</p> <ol style="list-style-type: none"> 1. CICSMapping 2. CICSContainer 3. IMSMapping 4. IMSContainer
8	Required	Create workitems.	<ol style="list-style-type: none"> 1. Select all process instance items and start them. 2. Click OK for every window that appears. 3. Create new worklist. 	<p>You will see workitems for:</p> <ol style="list-style-type: none"> 1. CICSMapping_Activity 2. CICSContainer_Activity 3. IMSMapping_Activity 4. IMSContainer_Activity
9	Required	Start new workitems.	<ol style="list-style-type: none"> 1. Start new workitems <ol style="list-style-type: none"> a. CICSMapping_Activity b. CICSContainer_Activity c. IMSMapping_Activity d. IMSContainer_Activity 2. Refresh the workitem list. 	<p>After successful completion, each workitem returns "Finished".</p>
10	Optional	Check results of CICSMapping_Activity .	<p>The CEEOUT section of your CICS job should contain:</p> <pre> FMCH2CMT: MQWF Program Execution Customization LastName: Smith FirstName: John Zip: 12345 Salary: 1000.42 Tax: 15.5 Customer LastName : EINSTEIN Customer FirstName : ALBERT Customer PhoneNumber : 3048 Customer LastName : NEWTON Customer FirstName : ISAAK Customer PhoneNumber : 4041 Customer LastName : KOHL Customer FirstName : HELMUT Customer PhoneNumber : 5154 New Salary: 1080.45 </pre>	
11	Optional	Check results of CICSContainer_Activity .	<p>The CEEOUT section of your CICS job should contain:</p> <pre> FMCH2CCT: MQWF Program Execution Customization OutContainer Name = SimpleDS InContainer Name = SimpleDS Set OutputContainer long value ml rc = 0 Main: Set OutputContainer rc = 0 </pre>	

Table 36. Verify program execution samples (continued)

Step number	Required or optional	Description	Action	Verification
12	Optional	Check results of IMSMapping_Activity .	The latest SYSxxxx section of your IMS region job should contain: FMCH3IMS: MQWF Program Execution Customization LastName: Smith FirstName: John Zip: 12345 Salary: 1000.42 Tax: 15.5 Customer LastName : EINSTEIN Customer FirstName : ALBERT Customer PhoneNumber : 3048 Customer LastName : NEWTON Customer FirstName : ISAAK Customer PhoneNumber : 4041 Customer LastName : KOHL Customer FirstName : HELMUT Customer PhoneNumber : 5154 New Salary: 1080.45 ISRT: CEETDLI successful	
13	Optional	Check results of IMSContainer_Activity .	The latest SYSxxxx section of your IMS region job should contain: FMCH3ICS: MQWF Program Execution Customization OutContainer Name = SimpleDS InContainer Name = SimpleDS Set OutputContainer long value ml rc = 0 Main: Set OutputContainer rc = 0	
14	Required	Logoff.	Log off the runtime client.	
15	Required	Stop the MQSeries Workflow for OS/390 system.	On the OS/390 system console, issue the command: MODIFY AdminServerID,STOP	
16	Required	Stop the OS/390 administration server.	On the OS/390 system console, issue the command: MODIFY AdminServerID,STOP ADM	Congratulations, you have now configured and verified your MQSeries Workflow for OS/390 system group containing one Workflow system.

Now that you have created a system group containing one Workflow system, you can proceed with the following as required:

- Add additional systems to the system group as described in “Chapter 5. Creating additional systems in an existing system group” on page 53.
- Add extra clients as described in “Chapter 6. Adding extra Workflow clients to an existing system” on page 65.

Chapter 5. Creating additional systems in an existing system group

After you have created one or more system groups containing a primary system, you can add additional systems to the system group. These additional systems inherit some attributes from the system group's primary system. Each MQSeries Workflow system which is in the same system group must have a separate queue manager, and the queue manager must be a member of the primary system's MQSeries cluster.

For each additional system that you want to add to a system group, you must perform the following steps:

1. "Decide the new system's identifiers"
2. "Data set allocation" on page 55
3. "Create input files for customizing an additional system in a system group" on page 56
4. "General DB2 customization (DB2 data sharing)" on page 56
5. "Update topology setting in the Workflow database" on page 57
6. "MQSeries customization" on page 58
7. "OS/390 trace customization" on page 58
8. "CICS API support customization" on page 59
9. "IMS API support customization" on page 61
10. "Workflow server customization" on page 61
11. "LAN client customization" on page 62

Decide the new system's identifiers

The following identifiers have scope over a Workflow system.

Table 37. Identifiers required for each new system

Parameter	Your value	Name in customization parameter file	Description
<i>CustHLQ</i>		MQWFCHLQ	The high level qualifier for the new MQSeries Workflow for OS/390 system you want to create.
<i>UniqueSystemKey</i>		MQWUFUKEY	Unique key for an MQSeries Workflow for OS/390 system, may be up to eight uppercase characters long. This is the name given to the Workflow server start job, and must be unique within SYS1.PROCLIB. This key is used in the START command to start an administration server on the <i>System</i> associated with this key.
<i>System</i>		MQWFSYSN	MQSeries Workflow for OS/390 system name. This name must be unique within the system group. This is the system where the administration server is started when the start administration server command is issued: START <i>UniqueSystemKey.AdminServerID</i> .

Table 37. Identifiers required for each new system (continued)

Parameter	Your value	Name in customization parameter file	Description
<i>SystemIdentifier</i>		MQWFSYID	This is the MQSeries Workflow system identifier. This value corresponds to the FDL keyword SYSTEM_IDENTIFIER. Each system in a system group must have a unique identifier. The primary system in a system group already has an identifier value of one. The value should be numeric and greater than one.
<i>MQWFConfiguration Key</i>		MQWFCFGK	This key must be unique for all MQSeries Workflow for OS/390 systems that you configure. It can be up to 8 uppercase characters long. This key is used inside the profiles and identifies a configuration for a system.
<i>ServerUserID</i>		STTSKUID	The server started task RACF user ID used by all MQSeries Workflow for OS/390 servers. This is the default user ID that OS/390 programs will be run under, by the PES, as a result of MQSeries Workflow process activity requests for OS/390 program invocations. This user ID requires EXECUTE rights on <i>DB2Plan</i>
<i>ServerGroupID</i>		STTSKGRP	The server started task RACF group ID for all MQSeries Workflow for OS/390 servers.
<i>CTComponent</i>		CTRCNAME	CTRACE component name.
<i>CTStartSuffix</i>		CTRCPMS1	CTRACE PARMLIB member suffix (start writer). This value may be any two alphanumeric characters.
<i>CTStopSuffix</i>		CTRCPMS2	CTRACE PARMLIB member suffix (stop writer). This value may be any two alphanumeric characters.
<i>CTWriter</i>		CTRCWPRC	CTRACE writer procedure name. This must not be more than seven characters long.
<i>ARMRestartPolicy</i>		ARMPOLNM	The name of the ARM restart policy.
<i>ARMRestartElement NameSuffix</i>		ARMRESFX	The 8 character suffix of the 16 character ARM restart element name. This value must be specified as part of the start administration server command to register the administration server with ARM. This suffix is concatenated to the constant prefix SYSMQWF_. For example, if you set ARMRESFX=MQWFS1, then the name of the ARM restart element that is used in the ARM restart policy will be SYSMQWF_MQWFS1.
EXE <i>Application EnvironmentName</i>		WLMAEEXE	Application Environment for the MQSeries Workflow for OS/390 execution server . It must be unique in the parallel sysplex environment.
PES <i>Application EnvironmentName</i>		WLMAEPES	Application Environment for the MQSeries Workflow for OS/390 program execution server . It must be unique in the parallel sysplex environment.
<i>ClusterNamelist</i>		MQCLNAME	This is the name of a MQSeries Namelist object which holds the MQSeries cluster name that you defined in Table 2 on page 10. The name must conform to the MQSeries naming rules for Object names. It is used later in all MQSeries object definitions which have cluster scope.

Table 37. Identifiers required for each new system (continued)

Parameter	Your value	Name in customization parameter file	Description
<i>QueueManager</i>		MQQMNAME	Name of the MQSeries queue manager that is to be used by MQSeries Workflow for OS/390. The queue manager name must be unique for the complete MQSeries network. Note: If you want to run CICS applications that use the MQSeries Workflow for OS/390 application program interface (API), this must either be the same queue manager that is used by CICS, or the queue managers must be members of the same MQSeries cluster.
<i>SubsystemTCP/IP Address</i>		STCPADDR	This is the TCP/IP address of the OS/390 Subsystem where the MQSeries Queue Manager resides.
<i>QueueManagerTCP/IP Port</i>		STCPPORT	This is the TCP/IP port of the listener of the Queue Manager. The MQSeries default is 1414. All Queue Managers on the OS/390 image must have different ports for their listeners.
<i>DB2SubsystemName</i>		DB2SSYSN	Name of the DB2 subsystem that is to be used by MQSeries Workflow for OS/390.

Data set allocation

This step creates the data sets that are required for the new system.

Table 38. Data set allocation

Step number	Required or optional	Description	Action	Verification
1	Required	Copy allocation job.	Copy the JCL <i>InstHLQ.SFMCCNTL</i> (FMCHJACD) to a private partitioned data set.	
2	Required	Customize allocation job.	Edit your copy of FMCHJACD, and make the changes described in the comment header of the file (replace <MQWFCHLQ> with your MQSeries Workflow for OS/390 customization high level qualifier, see <i>CustHLQ</i> in Table 37 on page 53).	
3	Required	Allocate customization data sets.	Submit your copy of FMCHJACD.	rc=0 indicates that the following libraries have been created: <ol style="list-style-type: none"> 1. <i>CustHLQ.SFMCCNTL</i> 2. <i>CustHLQ.SFMCDATA</i> 3. <i>CustHLQ.SFMCDB2</i> 4. <i>CustHLQ.SFMCMQS</i> 5. <i>CustHLQ.SFMC Parm</i> 6. <i>CustHLQ.SFMC PROC</i> 7. <i>CustHLQ.SFMCREXX</i> Note: The last two libraries are for the generated PROCLIB and PARMLIB members.

Create input files for customizing an additional system in a system group

In this task you specify all the identifiers that the customization process requires, and generate customization files from the values you have entered. If you later realize that the identifiers were not correct, you must repeat this task before repeating the customization process.

Table 39. Create input files for customization

Step number	Required or optional	Description	Action	Verification
1	Required	Copy customization templates.	<ol style="list-style-type: none"> 1. Copy the JCL <i>InstHLQ.SFMCCNTL(FMCHJCCT)</i> to a private partitioned data set. 2. Edit your copy of FMCHJCCT as described in the comment header. 3. Submit your copy of FMCHJCCT 	rc=0 for the copy step. The JCL also deletes any members in the libraries for the generated PROCLIB and PARMLIB members <i>CustHLQ.GENPROC</i> and <i>CustHLQ.GENPARM</i> . rc=8 can be accepted for the delete step, it indicates that there was nothing to delete.
2	Required	Copy customization template file for additional systems	Copy the file from <i>CustHLQ.SFMCDATA(FMCHCECSY)</i> for your primary system to new <i>CustHLQ.SFMCDATA(FMCHCECIF)</i> . Note that it is going from your primary system customization high level qualifier (see Table 3 on page 11) to your customization high level qualifier of your new system (see Table 37 on page 53).	
3	Required	Edit the customization parameter file.	Edit the customization parameter template member <i>CustHLQ.SFMCDATA(FMCHCECIF)</i> , and enter your values from Table 37 on page 53, as described in the comment sections of the file. Note: This file is described in “Customization parameter file for adding a system to a system group” on page 186. From now on, this member will contain your customization parameters for this new system, and from here on in this chapter, <i>CustHLQ</i> refers to the value in Table 37 on page 53. This member is used as an input file for the generation process in step 4.	
4	Required	Generate all the JCLs necessary to customize this product.	<ol style="list-style-type: none"> 1. Copy the JCL <i>CustHLQ.SFMCCNTL(FMCHJCUS)</i> to a private partitioned data set. 2. Edit your copy of FMCHJCUS as described in the comment header. 3. Submit your copy of FMCHJCUS. 	This requires rc=0. The program performs some syntax checking on the length and value of the variables you specified in the file <i>CustHLQ.SFMCDATA(FMCHCECIF)</i> . The program then substitutes your values for variables in the customization template files. Some PROCLIB and PARMLIB members are also copied with new names to the library <i>CustHLQ.GENPROC</i> and <i>CustHLQ.GENPARM</i> .

General DB2 customization (DB2 data sharing)

If one of the following statements is true, you can skip this section and continue with “Update topology setting in the Workflow database” on page 57, because you have already done, or you do not need to set up the DB2 Data Sharing/Coupling Facility:

- You are not using a DB2 Data Sharing Group because you want to create the additional Workflow system on the same OS/390 image where your primary system resides.
- You are using a DB2 Data Sharing Group, but you are not creating the first system in this system group on the current OS/390 image.

Before performing this customization you should ensure that you have DB2 SYSADM grants. This can be granted with the command: GRANT SYSADM TO *DB2AdminUserID*. Before submitting each JCL, be sure to insert your own job card.

Table 40. General DB2 customization (DB2 data sharing)

Step number	Required or optional	Description	Action	Verification
1	Required if you want to change the buffer pool names and sizes:	Edit buffer pool definitions.	1. Edit <i>CustHLQ.SFMCDB2(FMCHDDBP)</i> 2. Change the buffer pool definitions. Verify that you have the same table space/index space to buffer pool correlation as in your previous systems in the system group.	
2	Required	Define the buffer pools.	Submit JCL <i>CustHLQ.SFMCNTL(FMCHJDBP)</i>	rc=0
3	Required	Check structures in Coupling Facility:	Make sure that you have group buffer pool structures available for all local buffer pools included in <i>CustHLQ.SFMCDB2(FMCHDDBP)</i> . In the following you will find some recommendations for your structure sizing but they have to be monitored and adapted later (if required) for your environment – for more information please refer to <i>DB2 for OS/390 Data Sharing: Planning and Administration (SC26-8961-00)</i> . Group Buffer Pool Sizes: For each buffer pool you are using (see <i>CustHLQ.SFMCDB2(FMCHDDBP)</i>) sum the local buffer pool storage for the buffer pool number across all DB2s of the group. Then, use approximately one third of that total as your group buffer pool size. Check your data page/directory entry ratio . Make sure that you can have at least as many directory entries for each buffer pool number as the sum of pages in your local buffer pools.	
4	Optional	Change group buffer pool settings	1. Edit <i>CustHLQ.SFMCDB2(FMCHDDGB)</i> 2. Change the default values to values suitable for your environment. 3. Submit JCL <i>CustHLQ.SFMCNTL(FMCHJDBP)</i>	rc=0

Update topology setting in the Workflow database

To populate, and verify the Workflow database, you must perform the following steps:

Table 41. Updating topology setting in the Workflow database

Step number	Required or optional	Description	Action	Verification
1	Optional	Check the defaults for the new system.	Edit the reference FDL file <i>CustHLQ.SFMCDATA(FMCHES32)</i> , modifying the default definitions for the additional system where necessary.	

Table 41. Updating topology setting in the Workflow database (continued)

Step number	Required or optional	Description	Action	Verification
2	Required	Submit job to add a system to a system group.	Submit your copy of <i>CustHLQ.SFMCCNTL(FMCHJRIS)</i>	rc=0

MQSeries customization

This defines all the MQSeries resources required by MQSeries Workflow for OS/390. Before you perform this customization, make sure that your queue manager is started.

Table 42. MQSeries customization

Step number	Required or optional	Description	Action	Verification
1	Required	Define the MQSeries resources (except for program execution.)	Submit JCL <i>CustHLQ.SFMCCNTL(FMCHJDMQ)</i>	rc=0
2	Required	Define the MQSeries resources required by MQSeries Workflow for OS/390 program execution.	Submit JCL <i>CustHLQ.SFMCCNTL(FMCHJPMQ)</i>	rc=0

OS/390 trace customization

One Workflow Server Trace exists for each Workflow system. As this trace uses OS/390 system components (Component Trace or CTRACE for short), you must also update system resources. You are provided with two templates for members of SYS1.PARMLIB which control the tracing of Workflow servers. You are provided with one template for a JCL procedure member of SYS1.PROCLIB which controls a CTRACE external writer. Finally, you must provide and specify in the JCL procedure, the trace output data sets for the external writer.

Table 43. OS/390 trace customization

Step number	Required or optional	Description	Action	Verification
1	Required	Look up variable values.	<p>Check Table 37 on page 53, and note the values that you planned for the following identifiers:</p> <ol style="list-style-type: none"> 1. The two alphanumeric characters <i>nn</i> for <i>CTStartSuffix</i>. 2. The two alphanumeric characters <i>mm</i> for <i>CTStopSuffix</i>. 3. The value for <i>CTWriter</i>. <p>Note: These are the values that you should have assigned to the variables <CTRCPMS1>, <CTRCPMS2>, and <CTRCWPRC> respectively, in file <i>CustHLQ.SFMCDATA(FMCHCEIF)</i></p>	
2	Required	Check system libraries.	<p>Make sure that the system (or the sysplex) does not already contain the members:</p> <ol style="list-style-type: none"> 1. SYS1.PARMLIB(CTIFMCnn). 2. SYS1.PARMLIB(CTIFMCmm). 3. SYS1.PROCLIB(CTWriter). <p>Where <i>nn</i>, <i>mm</i>, and <i>CTWriter</i> are the values from step 1.</p>	

Table 43. OS/390 trace customization (continued)

Step number	Required or optional	Description	Action	Verification
3	Required	Copy PARMLIB members.	<ol style="list-style-type: none"> 1. Copy the Component Trace Start PARMLIB member <i>CustHLQ.GENPARM(CTIFMCnn)</i> to SYS1.PARMLIB 2. Copy the Component Trace Stop PARMLIB member <i>CustHLQ.GENPARM(CTIFMCmm)</i> to SYS1.PARMLIB 	
4	Required	Copy PROCLIB member	<ol style="list-style-type: none"> 1. Copy the trace writer <i>CustHLQ.GENPROC(CTWriter)</i> to SYS1.PROCLIB 	
5	Required	Create the extended trace output data sets.	<ol style="list-style-type: none"> 1. Edit data set <i>CustHLQ.SFMCCNTL(FMCHJCTR)</i> 2. Submit JCL <i>CustHLQ.SFMCCNTL(FMCHJCTR)</i> 	rc=0
6	Required	Provide RACF profiles.	<p>Give update access (to the trace data sets created in step 5) to the user ID that is assigned to the trace writer.</p> <p>Note: If no explicit user ID assignment is made, the trace writer runs under the user ID STCUSER and group ID STCGRP.</p>	

CICS API support customization

If you want to use the MQSeries Workflow for OS/390 API and trace in CICS, then you must perform this customization. If you only want to use CICS legacy applications, or if you do not want to use CICS at all you can skip this customization, and continue at “IMS API support customization” on page 32.

Before starting this customization, you should ensure that the queue managers used by CICS and MQSeries Workflow for OS/390 are members of the same cluster, you should also perform a CICS shutdown.

Table 44. CICS API support customization

Step number	Required or optional	Description	Action	Verification
1	Required	Enable LE and C/C++ features in CICS.	<p>If they are not already enabled:</p> <ol style="list-style-type: none"> 1. Enable LE in CICS. Note: The CSD definitions necessary to accomplish this task are located in <i>LEInstHLQ.SCEESAMP(CEECCSD)</i> 2. Enable the C/C++ feature in CICS. Note: A sample that may help you with this task is located in <i>CCPPIInstHLQ.SCLBSAM(CLB3YCSD)</i> <p>where <i>LEInstHLQ</i> and <i>CCPPIInstHLQ</i>, are your values from Table 4 on page 13.</p>	
2	Required	Specify the location of the Workflow executables, and start-up parameters.	<ol style="list-style-type: none"> 1. Edit your CICS start-up job. 2. Find the DFHRPL entry. 3. Add the MQSeries Workflow for OS/390 library called <i>InstHLQ.SFMCLOAD</i> to the DFHRPL entry. 4. Specify an EDSALIM value of at least 200M and a CICS region size that will accommodate your EDSALIM setting. For example, specify the CICS parameter <i>EDSALIM=200M</i> and <i>REGION=220M</i> in your CICS start-up job. 	

Table 44. CICS API support customization (continued)

Step number	Required or optional	Description	Action	Verification
3	Required	Create user profile, machine profile, and environment data in VSAM format.	<ol style="list-style-type: none"> 1. Edit <i>CustHLQ.SFMCCNTL(FMCHJCPR)</i> 2. Change the CICSVOL value to the name of the volume where you want the profiles to be located. 3. Submit JCL <i>CustHLQ.SFMCCNTL(FMCHJCPR)</i> 	rc=0
4	Optional	If you do not want to use the value for <i>CICSGroup</i> that you specified in Table 5 on page 14:	Change the group in the CSD file: <ol style="list-style-type: none"> 1. Edit the CSD file <i>CustHLQ.SFMCDATA(FMCHEPRO)</i> 2. Change the GROUP values to the one(s) you intended for the Workflow executable, profiles, etc.. 	
5	Required	Update CICS CSD with file definitions for C++ and MQSeries Workflow for OS/390.	<ol style="list-style-type: none"> 1. Edit <i>CustHLQ.SFMCCNTL(FMCHJCUP)</i> 2. Change the CICSNAME value to the name of the CICS system that you are customizing. 3. Submit JCL <i>CustHLQ.SFMCCNTL(FMCHJCUP)</i> 	rc=0
6	Required	Restart CICS.	Restart CICS.	
7	Required	Make MQSeries CICS stubs available in CICS.	Make the MQSeries CICS Stubs IMQB23IC and IMQS23IC from <i>MQInstHLQ.SCSQLOAD</i> available to CICS.	
8	Required	Make the C/C++ group CLB and the MQSeries Workflow for OS/390 group available in CICS.	<ol style="list-style-type: none"> 1. Make the C/C++ group CLB available in CICS, with the command: CEDA ADD G(CLB) LIST(<i>xxx</i>) 2. Make the MQSeries Workflow for OS/390 group (<i>CICSGroup</i>, unless you changed it in step 4) available in CICS, with the command: CEDA ADD G(<i>yyy</i>) LIST(<i>xxx</i>) <p>where <i>xxx</i> is a LIST used at CICS start-up, and <i>yyy</i> is the MQSeries Workflow for OS/390 group.</p>	
9	Required	Stop and restart CICS.	Stop and restart CICS.	
10	Required	Verify profile access.	<ol style="list-style-type: none"> 1. Logon to CICS. 2. Perform: CEMT I FI(FMCHEUPR) 3. One file should be displayed. Try to open the file by typing "OPE" over "CLO" (and pressing enter). If this works without resulting in an error message, the profile access has been established. If you get an error message, retry the previous steps for enabling CICS API support. If this does not help, contact your IBM representative. 4. You can now close the file again by typing "CLO" over "OPE" Since CICS will then disable the file, type "ENA" over "UNE" (UNEnabled). 	

IMS API support customization

This makes MQSeries Workflow for OS/390 DLLs available to IMS so that programs using the MQSeries Workflow for OS/390 container API can be executed in IMS. If you only want to use IMS legacy applications, or if you do not want to use IMS at all you can skip this customization, and continue at “Workflow server customization”.

Table 45. IMS API support customization

Step number	Required or optional	Description	Action
1	Required	Provide load modules for IMS	Add all members with the prefix “FMCH3” from the library <i>InstHLQ.SFMCLoad</i> library to your IMS PGMLIB library.

Workflow server customization

To enable a Workflow server, a JCL procedure has to be provided in SYS1.PROCLIB.

Table 46. Workflow server customization

Step number	Required or optional	Description	Action	Verification
1	Required	Copy definitions for Workflow servers into a procedure library.	Copy the JCL procedure <i>CustHLQ.GENPROC(UniqueSystemKey)</i> to SYS1.PROCLIB, where <i>UniqueSystemKey</i> is your value in Table 37 on page 53.	
2	Required	Assign RACF user ID and group to the Workflow server started task. Note: This is the <i>ServerUserID</i> , see Table 37 on page 53	Submit JCL <i>CustHLQ.SFMCCNTL(FMCHJDSC)</i>	rc=0
3	Required	Provide RACF profile.	Give the <i>ServerUserID</i> assigned in Step 2 read access to the data set <i>CustHLQ.SFMCDATA</i> .	
4	Required if you want more than eight server instances per address space, or if you want console messages to be in uppercase, otherwise optional	Modify the Workflow server start job definitions.	<ol style="list-style-type: none"> Edit SYS1.PROCLIB(<i>UniqueSystemKey</i>). If you want, you can extend the definitions to allow more than eight server instances per address space. If you want console messages from the server address space to be in uppercase, change the value for LANGC to ENP (the default value is ENU). If you want, you can modify the DD statements for the stdout and stderr output, and the simple trace output. If you want, you can modify the sysout class for stdout, stderr, and simple trace output. 	
5	Required only if you want servers and tools to give MMS messages in uppercase	Modify the language setting in the machine profile.	Edit the server machine profile <i>CustHLQ.SFMCDATA(FMCHEMPR)</i> , and change the Language setting to ENP for uppercase U.S. English. The default value is ENU (mixed-case U.S. English). Note: These messages are generally routed to SYSOUT data sets	

Table 46. Workflow server customization (continued)

Step number	Required or optional	Description	Action	Verification
6	Required	Grant the server user ID execute access to the database plan.	Issue the command: GRANT EXECUTE ON PLAN <i>DB2Plan</i> TO <i>ServerUserID</i> using your values for <i>DB2Plan</i> in Table 2 on page 10, and <i>ServerUserID</i> in Table 37 on page 53.	

LAN client customization

This task describes how to configure a MQSeries Workflow LAN client to connect to a MQSeries Workflow for OS/390 server. This task consists of two parts:

1. "Customize the MQSeries client connection"
2. "Customize the MQSeries Workflow client" on page 63

Note: It is very important that you check the files called *Readme.1st* and *Readme.xxx* (where *xxx* is your language code) on the *MQSeries Workflow Version 3.2.1 CD*.

Customize the MQSeries client connection

To set up an MQSeries client connection you must do the following:

Table 47. Customize the MQSeries client connection

Step number	Required or optional	Description	Action
1	Required	Install MQSeries client.	Install an MQSeriesclient from the MQSeries CD as described in the MQSeries Workflow product documentation.
2	Required	Generate a channel tab file.	<ol style="list-style-type: none"> 1. Edit <i>CustHLQ.SFMCCNTL(FMCHJCCC)</i> and replace the <volume> parameter with the DASD name where you want to have your generated channel tab stored. 2. Decide which client connection channel definitions you want to have to be created within the channel tab: (a) Only for the current Workflow system or (b) Also for further systems you want to connect to using the channel tab which will be created within this step. <ol style="list-style-type: none"> a. For the current Workflow system: Submit the JCL <i>CustHLQ.SFMCCNTL(FMCHJCCC)</i>, and expect rc=0. b. For multiple Workflow systems: <ol style="list-style-type: none"> 1) Edit the JCL <i>CustHLQ.SFMCCNTL(FMCHJCCC)</i> and concatenate all client connection definitions to the CSQUCMD DD statement in step FMCHJCC1. One way to do this is by inserting all required definitions into the already included file <i>CustHLQ.SFMCMQS(FMCHNCCC)</i>, by the concatenation of FMCHNCCC files of multiple systems (if you have them all on shared DASD). Another way to do this is to concatenate a file of your own to concatenate a file of your own to the collected client channel definitions. 2) Submit the JCL <i>CustHLQ.SFMCCNTL(FMCHJCCC)</i>, and expect rc=0.

For more information about MQSeries client connection, see the MQSeries documentation *MQSeries Clients*. Now your MQSeries client connection is defined; you are ready to customize the MQSeries Workflow client.

Customize the MQSeries Workflow client

To set up an MQSeries Workflow client you must do the following:

Table 48. Customize the MQSeries Workflow client

Step number	Required or optional	Description	Action
1	Required	Install an MQSeries Workflow client.	Install an MQSeries Workflow client from the <i>MQSeries Workflow Version 3.2.1 CD</i> as described in <i>IBM MQSeries Workflow: Installation Guide</i> .
2	Required	Download channel tab file.	Download the file <i>ClientCustHLQ.mqwfchl.tab</i> binary file from OS/390 to a directory on the workstation where you want to run your client.
3	Required	Configure MQSeries Workflow client.	Configure the client using the MQSeries Workflow customization tool, as described in the documentation.
4	Optional	Verify Workflow client sample application.	Perform the verification described in "Verify Workflow client sample application" on page 37.

Now you have customized the MQSeries Workflow client.

Now you have created a new Workflow system in a system group. To add another system to a system group, you must repeat this chapter from "Decide the new system's identifiers" on page 53.

Chapter 6. Adding extra Workflow clients to an existing system

This chapter describes how you can add extra clients on a queue manager to an existing system. Each MQSeries Workflow client on a queue manager must have a separate queue manager, the queue manager must be a member of the same MQSeries cluster as the MQSeries Workflow system to which you want to add the new client.

This chapter describes how you can customize the following:

- “Basic client customization”.
- “CICS API support customization for new client” on page 68
- “IMS API support customization for new client” on page 70
- “LAN Client Customization” on page 70
- “Client request concentrator customization” on page 71

Basic client customization

In this task you specify all the identifiers that the customization process requires, and generate customization files from the values you have entered. If you later realize that the identifiers were not correct, you must repeat this task before repeating the customization process.

Each time that you want to create a new MQSeries Workflow for OS/390 client on a queue manager, you must perform a customization. Before starting customization, you must complete the following pre-customization tasks. This creates the libraries and copies files from the customization high level qualifier *CustHLQ* of an existing MQSeries Workflow for OS/390 system to the location of the new client that is to be customized (*ClientCustHLQ*). The information you enter here is used to generate customization files.

The basic client customization includes the following steps:

1. “Decide the new client’s identifiers”
2. “Data set allocation for client” on page 66
3. “Create input files for customizing a new Client” on page 66
4. “MQSeries customization for a new client” on page 67
5. “Generate MQSeries channel tab file for LAN client” on page 67

Decide the new client’s identifiers

First you must decide the new Client’s identifiers. The following identifiers have the scope of a Workflow client.

Table 49. Identifiers required for each new client

Parameter	Your value	Name in customization parameter file	Description
<i>ClientCustHLQ</i>		CLNTCHLQ	This is the customization high level qualifier for the client you want to customize.

Table 49. Identifiers required for each new client (continued)

Parameter	Your value	Name in customization parameter file	Description
<i>ClientConfigKey</i>		CLNTCFGK	The key could be up to 8 uppercase characters long. This key is used inside the profiles and identifies a configuration for a client.
<i>ClientQueueManager</i>		CLQMNAME	This is the MQSeries queue manager name of the client inside the cluster of the server systems. The queue manager name must be unique in the MQSeries network. Note: If you want to run CICS applications that use the MQSeries Workflow for OS/390 application program interface (API), this must either be the same queue manager that is used by CICS, or the queue managers must be members of the same MQSeries cluster.
<i>TCP/IP address of Subsystem</i>		CTCPADDR	This is the TCP/IP address of the OS/390 subsystem where the MQSeries Queue Manager resides.
<i>TCP/IP port of MQSeries Queue Manager</i>		CTCPPORT	This is the TCP/IP port of the listener of the queue manager. The MQSeries default is 1414. All queue managers on the OS/390 image must have different ports for their listeners.
<i>ClientCICSGroup</i>		CICSGRPC	CICS group name used for program execution server invocations.

Data set allocation for client

This step creates the data sets that are required for customization.

Table 50. Data set allocation for client

Step number	Required or optional	Description	Action	Verification
1	Required	Copy allocation job.	Copy the JCL <i>CustHLQ</i> .SFMCNTL (FMCHJACL) to a private partitioned data set.	
2	Required	Customize allocation job.	Edit your copy of FMCHJACL, and make the changes described in the comment header of the file (replace <CLNTCHLQ> with your MQSeries Workflow for OS/390 client customization high level qualifier, see <i>ClientCustHLQ</i> in Table 49 on page 65).	
3	Required	Allocate customization data sets.	Submit your copy of FMCHJACL.	rc=0 indicates that the following libraries have been created: <ol style="list-style-type: none"> 1. <i>ClientCustHLQ</i>.SFMCNTL 2. <i>ClientCustHLQ</i>.SFMCDATA 3. <i>ClientCustHLQ</i>.SFMCMQS 4. <i>ClientCustHLQ</i>.SFMCPROC 5. <i>ClientCustHLQ</i>.SFMCREXX 6. <i>ClientCustHLQ</i>.GENPROC

Create input files for customizing a new Client

In this task you specify all the identifiers that the customization process requires, and generate customization files from the values you have entered. If you later realize that the identifiers were not correct, you must repeat this task before repeating the customization process.

Table 51. Create input files for client customization

Step number	Required or optional	Description	Action	Verification
1	Required	Copy customization templates.	<ol style="list-style-type: none"> 1. Copy the JCL <i>CustHLQ.SFMCCNTL(FMCHJCLT)</i> to a private partitioned data set. 2. Edit your copy of FMCHJCLT as described in the comment header. 3. Submit your copy of FMCHJCLT 	rc=0 for the copy step. rc=8 can be accepted for the delete step, it indicates that there was nothing to delete.
2	Required	Edit the customization parameter file.	Edit the customization parameter template member <i>CustHLQ.SFMCDATA(FMCHCCCL)</i> , and enter your values from Table 49 on page 65, as described in the comment sections of the file. This member is used as an input file for the generation process in step 3.	
3	Required	Generate all the JCLs necessary to customize this product.	<ol style="list-style-type: none"> 1. Copy the JCL <i>ClientCustHLQ.SFMCCNTL(FMCHJCCS)</i> to a private partitioned data set. 2. Edit your copy of FMCHJCCS as described in the comment header. 3. Submit your copy of FMCHJCCS. 	This requires rc=0. The program performs some syntax checking on the length and value of the variables you specified in the file <i>ClientCustHLQ.SFMCDATA(FMCHCCCL)</i> . The program substitutes your values for variables (see Table 49 on page 65) into the client customization template files.

MQSeries customization for a new client

This defines all the MQSeries resources required by an MQSeries Workflow for OS/390 client on a queue manager. Before you perform this customization, make sure that your queue manager is started.

Table 52. MQSeries customization for new client

Step number	Required or optional	Description	Action	Verification
1	Required	Define the MQSeries resources.	Submit JCL <i>ClientCustHLQ.SFMCCNTL(FMCHJCMQ)</i>	rc=0

Generate MQSeries channel tab file for LAN client

To generate the channel tab file for the new workstation client, you must perform the following:

Table 53. Generate MQSeries Channel tab file for use on LAN Client

Step number	Required or optional	Description	Action	Verification
1	Required	Generate MQSeries Channel tab file for use on LAN Client.	<ol style="list-style-type: none"> 1. Edit the job <i>ClientCustHLQ.SFMCCNTL(FMCHJCCL)</i> and replace the <volume> parameter with the DASD name where you want to have your generated channel tab stored. 2. Decide which client connection channel definitions you want to have to be created within the channel tab: (a) Only for the Workflow system to which you want to add the current client or (b) Also for further systems to which you want to connect to using the channel tab which will be created within this step. <ol style="list-style-type: none"> a. For the Workflow system to which you want to add the current client: Submit the JCL <i>ClientCustHLQ.SFMCCNTL(FMCHJCCL)</i>, and expect rc=0. b. For multiple Workflow systems: <ol style="list-style-type: none"> 1) Edit the JCL <i>ClientCustHLQ.SFMCCNTL(FMCHJCCL)</i> and concatenate all client connection definitions to the CSQUCMD DD statement in step FMCHJCC1. One way to do this is by inserting all required definitions into the already included file <i>ClientCustHLQ.SFMCMQS(FMCHNCCL)</i>, by the concatenation of FMCHNCCL files of multiple clients (if you have them all on shared DASD). Another way to do this is to concatenate a file of your own to the collected client channel definitions. 2) Submit the JCL <i>ClientCustHLQ.SFMCCNTL(FMCHJCCL)</i>, and expect rc=0. 	
2	Required	Download the generated tab file to your workstation.	Transfer the binary file <i>ClientCustHLQ.MQWFCHL.TAB</i> to your workstation where you want to configure the MQSeries client connection.	

CICS API support customization for new client

If you want to use the MQSeries Workflow for OS/390 API and trace in CICS, then you must perform this customization.

Before starting this customization, you should:

1. A new CICS system, that has not yet been customized.
2. Ensure that the queue managers used by CICS and MQSeries Workflow for OS/390 system to which you want to add a new client are members of the same cluster.
3. Complete “MQSeries customization for a new client” on page 67.
4. Perform a CICS shutdown.

Table 54. CICS API support customization for new client

Step number	Required or optional	Description	Action	Verification
1	Required	Customize the client queue manager.	Perform “MQSeries customization for a new client” on page 67.	

Table 54. CICS API support customization for new client (continued)

Step number	Required or optional	Description	Action	Verification
1	Required	Enable LE and C/C++ features in CICS.	<p>If they are not already enabled:</p> <ol style="list-style-type: none"> 1. Enable LE in CICS. Note: The CSD definitions necessary to accomplish this task are located in <i>LEInstHLQ.SCEESAMP(CEECCSD)</i> 2. Enable the C/C++ feature in CICS. Note: A sample that may help you with this task is located in <i>CCPPIInstHLQ.SCLBSAM(CLB3YCSD)</i> <p>where <i>LEInstHLQ</i> and <i>CCPPIInstHLQ</i>, are your values from Table 4 on page 13.</p>	
2	Required	Specify the location of the Workflow executables, and start-up parameters.	<ol style="list-style-type: none"> 1. Edit your CICS start-up job. 2. Find the DFHRPL entry. 3. Add the MQSeries Workflow for OS/390 library called <i>InstHLQ.SFMCLOAD</i> to the DFHRPL entry. 4. Specify an EDSALIM value of at least 200M and a CICS region size that will accommodate your EDSALIM setting. For example, specify the CICS parameter <i>EDSALIM=200M</i> and <i>REGION=220M</i> in your CICS start-up job. 	
3	Required	Create user profile, machine profile, and environment data in VSAM format.	<ol style="list-style-type: none"> 1. Edit <i>ClientCustHLQ.SFMCNTL(FMCHJCPT)</i> 2. Change the CICSVOL value to the name of the volume where you want the profiles to be located. 3. Submit JCL <i>ClientCustHLQ.SFMCNTL(FMCHJCPT)</i> 	rc=0
4	Optional	If you do not want to use the value for <i>ClientCICSGroup</i> that you specified in Table 49 on page 65:	<p>Change the group in the CSD file:</p> <ol style="list-style-type: none"> 1. Edit the CSD file <i>CustHLQ.SFMCDATA(FMCHJPCPT)</i> 2. Change the GROUP values to the one for the Workflow client. 	
5	Required	Update CICS CSD with file definitions for C++ and MQSeries Workflow for OS/390.	<ol style="list-style-type: none"> 1. Edit <i>ClientCustHLQ.SFMCNTL(FMCHJCUT)</i> 2. Change the CICSNAME value to the name of the CICS system that you are customizing. 3. Submit JCL <i>ClientCustHLQ.SFMCNTL(FMCHJCUT)</i> 	rc=0
6	Required	Restart CICS.	Restart CICS.	
7	Required	Make MQSeries CICS stubs available in CICS.	Make the MQSeries CICS Stubs <i>IMQB23IC</i> and <i>IMQS23IC</i> from <i>MQInstHLQ.SCSQLOAD</i> available to CICS.	
8	Required	Make the C/C++ group CLB and the MQSeries Workflow for OS/390 group available in CICS.	<ol style="list-style-type: none"> 1. Make the C/C++ group CLB available in CICS, with the command: <i>CEDA ADD G(CLB) LIST(xxx)</i> 2. Make the MQSeries Workflow for OS/390 group (<i>CICSGroup</i>, unless you changed it in step 4) available in CICS, with the command: <i>CEDA ADD G(yyy) LIST(yyy)</i> <p>where <i>xxx</i> is a LIST used at CICS start-up, and <i>yyy</i> is the MQSeries Workflow for OS/390 group.</p>	

Table 54. CICS API support customization for new client (continued)

Step number	Required or optional	Description	Action	Verification
9	Required	Stop and restart CICS.	Stop and restart CICS.	
10	Required	Verify profile access.	<ol style="list-style-type: none"> 1. Logon to CICS. 2. Perform: CEMT I FI(FMCHEUPR) 3. One file should be displayed. Try to open the file by typing "OPE" over "CLO" (and pressing enter). If this works without resulting in an error message, the profile access has been established. If you get an error message, retry the previous steps for enabling CICS API support. If this does not help, contact your IBM representative. 4. You can now close the file again by typing "CLO" over "OPE" Since CICS will then disable the file, type "ENA" over "UNE" (UNEnabled). 	

IMS API support customization for new client

This makes MQSeries Workflow for OS/390 DLLs available to IMS so that programs using the MQSeries Workflow for OS/390 container API can be executed in IMS.

Table 55. IMS customization

Step number	Required or optional	Description	Action
1	Required	Provide load modules for IMS	Add all members with the prefix "FMCH3" from the library <i>InstHLQ.SFMCLoad</i> library to your IMS PGMLIB library.

LAN Client Customization

To customize a LAN client, you must perform the following:

Table 56. Enabling a client to use the Workflow API

Step number	Required or optional	Description	Action
1	Required	Customize MQSeries on OS/390.	Perform "MQSeries customization for a new client" on page 67.
2	Required	Check / decide identifier names required for step 3.	<ol style="list-style-type: none"> 1. Collect the following information: <ul style="list-style-type: none"> • <i>SystemGroup</i> from Table 2 on page 10. • <i>MQWFSystemPrefix</i> from Table 2 on page 10. • <i>System</i> from Table 3 on page 11. • <i>ClientQueueManager</i> from Table 49 on page 65. 2. Decide how do you want to name the new configuration
3	Required	Customize MQSeries on your workstation client.	Perform the MQSeries customization for your workstation client, as described in the appropriate MQSeries documentation.

Client request concentrator customization

You can configure a 'client concentrator' that accepts requests from several clients, and forwards the requests to a cluster of servers. The client request concentrator acts as point where many MQSeries clients are connected. A concentrator isolates the other queue managers from the client connection workload. It also allows you to use the MQSeries clustering features without writing a cluster workload exit.

The main advantages of doing this are:

- They connect to the same queue manager.
- They have the same channel tab.
- The work requests from the clients are distributed evenly across all the servers in the MQSeries cluster.
- Reduces the workload for client connections on the other queue managers in the cluster.

Note: Only MQSeries Workflow Version 3.2.1 and higher clients can be attached to a client request concentrator.

Table 57. Enabling a client to act as a client request concentrator

Step number	Required or optional	Description	Action
1	Required	Basic customization.	Perform "Basic client customization" on page 65.
2	Required	MQSeries customization.	Perform "MQSeries customization for a new client" on page 67.

Part 2. System administration

Chapter 7. Introduction to system administration	75
Objects you will need to administer or use	75
Administration in an MQSeries Workflow system	78
System administration client/server components	78
The administration server	79
The administration console	80
Overview of administration tasks	80
System and server administration tasks	80
Program and user administration tasks	81
Chapter 8. Administration console tasks	83
Administration server commands	83
Starting the administration server and administration console	83
Stopping the administration server	84
System commands	84
Starting the system	84
When using WLM	85
Stopping the system	85
Restarting the system	85
Displaying all server instances in the system	86
Server commands	86
Starting servers	86
Starting WLM managed servers	87
Starting execution servers	87
Starting program execution server instances	87
Stopping servers	87
Restarting servers	88
Restarting the program execution server	88
Restarting the administration server	88
Restarting WLM managed servers	89
Displaying the number of instances of a server	89
Hold queue commands	89
Displaying number of messages in the hold queue	90
Displaying messages in the hold queue	90
Replaying messages from the hold queue	90
Deleting messages from the hold queue	91
Chapter 9. Buildtime administration tasks	93
Defining process models	93
Defining server properties	93
Server properties that can be changed	93
Server properties that should not be changed	94
Server properties that are ignored on OS/390	94
Switching servers between WLM and non-WLM mode using Buildtime	94
Defining program properties	96
Defining the connection between a program activity and the PES	100
Uploading process models to the host	101
Importing and exporting process models	101
Using the FDL import/export tool	101
Chapter 10. Program execution	103
Administering the Program Execution Server directory	105
Adding a new service definition and the related user resolution information	106
Adding a user-defined invocation type	107
Adding a user-defined mapping type	107
Importing the PES directory	107
Caching the PES directory at runtime	108
Refreshing the PES directory cache	108
Administering programs	108
Enabling an OS/390 program to be run as a program activity	108
Creating a program mapping	108
Defining a new program in the process model	109
Defining a security profile	109
Enabling an OS/390 program to run as a safe application	109
Disabling a program	109
Authorizing a user to access an OS/390 program	109
Revoking a user's access to OS/390 programs	110
Administering program mapping	110
Importing a program mapping definition	110
Return codes	112
Enabling a program's mapping	112
Disabling a program's mapping	113
Deleting a program mapping definition	113
Enabling a mapping type	113
Disabling a mapping type	114
Administering invocation types	114
Enabling an invocation type	114
Disabling an invocation type	114
Program execution security	114
Information in the PES directory that is relevant to security	116
Program security	116
Chapter 11. Administering Servlets on the WebSphere Application Server	117
Placing servlet class files on the Application Server	117
If necessary, create a new servlet sub-directory	117
Monitoring your servlet, or setting servlet initialization parameters	118
Placing the HTML files on the Application Server	118
Running a sample servlet, to log on MQSeries Workflow	118
Chapter 12. Performance tuning	121
Changing the number of running server instances	121
Changing the number of server instances per address space	121
Caching the PES directory	122
Chapter 13. Problem determination	123

Server problems	123	Running out of spool space.	128
Message catalog not available	123	Is tracing turned on?	128
Servers cannot be started	123	The MQSeries Workflow for OS/390 system trace	
Server terminates immediately	123	facility.	129
All but one server instance terminates		Turning tracing on.	129
immediately after starting	123	Restarting the component and reproducing the	
No execution server instances start	124	problem	130
The administration server cannot be started	124	Viewing the trace	130
Is the queue manager started?	124	Turning tracing off	131
Is an administration server already running?	124	Tracing in CICS	131
Are its queues inhibited?	124	What do I do if I get an SVC dump?	131
The administration server does not respond to		Create a problem summary.	131
console commands	124		
The program execution server cannot be started	124		
Is the administration server running?	124		
Are its queues inhibited?	124		
Server instance terminates unexpectedly without			
reporting an error	125		
One or more program execution server instances			
terminate, request goes in state error			
(FMC_ERROR_RETRY_LIMIT_REACHED).	125		
Program execution request stays in state			
running	125		
A dump is written before all server instances			
are started	125		
Have you exceeded the limit set in the server			
start job?	125		
Is the number of server instances per address			
space too high?	126		
Cannot stop servers	126		
Did you wait long enough?	126		
Do your transactions take longer than 30			
seconds?	126		
PES cannot be stopped	126		
Changes made to the machine profile are not			
activated	126		
Have you restarted the administration			
server?	126		
Changes made to the PES directory are not			
activated	127		
Is PES directory caching enabled?	127		
Changes made to the program mapping			
definition are not activated	127		
Have you restarted the program execution			
server?	127		
Hold queue problems (undelivered messages)	127		
DELETE or REPLAY affected fewer messages			
than expected	127		
REPLAY is not executed, the hold queue			
contains less than the expected number of			
messages	127		
Resource and performance problems	127		
User response times are unacceptably long	127		
Is tracing turned on?	127		
Are enough server instances running?	127		
Are too many server instances running?	128		
Does the workload exceed your system's			
capacity?	128		
Invalid password	128		
Are you using an old version of the runtime			
client?	128		

Chapter 7. Introduction to system administration

This chapter introduces you to system administration in an MQSeries Workflow for OS/390 system and describes the two main system administration components, namely, the administration server and the administration console.

Objects you will need to administer or use

The administration of MQSeries Workflow for OS/390 requires that you use the following MQSeries Workflow and OS/390 objects and products.

Workflow system

A set of Workflow servers that includes:

- One administration server.
- One or more execution server instances.
- One scheduling server.
- One cleanup server
- Zero or more program execution server instances.

How many of each type are started automatically when the system is started can be specified in Buildtime.

System console

Before you can administer a system, you must start the administration server for that system. You do this by issuing the start command on the system console. Having an administration server running makes the administration console available.

Administration server

The component that performs administration functions within an MQSeries Workflow system. For OS/390 the administration server must be started manually from the system console, as described in "Starting the administration server and administration console" on page 83.

Administration console

The MQSeries Workflow for OS/390 administration console accepts administration commands for starting and stopping systems and servers. You can also use it to display how many of server instances are running. For more information, see "Chapter 8. Administration console tasks" on page 83.

Execution server

The component that performs the processing of process instances at runtime. MQSeries Workflow allows multiple execution server instances to be started.

Program execution server

The program execution server (PES) manages all requests for programs to be executed on CICS or IMS service systems. These programs may be either MQSeries Workflow applications running in CICS or IMS using input and output containers or existing legacy programs that require a mapping routine to transform these containers to the programs call and reply parameter. Invoking legacy programs requires the definition of forward and backward mappings. The PES supports different invocation

types and mapping types. Connection information is stored in the PES directory. The PES provides a security mechanism to restrict program access to dedicated users.

Program execution server directory

The program execution server directory defines invocation types, mapping types, and the services where MQSeries Workflow program activities can be executed. It also contains information to map an MQSeries Workflow user ID to an OS/390 execution user ID. The PES directory must be updated when you add services, users, invocation types, and mapping types.

Invocation type

An invocation type specifies the type of invocation that is used by the program execution server to execute a request. This type is part of the program definition in the process model and must also be defined in the program execution server directory. An invocation type is uniquely associated with an invocation exit.

Mapping type

A mapping type specifies the type of mapping that is used by the program execution server to execute a legacy program. This type is part of the program definition in the process model and must also be defined in the program execution server directory. A mapping type is uniquely associated with a mapping exit.

Invocation exit

An invocation exit is the executable that is called by the program execution server to perform an invocation according to the invocation type specified in the program definition.

Mapping exit

A mapping exit is the executable that is called by the program execution server to perform mapping according to the mapping type specified in the program definition.

Program mapping

The program execution server provides a default program mapper. You can define mapping rules for legacy programs so that they can be invoked. How to write program mapping rules is described in *MQSeries Workflow for OS/390: Programming*. How to administrate program mapping is described in “Administering program mapping” on page 110.

Buildtime

The MQSeries Workflow Buildtime is used to define process models and system configurations. Buildtime runs on a Windows workstation, you will use it to define the services to be made available to the MQSeries Workflow activities. Buildtime exports the process models in a format that is known as MQSeries Workflow Definition Language (FDL). You can use Buildtime to define the number of instances of execution and program execution servers that are started when the system is started up. You will have to transfer the FDL file to your OS/390 system, and import it into the MQSeries Workflow for OS/390 database using the import tool. See “Chapter 9. Buildtime administration tasks” on page 93. For more information about Buildtime see *IBM MQSeries Workflow: Getting Started with Buildtime*.

ARM You can register the MQSeries Workflow for OS/390 administration server with the OS/390 Automatic Restart Manager to ensure that the MQSeries

Workflow for OS/390 system is automatically restarted in the event of an abnormal termination of the administration server.

CICS If you want to make CICS programs available to MQSeries Workflow activities, you may have to install and configure an MQSeries CICS bridge or the EXCI invocation type. For information about setting up CICS invocation types see “Customize CICS EXCI invocation” on page 39 and “Customize MQSeries CICS bridge invocation” on page 41.

DB2 DB2 databases are used as a repository for Workflow process models and to store the in-flight state of created and running process instances. Each MQSeries Workflow for OS/390 system group needs its own database, and contains one or more systems. The systems in a system group can reside on different OS/390 images, then they have to use the same database using DB2 Data Sharing. Multiple MQSeries Workflow for OS/390 system groups can share one DB2 subsystem.

IMS If you want to make IMS programs available to MQSeries Workflow activities, you may have to install and configure an MQSeries IMS bridge or the CPIC invocation type. For information about setting up IMS invocation types see “Customize IMS CPIC invocation” on page 44 and “Customize MQSeries IMS bridge invocation” on page 46.

MQSeries

MQSeries Workflow for OS/390 uses MQSeries message transportation, and requires a MQSeries for OS/390 queue manager that is a member of an MQSeries cluster.

RACF RACF is used to define the security for resources such as queues and programs. Various administration tasks require RACF settings to be defined or changed.

Note: This manual assumes that you are using RACF for your security. If you are using a different security system, you must apply the equivalent security access controls for your system.

WLM You can configure MQSeries Workflow for OS/390 to use the OS/390 Workload Manager to administer multiple instance server address space.

System trace

You can use the system trace facility for problem determination. For information about tracing see “The MQSeries Workflow for OS/390 system trace facility” on page 129.

Address spaces

During normal operation, you should only administer servers and systems using the administration console. On OS/390, servers run in address spaces. Several servers may run in the same address space, but for each server type a different address space is used. For a single-instance server like the administration server this means that it runs alone in an address space. Some of the servers are multiple-instance servers: If the workload requires it, additional server instances of this server type can be started. The number of server instances that can run in a single address space depends on the size of the instances. The maximum number of server instances that shall be started in one address space is a tuning parameter, which can be changed when tuning the performance (see “Changing the number of server instances per address space” on page 121). If more than this maximum number of server instances is started, additional address spaces will be used.

Administration in an MQSeries Workflow system

System administration is implemented using an administration component that controls and manages one MQSeries Workflow system within a system group. It provides vital management, control, security, and operational functions that govern the running of a particular selected system within a system group. The administration component is made up of an administration server and the administration console.

MQSeries Workflow has a hierarchical structure. The **domain** is the highest level in the hierarchy and may contain no more than one system group. Each **system group** is made up of one **system** which contains an administration server, one or more execution server instances, one scheduling server, one cleanup server, and zero or more program execution server instances. All Workflow systems running in the same OS/390 system have their own administration server, but are administered by the same administration console.

An MQSeries Workflow system has a tiered structure:

- **Tier 1 — Client tier**

Tier 1 contains the MQSeries Workflow system clients, application programming interfaces, and Buildtime. They use the MQSeries client and MQSeries Workflow APIs to connect with the second tier.

- **Tier 2 — Server tier**

Tier 2 can be split into two parts. The first part contains all the various MQSeries Workflow servers. This is the working center where all the scheduling, distribution, cleanup, administration, server communication, and execution is done. The administration server and administration console are located in this tier.

The second part contains the MQSeries Workflow DB2 database either on a local DB2 subsystem, or via a DB2 Data Sharing member. This holds system status and setup information for the complete system. The administration server accesses system tables within the database and returns the contents back to the administration console when requested to do so. The database is automatically updated by the administration server in response to system events.

The administration console is the user interface to the administration server and is used by a system administrator to request services from the administration server, see “Chapter 8. Administration console tasks” on page 83. For details on how to use the administration utility to administer the MQSeries Workflow system on AIX, and Windows NT, see *IBM MQSeries Workflow: Administration Guide*.

For further details about the system structure, see *IBM MQSeries Workflow: Concepts and Architecture*.

System administration client/server components

Every MQSeries Workflow system has an administration server. Using the administration console, any authorized system administrator can access an administration server within any specified system as long as those systems are members of system groups that are in the same domain.

Figure 5 on page 79 illustrates the implementation of the administration component within an MQSeries Workflow system. All administration components within a system group are implemented in a similar way.

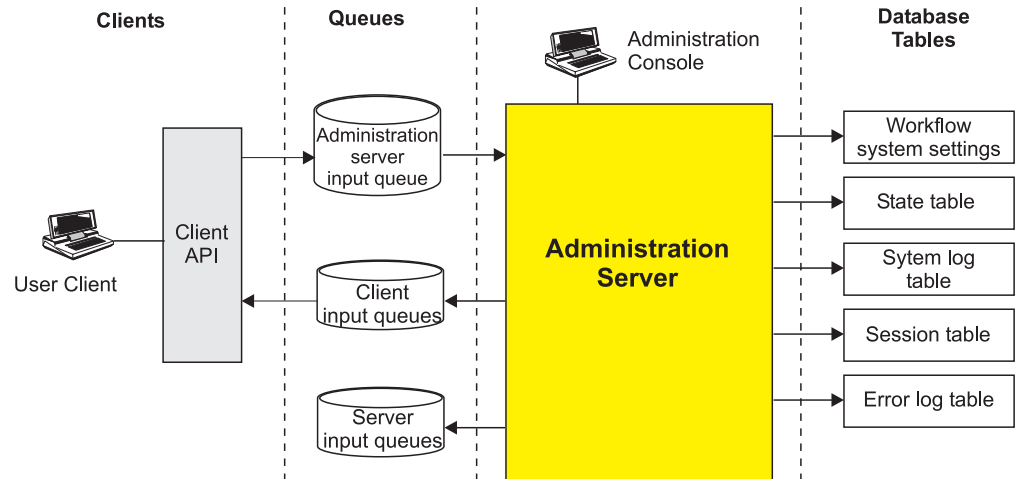


Figure 5. Implementation of the administration component in an MQSeries Workflow system

The administration server

The administration server is the working center of the administration component. It is responsible for the management of all components in an MQSeries Workflow system. It performs administrative functions in response to system administration requests, as well as, automatic internal functions that are transparent to the system administrator. The administration server communicates with all other components in an MQSeries Workflow system and is responsible for session management in the MQSeries Workflow system. It handles all logon requests and checks user identification, password, and authorization for a requested session.

The administration server is always the first component in an MQSeries Workflow system that is started. After you have started the administration server, you can start the system. The system must be shut down using the administration server. The administration server can be restarted while the system is running. Shutting down the administration server does not shut down the complete system.

The administration server sends messages via queues that are managed by MQSeries, and has access to various system tables held in the system database.

Queues

All components in the system receive messages from input queues that are managed by MQSeries. MQSeries is used to manage communications within an MQSeries Workflow system. The administration server uses MQSeries to send messages to all system server and client input queues. It maintains its own input queue from which all messages are received. The administration server uses boot queues for the start-up of the program execution server.

Database tables

The administration server accesses domain, system group, and system tables in the MQSeries Workflow database. The following lists the tables that can be accessed by the administration server:

- The administration server state table, which lists the administration server state properties, and the operational status of system servers.
- Property tables for all servers.
- The system properties table in which properties that determine the behavior of the system are contained

- The system group properties table in which properties that determine the behavior of the system group and some system properties are contained.
- The domain properties table contains properties that determine the behavior of the domain and some system group and system properties.
- A session table in which a session record is created for each authorized user after logon.

The administration console

The administration console provides a command line interface to the administration server. It allows the OS/390 administrator to start and stop MQSeries Workflow systems and servers, and to query how many server instances are running. The tasks that can be performed from the OS/390 administration console are described in “Chapter 8. Administration console tasks” on page 83.

Overview of administration tasks

To administrate MQSeries Workflow for OS/390 systems, servers, programs, and users you will have to use several different administration tools.

System and server administration tasks

The following table gives you an overview of the main system and server tasks that can be performed using the system and administration consoles.

Table 58. System and server administration tasks

Task	Buildtime	System console command	Admin console command
“Starting the administration server and administration console” on page 83		•	
“Stopping the administration server” on page 84		•	•
“Starting the system” on page 84			•
“Stopping the system” on page 85			•
“Restarting the system” on page 85			•
“Starting servers” on page 86			•
“Stopping servers” on page 87			•
“Restarting servers” on page 88			•
“Displaying the number of instances of a server” on page 89			•
“Hold queue commands” on page 89			•
“Switching servers between WLM and non-WLM mode by importing an FDL file” on page 154		•	•
“Switching servers between WLM and non-WLM mode using Buildtime” on page 94	•	•	•
“Chapter 11. Administering Servlets on the WebSphere Application Server” on page 117		•	

System console

Before you can administer a system, you must start the administration server for that system. You do this by issuing the start command on the system console. Having an administration server running makes the administration console available.

Administration console

You issue commands to start and stop systems and servers using the administration console. You can also use it to display how many server instances are running. The console commands are entered on the system console, but are forwarded by the administration console program to the administration server specified.

Program and user administration tasks

The following table gives you an overview of which administration tools and components are required for each administration task. The recommended sequence that the tools should be used are described in each task description.

Table 59. Program and user administration tasks: tool dependencies

Task	Buildtime	PES directory	RACF	Program mapping
"Defining process models" on page 93	•			
"Uploading process models to the host" on page 101				
"Importing and exporting process models" on page 101				
"Enabling an OS/390 program to be run as a program activity" on page 108	•	•	•	•
"Disabling a program" on page 109	•			
"Enabling an OS/390 program to run as a safe application" on page 109	•			
"Authorizing a user to access an OS/390 program" on page 109		•	•	
"Revoking a user's access to OS/390 programs" on page 110		•	•	
"Importing a program mapping definition" on page 110				•
"Enabling a program's mapping" on page 112	•	•		
"Disabling a program's mapping" on page 113	•			

Buildtime

You will use the MQSeries Workflow Buildtime tool to modify server and program properties in the process model definition. Exporting the process model creates a FDL file that must be uploaded to the mainframe, and then imported into the Workflow database. You can also define the number of instances of each server type that are to started when the system is started.

PES directory

You must modify the PES directory to define new services, invocation types, mapping types, or new users. After changing the PES directory, you must import it into the PES directory database.

RACF When you add programs or users, you have to use RACF (or an equivalent security program) to enable access to the necessary resources.

Program mapping

If you define or change a program mapping for a legacy application, you must run the import tool to update the program mapping database.

Chapter 8. Administration console tasks

Most MQSeries Workflow system administration is done using the administration console. This console allows tasks to be performed using command line calls. Before you can use the administration console, the administration server and console must be running as described in “Starting the administration server and administration console”.

The following sections describe the main groups of on-line administration tasks:

- “Administration server commands”
- “System commands” on page 84
- “Server commands” on page 86
- “Hold queue commands” on page 89

Note: Servers are started and stopped asynchronously. This means, for example, if you have issued the command to START the system, and then, before the system has fully started, you issue the command to STOP the system, the result can be that servers are still running. You can verify the actual system status using the DISPLAY command described in “Displaying all server instances in the system” on page 86.

Some administration console commands have a slightly different behavior when acting on server types that are managed by WLM. Where there are differences, they are noted in a separate subsection. The differences only apply to those multiple instance server types that are being managed by WLM (currently only the execution server can be WLM managed). For more information about using WLM, see “Part 3. Using OS/390 Workload Manager with Workflow” on page 133. You should administer the WLM managed servers using the normal administration console commands as described in “Server commands” on page 86.

Administration server commands

Starting the administration server and administration console

If necessary, start DB2, MQSeries *QueueManager*, CICS, or IMS.

Before you can issue any administration console commands, the administration server and console must be started. You start the administration server and the administration console by issuing the following command on the system console:

```
START UniqueSystemKey.AdminServerID [,ARMRE=ARMRestartElementNameSuffix]
```

This establishes the connection between the administration server ID and the system specified in *UniqueSystemKey*.

UniqueSystemKey

Your value specified during planning in Table 3 on page 11. It must be unique within the OS/390 image, and not more than 8 characters long.

AdminServerID

A name you will use to identify the administration server when you issue console commands. It must be unique within the OS/390 image, and not more than 8 characters long. As there is only one administration server per

Workflow system, it is recommended that you construct the name from the letters FMCA and some unique characters from the system name, for example FMCASYS1.

ARMRE=*ARMRestartElementNameSuffix*

This parameter is optional. If you specify the *ARMRestartElementNameSuffix* (see your value in Table 3 on page 11), the administration server is registered with the OS/390 Automatic Restart Manager service. In the event of an abnormal termination, ARM will automatically restart the administration server's address space on the same system. When the administration server terminates normally, it is de-registered from the ARM service.

For example, `START MQWFS1.FMCASYS1,ARMRE=MQWFS1`

Only one administration server can be started for each system. After executing this command you can start the system as described in "Starting the system".

Stopping the administration server

You can stop the administration server (without affecting any other running servers) by issuing the following command on the system console:

```
STOP AdminServerID
```

This has the same effect as the stop server command:

```
MODIFY AdminServerID,STOP ADM
```

```
AdminServerID
```

The ID that was specified when the administration server was started.

For example, `STOP FMCASYS1`.

When the last administration server on the OS/390 image is stopped, the administration console terminates.

System commands

You can start, stop, and display any MQSeries Workflow for OS/390 system from the administration console, but only if an administration server is running on that system. The system related tasks are described in:

- "Starting the system"
- "Stopping the system" on page 85
- "Restarting the system" on page 85
- "Displaying all server instances in the system" on page 86

Starting the system

Which and how many server instances will be started is specified in the server settings in Buildtime, see "Defining server properties" on page 93 for more information.

To start the MQSeries Workflow for OS/390 system, issue the console command:

```
MODIFY AdminServerID,START
```

or the short form

```
F AdminServerID,S
```

AdminServerID

The ID that was specified when the administration server was started. The *UniqueSystemKey* associated with this by the start command identifies the system that will be started.

For example: MODIFY FMCASYS1,START will start the system where the administration server FMCASYS1 is running

Note: The server start and stop time settings in Buildtime are ignored by MQSeries Workflow for OS/390.

When using WLM

When you start the system, the initial number of instances specified in the system's topology definition is ignored for any servers that are being managed by WLM. The server application environments are 'resumed', just as if the servers were started as described in "Starting servers" on page 86.

Stopping the system

To stop the MQSeries Workflow for OS/390 system, issue the console command:

```
MODIFY AdminServerID,STOP
```

or the short form

```
F AdminServerID,P
```

AdminServerID

The ID that was specified when the administration server was started. The *UniqueSystemKey* associated with this by the start command identifies the system that will be stopped.

For example: MODIFY FMCASYS1,STOP will stop the system where the administration server FMCASYS1 is running.

This stops all servers (except for the administration server) that are running on the Workflow system. How to stop the administration server is described in "Stopping the administration server" on page 84.

Restarting the system

Changes made to the machine profile or the environment variable file will only affect new server instances and tools that are started. If you want such changes to affect all running server instances then you must restart the whole system, including the administration server in the following sequence:

1. "Stopping the system"
2. "Stopping the administration server" on page 84
3. "Starting the administration server and administration console" on page 83
4. "Starting the system" on page 84
5. Then start extra server instances if required, see "Starting servers" on page 86.

which requires the command sequence:

```
MODIFY AdminServerID,STOP  
STOP AdminServerID  
START UniqueSystemKey.AdminServerID  
MODIFY AdminServerID,START  
MODIFY AdminServerID,START ServerType [INST(NumberOfInstances)]
```

Displaying all server instances in the system

To display the number of running instances of each server type in the MQSeries Workflow for OS/390 system, issue the console command:

```
MODIFY AdminServerID,DISPLAY
```

or the short form

```
F AdminServerID,D
```

AdminServerID

The ID that was specified when the administration server was started. The *UniqueSystemKey* associated with this by the start command identifies the system that will be displayed.

For example: MODIFY FMCASYS1,D will display the number of each server type running on the system where the administration server FMCASYS1 is running.

To display the number of instances of a particular server type, see “Displaying the number of instances of a server” on page 89.

Server commands

You can start and stop any MQSeries Workflow for OS/390 servers from the administration console. You can also query how many server instances are running. These tasks are described in:

- “Starting servers”
- “Stopping servers” on page 87
- “Restarting servers” on page 88
- “Displaying the number of instances of a server” on page 89

All server commands require *AdminServerID* that was specified when the administration server was started; this uniquely identifies the Workflow system that the command will be executed on. Server commands also require one of the following *ServerType* names to identify which server type the command applies to:

Table 60. Server types

<i>ServerType</i>	Server	Instantiation type
ADM	Administration server.	Single instance
SCH	Scheduling server	Single instance
CLE	Cleanup server.	Single instance
EXE	Execution server.	Multiple instance
PES	Program execution server.	Multiple instance

Note: The address spaces also use the *ServerType* as an identifier.

Starting servers

To start a given number of instances of an MQSeries Workflow server, issue the following command:

```
MODIFY AdminServerID,START ServerType [INST(NumberOfInstances)]
```

or the short form

```
F AdminServerID,S ServerType [INST(NumberOfInstances)]
```

AdminServerID

The administration server that is to start the server.

ServerType

The type of server to be started. You cannot start the administration server using this command, see “Starting the administration server and administration console” on page 83.

NumberOfInstances

This optional parameter specifies how many new instances of the server should be started. For single instance server types, the default is one. For multiple-instance server types, the default value is specified in the server settings in Buildtime.

Note: To verify the success of this command you can issue the display command:

```
MODIFY AdminServerID,DISPLAY ServerType RUNINSTANCE
```

For example, `MODIFY FMCASYS1,S PES` would start the program execution server with the number of instances specified in the process model.

Starting WLM managed servers

The command `MODIFY AdminServerID,START ServerType` delegates the starting of server instances to WLM. WLM determines how many instances are started, and when, depending on the workload demands. Specifying the `INST` parameter will generate an error.

If your servers are running in manual mode, then you must start the server instances manually as described in “Starting WLM-managed servers when WLM is in manual mode” on page 155.

Starting execution servers

To activate the MQSeries Workflow for OS/390 system at least one execution server must be running. It is possible to start multiple instances of the execution server to share the work load. For example, if you want to start two new execution servers running on the system where *AdminServerID* is running, issue the command:

```
MODIFY AdminServerID,START EXE INST(2)
```

or the short form:

```
F AdminServerID,S EXE INST(2)
```

Note: If the execution server is managed by WLM, see “Starting WLM managed servers”.

Starting program execution server instances

Starting the PES is no different to starting any other MQSeries Workflow server. For example, if you want to start 3 new instances of the PES, issue the command:

```
MODIFY AdminServerID,START PES INST(3)
```

Stopping servers

To stop all MQSeries Workflow for OS/390 server instances of a particular type, issue the console command:

```
MODIFY AdminServerID,STOP ServerType
```

or the short form

```
F AdminServerID,P ServerType
```

AdminServerID

The administration server that is to stop the server instances, effectively identifying the system.

ServerType

The type of server to be stopped.

For example, `MODIFY FMCASYS1,P EXE` would stop all execution server instances on the system where the administration server named `FMCASYS1` is running.

The server instances must complete the current transaction within a defined time window. If some server instances ignore the stop command, repeat the command. If this does not help, see “Cannot stop servers” on page 126.

Note: To verify the success of this command you can issue the display command:

```
MODIFY AdminServerID,DISPLAY ServerType RUNINSTANCE
```

Restarting servers

Occasionally, you may want to restart the instances of a particular server type. You can do this by simply issuing the stop server command, using the display command to verify that the server type has stopped, and then issuing the start server command.

Restarting the program execution server

There are special circumstances when you may need to restart the PES, for example:

- If you have modified an existing and currently activated program mapping definition in the process model, you must restart the PES. Doing this forces the mapping engine to use the new mapping settings. This situation is described in “Enabling a program’s mapping” on page 112.
- If PES directory caching is enabled as described in “Caching the PES directory at runtime” on page 108, a restart is required to force the cache to be refreshed.

To restart the PES, you simply issue the stop command and then the start command. For example, if your administration server ID is `FMCASYS1`, and you want 4 PES instances:

```
MODIFY FMCASYS1,STOP PES
MODIFY FMCASYS1,START PES INST(4)
```

Restarting the administration server

There are special circumstances when you may need to restart the administration server, for example:

- If you have modified the machine profile. For example:
 - To change the maximum number of server instances that will be started in a single address space.
 - To change the server queue disable time period.
- If you want to activate changes made to the server properties in the process model.

To restart the administration server, you simply issue the stop command and then the start command. There is no need to stop the system, the system can continue while the administration server is restarted.

For example, if your administration server ID is `FMCASYS1`, and your *UniqueSystemKey* is `MQWFS1` (see your value in Table 3 on page 11), then

```
MODIFY FMCASYS1,STOP ADM
START MQWFS1.FMCASYS1
```

will restart the administration server.

Restarting WLM managed servers

For WLM managed servers, the easiest way to restart server instances is to issue the WLM REFRESH command:

```
V WLM,APPLENV=ApplicationEnvironmentName,REFRESH
```

where *ApplicationEnvironmentName* is the appropriate value from Table 3 on page 11.

Displaying the number of instances of a server

You can find out how many server instances are currently running on a given system by issuing the console command:

```
MODIFY AdminServerID,DISPLAY ServerType [RUNINSTANCE]
```

or the short form

```
F AdminServerID,D ServerType RUNINST
```

AdminServerID

The name of the administration server (effectively a system) where you want to count the server instances.

ServerType

The type of server instances to be counted.

For example, `MODIFY FMCASYS1,DISPLAY EXE RUNINSTANCE` will display the number of execution server instances that are running on the system where administration server FMCASYS1 is running.

Note: To display the number of instances of all server types, issue the command `MODIFY FMCASYS1,DISPLAY` as described in “Displaying all server instances in the system” on page 86.

Hold queue commands

Hold queue commands are used to administer undelivered messages. If an input message for the execution server can not be processed, it is rolled back. If this happens several times, so that the retry limit is exceeded, the message is put into the execution server’s hold queue. If several consecutive messages are put into the hold queue of a server, the server is considered to be unreliable, and it is shutdown.

The messages in the hold queue can be displayed, deleted and replayed. Only messages which have been displayed before can be deleted or replayed, this prevents messages being removed from the hold queue by mistake. Before replaying a message, the problem that caused the message execution to fail should be fixed. When a message is replayed, it is reinserted into the servers input queue.

Note: Currently, only the execution server offers hold queue handling, so the *ServerType* parameter in the following commands can only have the value EXE.

Displaying number of messages in the hold queue

You can find out how many messages are currently in the hold queue of the specified server type by issuing the console command:

```
MODIFY AdminServerID,DISPLAY ServerType HOLDQDEPTH
```

or the short form:

```
F AdminServerID,D ServerType HQD
```

where

ServerType

the type of server to be displayed. For example, EXE.

Displaying messages in the hold queue

You must display the messages that are in the hold queue before you can replay or delete them. To display which messages are in the hold queue for a particular server type, issue the following command:

```
MODIFY AdminServerID,DISPLAY ServerType HOLDQ(NumberOfMessages|*)
```

or the short form:

```
F AdminServerID,D ServerType HQ(NumberOfMessages|*)
```

where

ServerType

The server type whose hold queue messages are to be displayed.

NumberOfMessages

Determines how many messages are displayed from the front of the hold queue, and in which format they are displayed.

NumberOfMessages=1 provides full details about the first message in the hold queue in the following format:

```
System:           System
Message:          ProgramFinished
Component:        Program execution agent
Number of failed replays: 1
Message content: ActImplCorrelID=actImplCorrelID
SessionID=OID(00000040000000000000000000000001)
```

NumberOfMessages=n for any number, **n**, greater than one, overview information about the first **n** messages is displayed in the following overview format:

System	Message	Component	#
SYSTEM	ProgramFinished	PEA	1
SYSTEM	ProgramFinished	PEA	0
...			

where # indicates the number of failed replays.

NumberOfMessages=* causes all messages in the hold queue to be displayed in the overview format.

Replaying messages from the hold queue

After a message has been displayed, it can be replayed. This means that the message is moved to the server's input queue. To replay any number of previously displayed messages issue the following command:


```
MODIFY AdminServerID,REPLAY ServerType HOLDQ(NumberOfMessages|*)
```

or the short form:

```
F AdminServerID,REPLAY ServerType HQ(NumberOfMessages|*)
```

where

ServerType

The server type whose hold queue messages are to be replayed.

NumberOfMessages

The number of messages to be replayed. If * is specified, all messages for this server type that have been displayed before will be replayed. If an integer value is specified, then this number of messages will be replayed (if they have previously been displayed).

Note: After issuing a REPLAY, you must wait for confirmation that it has completed, before issuing a REPLAY or DELETE command.

Deleting messages from the hold queue

After a message has been displayed, if it is not to be replayed, you can delete it from the hold queue by issuing the following command:

```
MODIFY AdminServerID,DELETE ServerType HOLDQ(NumberOfMessages|*)
```

or the short form:

```
F AdminServerID,DELETE ServerType HQ(NumberOfMessages|*)
```

where

ServerType

The server type whose hold queue messages are to be delete.

NumberOfMessages

The number of messages to be deleted. If * is specified, all messages for this server type that have been displayed before will be deleted. If an integer value is specified, then this number of messages will be deleted (if they have previously been displayed).

Note: Never use the DELETE command while replaying messages.

Chapter 9. Buildtime administration tasks

Some administration tasks, for example, adding new programs or defining server properties, need changes in the process model. For more information about the Buildtime tool see *IBM MQSeries Workflow: Getting Started with Buildtime*.

This chapter only describes the settings and actions that are specific to using Buildtime to perform administration tasks for MQSeries Workflow for OS/390.

Defining process models

You must use the Buildtime tool to define the programs and servers that are running as a part of MQSeries Workflow for OS/390. Each server and program in the system has a set of properties that can be changed in Buildtime.

Note: To avoid code page conversion problems when uploading FDL to the host, your Buildtime object names should conform to the guidelines that are described in “Appendix D. Naming and code page restrictions” on page 173.

The main administration tasks relating to Buildtime are:

- “Defining server properties” which includes:
 - Specifying the number of server instances to be started when the system is started.
 - Specifying the user support mode for the program execution server.
- “Defining program properties” on page 96
- “Defining the connection between a program activity and the PES” on page 100

After you have changed properties, and exported the process model, you will have an FDLfile that must be imported into the Workflow database, as described in “Importing and exporting process models” on page 101.

Defining server properties

Server properties that can be changed

The following OS/390 server properties can be modified in the process model.

Table 61. Server properties that can be changed

Property	Description	Initial default value
Number of instances	The number of servers that will be started when the system is started.	5
External Control	Determines whether WLM is used to manage resources. Before activating this option for the execution server or the program execution server, their application environments must be defined as described in “Chapter 15. Setting up WLM for MQSeries Workflow for OS/390” on page 145.	Inherited (None)
Context Information for External Control	Specifies the application environment name for the server type.	

Table 61. Server properties that can be changed (continued)

Property	Description	Initial default value
User support (on the PES properties / Support modes page)	Indicates whether the PES is able to execute programs using the <i>ServerUserID</i> or the ID of the user requesting the invocation. Agent selects the <i>ServerUserID</i> , Program selects <i>UserID</i> option.	Agent
Start mode	Only when this is set to DEFERRED/Automatic for the cleanup server, then Start time and Stop time can also be changed – otherwise these settings are ignored.	'Immediate' when started manually. 'Automatic' when in WLM mode.
Check_Interval	Specifies how often the administration server checks whether the cleanup server is still running.	300 seconds

Server properties that should not be changed

The following OS/390 server properties should not be modified.

Table 62. Server properties that should not be changed

Property	Description	Value
Name	Fixed name for the OS/390 program execution server.	PESERVER
Implementation support		External
Platform		OS/390
Attach mode		Local
Support mode		Safe

Server properties that are ignored on OS/390

The following OS/390 server property settings are ignored.

Table 63. Server properties that are ignored on OS/390

Property	Description	Default
Start mode	This setting is ignored for all servers on OS/390 except for the cleanup server. When this is set to DEFERRED/Automatic for the cleanup server – then Start time and Stop time below are active.	'Immediate' when started manually. 'Automatic' when in WLM mode.
Start time	This setting is ignored on OS/390 – except when Start mode is set to DEFERRED/Automatic for the cleanup server.	Automatic / Immediate
Stop time	This setting is ignored on OS/390 – except when Start mode is set to DEFERRED/Automatic for the cleanup server.	Duration = Forever

Switching servers between WLM and non-WLM mode using Buildtime

If you have enabled OS/390 Workload Manager support as described in “Chapter 15. Setting up WLM for MQSeries Workflow for OS/390” on page 145, you can let WLM manage the server address spaces for one or both of the multiple instance server types:

- Execution server
- Program execution server

The easiest way to switch between manual and WLM modes is described in “Switching servers between WLM and non-WLM mode by importing an FDL file” on page 154. The following steps describe how to do it using Buildtime:

1. Using Buildtime, locate the *General* page on the *Program execution server properties* and/or *Execution server properties*, as shown in Figure 6.

Note: You can find these windows:

- a. Starting from the *Network* page.
- b. Open the *Domain* and *System Group* to find the *System*, and click on it with the right mouse button.
- c. Display the *System Properties* window.
- d. Select the *Server* page.
- e. Clear the inherited box, and click on the *Complete ... server settings* button for the server type that you want to switch.

Then:

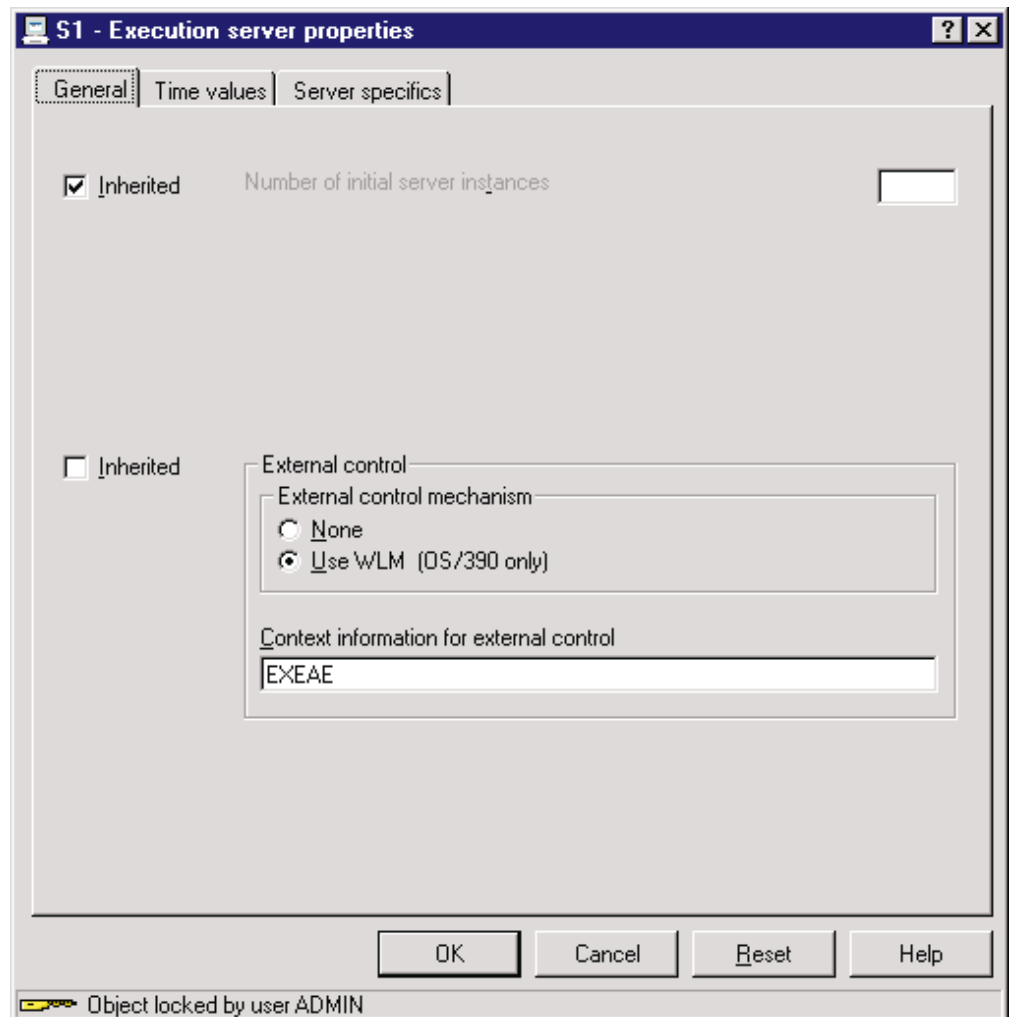


Figure 6. Execution server properties: General page

2. Set *External control* to the option *Use WLM* or *None* (clear the *Inherited* option first)

3. If you are activating WLM, you must also enter the name of the application environment for the server type in the *External Control: Context Information* field. This is the name that you defined when you created the WLM service definition as described in “Application environment” on page 150.
4. Import the new settings into the runtime database.
5. Restart the administration server to activate the changes.

Defining program properties

Every OS/390 program that is to be executed as part of a MQSeries Workflow process activity must be defined in the process model. The following screen-shots show and describe the pages and parameters that are required for defining an OS/390 program in Buildtime.

A program is defined by its name. You must also specify any optional input and output data structures, specific settings for the program itself, and the platform that the program runs on.

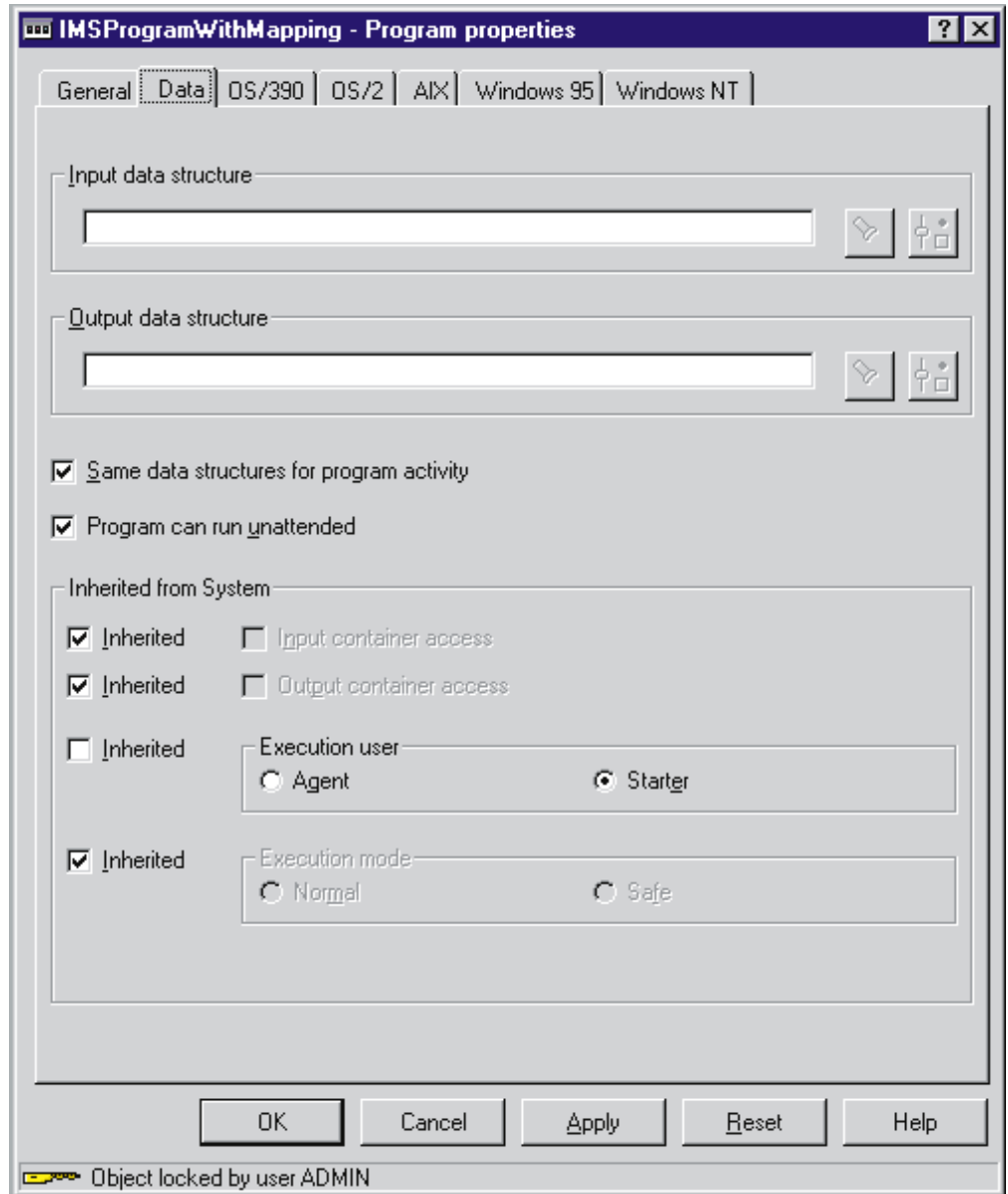


Figure 7. Program properties: Data page

Figure 7 shows the program properties data page for the an example program named IMSProgramWithMapping. The option **Execution user = Starter** means that the PES has to execute the program using the execution user ID of the starter of the activity. This option requires the PES setting **User support = Starter**. In this case, the PES has to map the Workflow user ID of the starter of the request to an execution user ID known to OS/390. See “Adding a new service definition and the related user resolution information” on page 106.

For more information about these settings, see “Program execution security” on page 114.

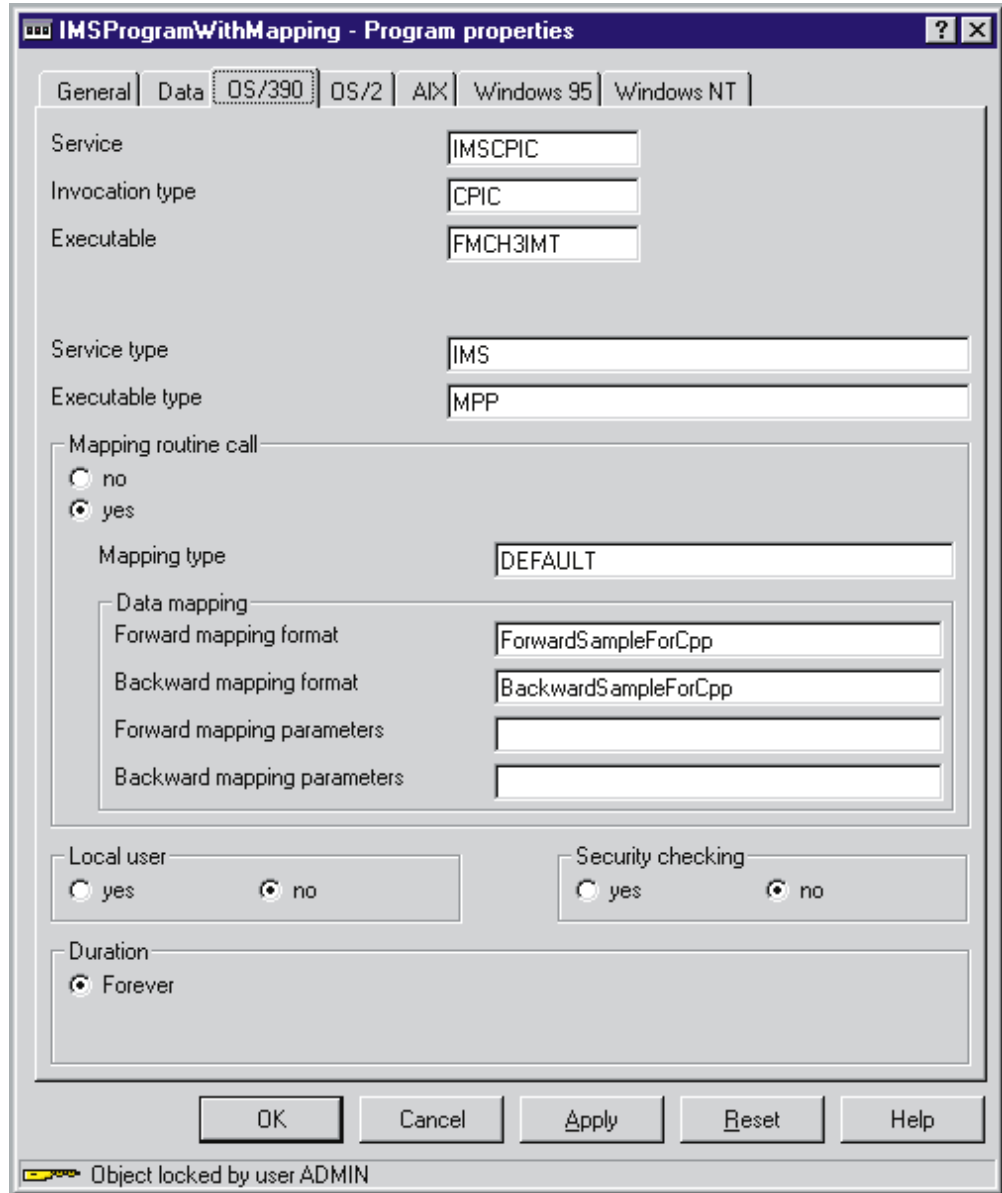


Figure 8. Program properties: OS/390 page

You must complete this screen to define an OS/390 program. The following program property settings are important for correct execution.

Table 64. Program properties: OS/390 page settings

OS/390 program property	Description
Service	The service name is the logical name of the service system where the program is executed. This setting is mandatory. This name may be up to 8 characters long, uppercase only, first character (A..Z,\$,#,@), other characters (A..Z,0..9,-,\$,#,@). The value entered must match a service name in the PES directory. The service must be defined for the invocation type that is used.
Invocation type	This defines the logical name of the invocation type that is used to invoke the program. This setting is mandatory. This name may be up to 8 characters long, uppercase only, first character (A..Z,\$,#,@), other characters (A..Z,0..9,-,\$,#,@). The value entered here must match an invocation type that is defined in the PES directory.

Table 64. Program properties: OS/390 page settings (continued)

OS/390 program property	Description
Executable	The name of the program to be executed, as defined to the service system. This setting is mandatory. This name may be up to 8 characters long, uppercase only, first character (A..Z,\$#,@), other characters (A..Z,0..9,-,\$#,@).
Service type	This defines the type of service system on which the program runs. The value specified here must match the service type that is specified in the invocation section in the PES directory, for the invocation type that is specified in this page. This setting is mandatory.
Executable type	This defines the type of executable. For CICS programs it should be set to DPL. For IMS programs it should be set to MPP or IFP. This setting is mandatory.
Mapping routine call	If set to no , the PES will not call the mapping routine for invocations of this program. If set to yes , the PES will call the mapping routine for invocations of this program. It uses the mapping formats and parameters that are specified below.
Mapping type	This defines the logical name of the mapping type. If mapping is used for the program, the name entered here must match a type specified in the mapping section in the PES directory. The mapping type supplied by IBM is called 'DEFAULT'. If Mapping routine call = yes , then this setting is required.
Forward mapping format	This defines the logical name of the forward mapping that is to be used to map the contents of the program's input container. The name entered here must match the name of the forward mapping definition that is imported into the mapping database. For more information about creating program mappings, see <i>MQSeries Workflow for OS/390: Programming</i> .
Backward mapping format	This defines the logical name of the backward mapping that is to be used to map the legacy application data. The name entered here must match the name of the backward mapping definition that is imported into the mapping database.
Forward mapping parameters	These parameters are only required if the mapping uses user-types that require mapping parameters. For more information about user-types and creating program mappings, see <i>MQSeries Workflow for OS/390: Programming</i> .
Backward mapping parameters	These parameters are only required if the mapping uses user-types that require mapping parameters. For more information about user-types and creating program mappings, see <i>MQSeries Workflow for OS/390: Programming</i> .
Local user	If set to yes , the OS/390 program will be executed under the OS/390 user ID associated with the calling MQSeries Workflow user ID. This mapping is defined in the service section of the PES directory. If set to no , the OS/390 program will be executed under the <i>ServerUserID</i> . In this case, no user resolution information is required in the PES directory for this service.
Security checking	If set to yes , the PES security routine will be called for each invocation. In this case a security profile must be defined as described in "Program security" on page 116.
Duration	For OS/390 programs, this value is set to Forever.

Defining the connection between a program activity and the PES

After an OS/390 program has been defined, it must be associated with a program activity, and the program execution server. This is done on the screen that is shown in Figure 9. You can reach this Buildtime screen in the following way:

1. Select the process containing the activity that should execute the OS/390 program.
2. Open the process diagram.
3. Select activity and open its **properties** notebook.
4. Select the notebook's **execution** page

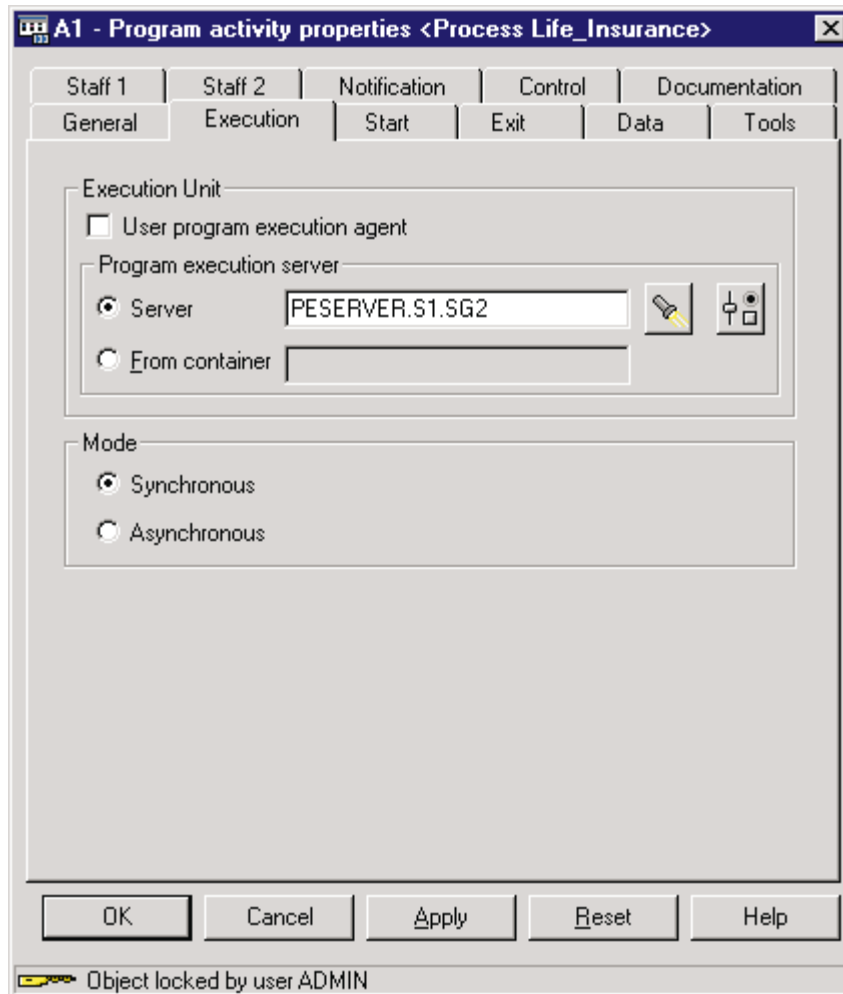


Figure 9. Program activity properties: Execution page

The information you enter on this screen defines the connection between the program and the program execution server. You must perform the following:

1. Clear the **User program execution agent** check box.
2. Click on the flashlight button to find the program execution server and select PESERVER.
3. Select the **Synchronous** mode.
4. Press **OK** then close the process diagram and save the process.

Uploading process models to the host

After exporting the process model information from the Buildtime database into an FDL file, you must upload it to the host before you can import the FDL file into the Workflow database. Ideally, inter-platform character conversion should be performed automatically during the upload process. You can use FTP or any other text transfer method to upload the FDL file to the host providing the code page conversion does not corrupt the data. Only if your transfer method corrupts the data during the upload process, should you upload your FDL file as a binary image, and then use the tool described in “Appendix E. FDL code page conversion tool” on page 175.

Note: To avoid problems with code page conversion during the upload process, your Buildtime object names should conform to the guidelines that are described in “Appendix D. Naming and code page restrictions” on page 173.

Importing and exporting process models

To activate the modified properties, you will need to import the uploaded FDL process model into the MQSeries Workflow for OS/390 database. This is done using the import/export tool, see “Using the FDL import/export tool”.

You can also use the import/export tool to do the following:

- Translate workflow definitions from Buildtime.
- Translate an existing process model in the Workflow database.
- Import an FDL file that you created outside of MQSeries Workflow.
- Export Workflow definitions from the Workflow database into an FDL file.

For more information about the export and translation options, see “FDL import/export tool’s syntax” on page 177, “FDL export examples” on page 181, and “Translate examples” on page 182.

Using the FDL import/export tool

If you define or change your process model definition in Buildtime, you will need to import it into your Workflow database. If you want to export a single object, or all workflow objects from your Workflow database you can use the export option of the FDL import/export tool.

The following describes how to use the import/export tool named FMCH0IBA:

1. Customize the JCL *CustHLQ.SFMCCNTL(FMCHJRIF)*
 - a. Specify the options that the import tool should use (see “Options for the import / export tool” on page 179)
 - b. Specify the input and output files using the predefined DD-names FMCIIMP, FMCIEXP, FMCICMD, and FMCILOG, as illustrated in “FDL import examples” on page 180.
 - c. If you want to use a specific log file instead of SYSOUT you will need to specify a data set for the DD-name FMCILOG.
2. Submit the JCL *CustHLQ.SFMCCNTL(FMCHJRIF)*

Chapter 10. Program execution

The program execution server (PES) manages program execution requests for programs running on external services like CICS or IMS. These programs may be legacy programs or may use MQSeries Workflow APIs. Legacy programs require mapping to transform Workflow data containers to (and from) the format and representation of parameters expected by the existing application. Programs invoked by the PES must conform to a request-reply model. The PES has a component based structure as shown in Figure 10 on page 104.

Invocation and mapping

Programs are executed on external services that the PES connects to via an invocation exit. Such an exit is based on an invocation protocol like CICS External Interface (EXCI), IMS/APPC or the MQSeries CICS and IMS Bridges. Each invocation exit is uniquely identified by the PES by an invocation type. Similarly, program mapping is performed by program mapping exits. Each mapping exit is uniquely identified by the PES by a mapping type. You can define user exits for mapping and invocation. The external interfaces these exits have to conform to are described in *MQSeries Workflow for OS/390: Programming*.

PES directory

Which invocation and mapping exits can be used by the PES at runtime is defined in the PES directory. This directory is the link between the Workflow program definitions specified in the process model and the components and resources necessary to run the program successfully. For instance, if a program specifies an invocation type EXCI, the directory must contain a definition for this invocation type. For more information about the PES directory, see “Administering the Program Execution Server directory” on page 105 and “Appendix A. Program Execution Server directory” on page 161.

Security

The PES can invoke programs on behalf of different MQSeries Workflow users. A Workflow user ID must be resolved to an execution user ID known to OS/390, as described in “Adding a new service definition and the related user resolution information” on page 106. The PES can perform security checks to ensure that only authorized users can run a program, as described in “Program security” on page 116.

Runtime properties

The PES is a multi-instance server, with each server instance running as a single task. Server instances can process both synchronous and asynchronous invocation types. A server instance is blocked while it processes a synchronous request. For this reason it is important to ensure that there are enough PES instances to handle the request workload. Asynchronous requests are split into a request part and a reply part that are correlated to fulfill the request. All program execution server instances share the same external resources.

Error handling

The program execution server returns error indications whenever possible. Otherwise error notifications are sent to the administration server.

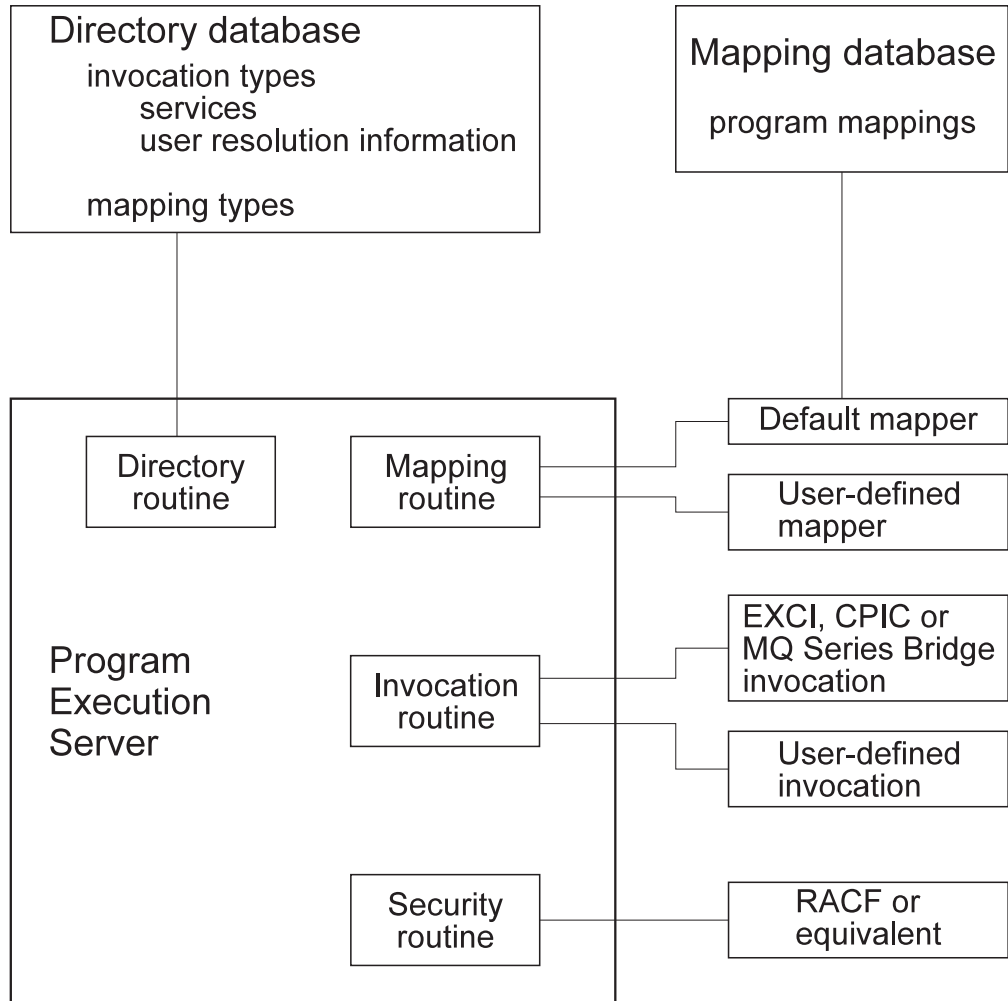


Figure 10. OS/390 Program execution server: component structure

The PES processes a synchronous request in the following way:

1. Analyze the request.
2. Locate the invocation type and service in the directory.
3. For a legacy program locate the mapping type in the directory.
4. If the program has to be run on behalf of a specific user, get the connection information from the directory and the execution user ID.
5. Check the security requirements for service, invocation type, the program to be executed and the execution user ID.
6. For a legacy program call the mapping routine to map input container to program parameter data.
7. Call the invocation routine to handle the request.
8. For a legacy program call the mapping routine to map output parameters to the output container.
9. If none of the above step has failed, the PES returns a completion message.
10. If any of the above steps fails, and the error is considered recoverable, the PES returns an error indication. For a non-recoverable errors the PES instance will terminate.

Invocation types supported

Invocation types may be defined as asynchronous or synchronous invocation exits. The IBM supplied invocation types are EXCI and MQCICS for CICS, CPIC and MQIMS for IMS. The definitions for these types are contained in the directory template source file provided by IBM. The types of CICS and IMS programs that can be executed depends on the invocation protocol used by the invocation type.

CICS program types supported

The PES supports the invocation of CICS DPL programs. CICS 3270 applications and CICS transactions are not supported.

IMS program types supported

The PES supports the invocation of IMS MPP and FastPath programs. Each program has to have a single input message and a single, predefined output message. The execution of IMS conversations is not supported. A sequence of non-conversational transactions can be modeled as a sequence of separate program execution requests.

Mapping types supported

MQSeries Workflow for OS/390 supplies a program mapping type named DEFAULT which should be able to handle most mapping requirements. A user type can be defined to handle special cases that are not covered by the DEFAULT mapping types' interface types. For more information about how to define mappings and user types, see *MQSeries Workflow for OS/390: Programming*. The mapping type to be used for each program is defined in the process model as shown in Figure 8 on page 98.

User-defined invocation and mapping types

You can extend the program execution server by defining your own mapping and invocation types, and their corresponding exits. These user-defined types may be used instead of an IBM supplied type, or in parallel. These exits are defined in *MQSeries Workflow for OS/390: Programming*.

If you define your own invocation and mapping types, they must be defined consistently in the process model and in the PES directory, as explained in more detail in "PES directory dependencies on the process model's OS/390 program definitions" on page 164.

Administering the Program Execution Server directory

The PES directory provides persistent runtime information for the OS/390 program execution server. It is a DB2 database, and it must be located in the same DB2 subsystem as the MQSeries Workflow for OS/390 database. The directory is used by the PES as a read-only database. The directory may be updated while the server is running. By default, any data retrieved from the directory is not cached, for more information about caching, see "Caching the PES directory at runtime" on page 108.

The directory is the connecting element between the program properties specified in the process model and the existing OS/390 services. It contains information about invocation types, and the services that can be called using each invocation type. It also defines the users that can use a service, and the mapping types. The PES directory's structure, template, and dependencies are described in "Appendix A. Program Execution Server directory" on page 161.

In order to define new invocation types, mapping types, or services, you must add new definitions to your current PES directory source file as described in the following sections. After you have changed your PES directory source file, you will have to activate it by importing it into the PES directory database. This is done using the PES directory import tool, see “Importing the PES directory” on page 107.

You can also use the import tool to perform the following:

- Update existing service definitions.
- Delete existing service definitions.
- Delete the entire contents of the PES directory database.

For more information about the possible options, see:

- “Appendix B. The PES directory import tool’s syntax and semantics” on page 167, and
- “PES directory import examples” on page 167.

Adding a new service definition and the related user resolution information

To add a new service definition, you have to update the PES directory source as provided in *CustHLQ.SFMCDATA(FMCHEDTP)*, then import it as described in “Importing the PES directory” on page 107. For more information about the PES directory structure see “Appendix A. Program Execution Server directory” on page 161.

The following example shows how to add a second service definition to an existing EXCI invocation type in your PES directory source file. In the example below, a service system named CICSEXC2 that is reached through EXCI is defined.

```
(INVOCATION1SERVICE<m>)
  type           =CICS
  name           =CICSEXC2
  connectionParameters =APPLID=<applid>;TRANSID=CSMI
  user           =INVOCATION1SERVICE1USER

; user section of PES Directory

(INVOCATION1SERVICE<m>USER1)
  userID         =<user1>
  executionUserID =<xxxxxxxx>
(INVOCATION1SERVICE<m>USER2)
  userID         =<user2>
  executionUserID =<xxxxxxxx>
```

1. In your version of the PES directory source, copy an existing CICS service definition `INVOCATION1SERVICE<m>`.

Note: The “PES directory template” on page 163 shows the template that is provided in *CustHLQ.SFMCDATA(FMCHEDTP)*.

2. Replace `<m>` with the number of the service inside the invocation section. For example, if you have already defined two service definitions for the first invocation type during customization, you should use the number three.
3. Set the value for `name` to the name of the service. For example, `CICSEXC2`.
4. For the `CICSEXC2` service, set the `connectionParameters` value for `APPLID` to the application ID.

- To add user resolution information, substitute `<user1>` with the MQSeries Workflow user identification used to start the program and `<xxxxxxx>` with the execution user identification for this user as defined in RACF.
Each additional user may be added by appending a corresponding `INVOCATION1SERVICE<m>USER<n+1>` section containing `userID` and `executionUserID`.

Additional CICS service systems may be added by appending an `INVOCATION1SERVICE<m+1>` section and completing it in the same way as described above.

Adding an IMS system that is reached by CPIC is similar, except that you have to add CPIC connectionParameters: `netid`, `luname`, and `mode`. Additional IMS service systems may be added by appending an `INVOCATION2SERVICE<m+1>` section and completing it in the same way as described above.

Adding a user-defined invocation type

To add a new invocation type to the PES directory you need to:

- Copy an existing invocation section including the service and user sections.
- Increase the running suffix numbers of the invocation for all sub-sections. For example, if you copied the section `(KEYTOINVOCATION5)`, change it and all subsequent keys that refer to this new invocation type to `(KEYTOINVOCATION6)`.
- Then update

```
(KEYTOINVOCATION6)
type                = <your invocation type>
exitName            = <your invocation exit dll name>
exitParameters     = <parameters needed by your invocation exit>.
```

- Add or change the service definitions according to the new invocation section:

```
(INVOCATION6SERVICE1)
type                = <service type>
name                = <service name>
connectionParameters = <connection parameters as needed by the new invocation>
user                = < ... user section ...>
```

- Add or change the user related information as required.

Adding a user-defined mapping type

To add a new mapping type to the PES directory you need to:

- Copy the last existing mapping section `KEYTOMAPPING<m>`.
- Increase the running number of the mapping section, for instance if you copied `(KEYTOMAPPING1)`, use `KEYTOMAPPING2` for all subsequent keys referring to this new mapping type.
- Then update

```
(KEYTOMAPPING2)
type                = <your mapping type>
exitName            = <your mapping exit dll name>
exitParameters     = <parameters needed by your mapping exit>
```

Importing the PES directory

After changing the PES directory source file you need to import it into the PES directory runtime database. You do this using the `FMCH1PIT` tool in the following way:

- Customize the JCL `CustHLQ.SFMCCNTL(FMCHJPIF)`

- a. Specify the **options** the import tool should use. These options are described in “Appendix B. The PES directory import tool’s syntax and semantics” on page 167.
 - b. Specify the **input file** using the predefined DD-name FMCDIMP.
 - c. If you want to use a specific **log file** instead of SYSOUT you must specify a data set using the predefined DD-name FMCDLOG.
2. Submit the JCL *CustHLQ.SFMCCNTL(FMCHJPIF)*

Caching the PES directory at runtime

To improve performance, the content of the PES directory may be cached in the program execution server at runtime. Directory caching is enabled through a system setting, `PESDirectoryInCache`, in the machine profile. If set to 1 caching is enabled. The default is no caching (zero). Both, connection parameters and user information are cached. The cache is built up dynamically.

For more information about making changes to the machine profile, see “Appendix H. Machine profiles” on page 193.

Refreshing the PES directory cache

The contents of the cache are only updated when program execution server instances are started. This means that changes made to the PES directory only become activated after all running program execution server instances are stopped and restarted. If WLM is managing the PES address spaces, then the PES application environment must be refreshed.

Administering programs

Program administration consists of the following tasks:

- “Enabling an OS/390 program to be run as a program activity”
- “Enabling an OS/390 program to run as a safe application” on page 109
- “Disabling a program” on page 109
- “Authorizing a user to access an OS/390 program” on page 109
- “Revoking a user’s access to OS/390 programs” on page 110

Enabling an OS/390 program to be run as a program activity

To enable an OS/390 program to run as an MQSeries Workflow program activity you need to perform the following tasks:

1. “Defining program properties” on page 96
2. “Creating a program mapping”
3. “Defining a security profile” on page 109

Creating a program mapping

If you want to invoke an OS/390 legacy program as an activity, you may have to create a program mapping. The program mapping transforms between the different format and data representations that are used by MQSeries Workflow and the invoked program. For more information about creating a program mapping, see *MQSeries Workflow for OS/390: Programming*.

If you have created a program mapping, you will have to activate the mapping in the process model. This task is described in “Enabling a program’s mapping” on page 112.

Defining a new program in the process model

You must define the new program in the process model using the MQSeries Workflow Buildtime program. Then you must import the process model FDL file into the Workflow database. These tasks are described in the following:

1. “Defining program properties” on page 96
2. “Uploading process models to the host” on page 101
3. “Importing and exporting process models” on page 101

Defining a security profile

If you defined the program in Buildtime with the option **Security checking=Yes**, then you must define a security profile for service, invocation type, executable and execution user identification as described in “Program security” on page 116.

Enabling an OS/390 program to run as a safe application

When an IMS or CICS program runs as a safe application, each invocation request will cause it to be executed once and only once, or not at all. Safe applications are executed in the same transactional context as the program execution server request transaction. This uses the OS/390 Resource Recovery Service (RRS).

For a program to run as a safe application, the following conditions must be satisfied:

1. The program must not issue its own RRS commit and RRS rollback calls.
2. The program must be invoked using a transactional invocation such as EXCI or CPIC.
3. The PES must be defined as supporting safe mode in the process model — this is the default setting.
4. The program must be defined as using **Execution mode=Safe** in the process model.

This makes the PES call the program in a transactional context using the invocation type specified in the programs’ external settings definition in the process model.

Disabling a program

You can disable a program by removing its definition from the process model, exporting the process model as an FDL file, and then importing it into the Workflow database.

Authorizing a user to access an OS/390 program

If a program has been defined in the process model with **Security routine call** set to **yes**, and **Local user** set to **yes** (as shown in Figure 8 on page 98), then every MQSeries Workflow user ID that is to be able to start the corresponding activity must be specified in the PES directory, and associated with a valid execution user ID. You must provide this user resolution information for the service where the program is executed.

To authorize a user to access an OS/390 program you must do the following:

1. If the program has no security profile, you must create one, as described in “Program security” on page 116.
2. Authorize the user by giving read access to the security program profile.

3. Update the PES directory entries for the service and the invocation type, adding the MQSeries Workflow user ID and the corresponding execution user ID, as described in “Adding a new service definition and the related user resolution information” on page 106.
4. Reload the PES directory as described in “Importing the PES directory” on page 107.

Revoking a user’s access to OS/390 programs

There are two possible ways to revoke a user’s access to an OS/390 program:

1. By removing their execution user ID’s access to the security profile for the program.
2. By deleting the user’s ID mapping associated with the service in the PES directory, then reloading the PES directory.

If you want to revoke a user’s access to all OS/390 services, mappings may exist for that user’s MQSeries Workflow user ID in several service sections in the PES directory. You should ensure that all of them are commented out or deleted.

Administering program mapping

The program execution server uses a program mapping exit to transform the format and representation of parameters in data containers so that it can be accepted by existing IMS and CICS applications. This allows you to define MQSeries Workflow processes that invoke legacy applications on the mainframe, without having to modify the legacy application.

MQSeries Workflow for OS/390 offers a mapping type named DEFAULT which can be used to convert and translate data between legacy applications and MQSeries Workflow. You can write your own exit to perform this conversion, providing it conforms to the MQSeries Workflow for OS/390 mapping exit interface. Mapping exits can be used in parallel to the default mapping exit or replace the default mapping exit. The invocation and reply data can be mapped separately by defining the interfaces and structure and connect them to each other. You can find more information about creating a program mapping in *MQSeries Workflow for OS/390: Programming*.

After you have created a program’s mapping definitions:

- You must import the mapping definitions into the mapping database as described in “Importing a program mapping definition”.
- A program mapping in the mapping database only becomes active after the mapping has been enabled as described in “Enabling a program’s mapping” on page 112.

Importing a program mapping definition

After you have created a program mapping definition you must import it into the mapping database. If the corresponding program mapping definition already exists, and is already active, importing new definitions will immediately affect this mapping. If you have modified mapping definitions in the mapping database, you must restart the PES, as described in “Restarting the program execution server” on page 88.

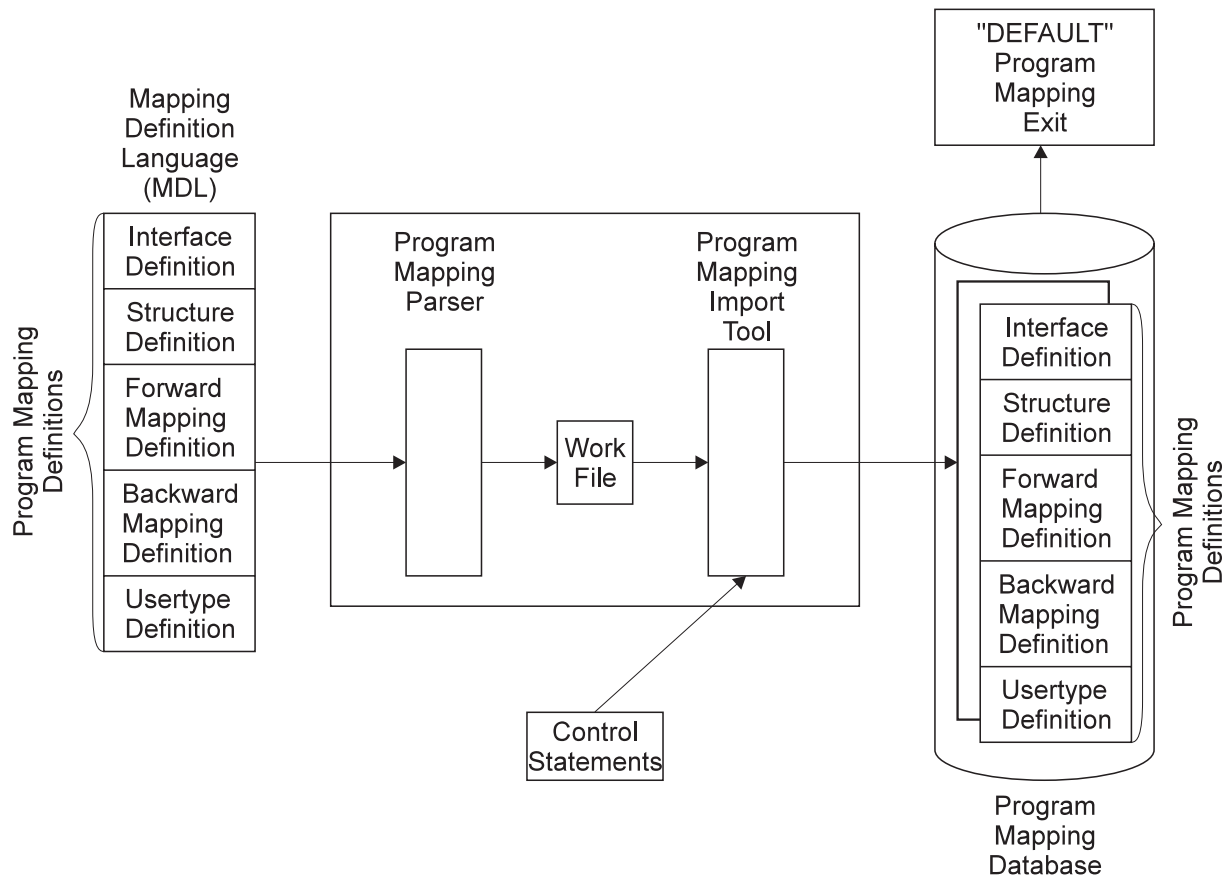


Figure 11. Program mapping definition process and components

To import a new program mapping definition, or to update an existing mapping definition you must do the following:

1. Create the program mapping definitions as described in *MQSeries Workflow for OS/390: Programming*.
2. Customize a copy of *CustHLQ.SFMCNTL(FMCHJMPR)*, so that the first DD FMCIN uses the new mapping definitions. FMCHJMPR contains statements similar to this:

```

/*****
/**
/** Description:
/** Invoke the default program mapper parser and import tool
/**
/*****
/**
//PROCLIB JCLLIB ORDER=(CustHLQ.SFMCPROC)
/**
/** Invoke Program Mapping Parser
/**
//FMCMPRS EXEC PROC=FMCHPBAT,PROGRAM=FMCH1XMP,
// PARM='ENVAR("_CEE_ENVFILE=DD:FMCHENV")/'
/**** Mapping Definitions
//FMCIN DD DSN=CustHLQ.SFMCDATA(FMCHMDL),DISP=SHR
/**** Work File
//FMCOUT DD DSN=&&BIN,DISP=(NEW,PASS),
// DCB=(RECFM=U,BLKSIZE=6144,LRECL=0),
// SPACE=(CYL,(2,2))
/**** Listing
//FMCLST DD SYSOUT=*
/**

```

```

/* Invoke Program Mapping Import Tool
/*
//FMCRMUTL EXEC PROC=FMCHPBAT,PROGRAM=FMCH1XMU,COND=(8,LE),
//          PARM='ENVAR("_CEE_ENVFILE=DD:FMCHEENV")/'
//**** Work File
//FMCCIN DD DSN=*.FMCRMPRS.FMCHPBAT.FMCCOUT,DISP=(SHR,DELETE)
//**** Program Mapping Import Tool Control Statements
//FMCCCTL DD DSN=CustHLQ.SFMCCDATA(FMCCHEMCT),DISP=SHR
//**** Listing
//FMCCCLST DD SYSOUT=*
/*

```

3. Customize a copy of *CustHLQ.SFMCCDATA(FMCCHEMCT)* which contains the control statements for the utility so that the parsed mapping definitions are created. Update DD FMCCCTL to use the new control statements.
4. Run your copy of FMCHJMPR.

Note: You may find it helpful to view the sample mapping definition *CustHLQ.SFMCCDATA(FMCCHEMDL)* and the sample control statements *CustHLQ.SFMCCDATA(FMCCHEMCT)*.

Return codes

The program mapping parser and import tool can return the following return codes:

Table 65. Program mapping parser and import tool's return codes

Value	Description	Effect of modifications to the database (import tool only)
0	Successful execution	Any database modifications have been completed.
4	Warning	
12	Error	The import tool has made a rollback of the transaction. The database remains unchanged.
16	Severe error	

Enabling a program's mapping

To make a program mapping active, you must modify the process model definition, turning the mapping on and specifying the necessary mapping names and parameters:

1. Enable the mapping in the process model using Buildtime:
 - a. Locate (or create) the program properties definition for the program that requires the program mapping, then set the following properties as described in "Defining program properties" on page 96:
 - 1) **Mapping routine call = yes**
 - 2) **Mapping type = DEFAULT**
 - 3) **Forward mapping format = name of the forward mapping definition**
 - 4) **Backward mapping format = name of the backward mapping definition**
 - 5) If the mapping uses user-types that require mapping parameters, they should be specified in the fields **Forward mapping parameters** and **Backward mapping parameters**.
2. Upload the new process model to the host, as described in "Uploading process models to the host" on page 101.
3. Import the process model on the host, as described in "Using the FDL import/export tool" on page 101.

4. If mapping definitions were modified, then restart the PES as described in “Restarting the program execution server” on page 88.

Disabling a program’s mapping

A program mapping can be disabled by doing the following:

- Disable the program mapping in the process model using Buildtime by doing the following:
 1. Set **Mapping routine call = no** in Figure 8 on page 98.
 2. Upload the new process model to the host, as described in “Uploading process models to the host” on page 101.
 3. Import the process model on the host, as described in “Using the FDL import/export tool” on page 101.

Deleting a program mapping definition

When program mappings have been disabled, the corresponding definitions should also be deleted from the program mapping database. To delete program mapping definitions, you must do the following:

1. Create a member for the program mapping import tool which contains delete statements for the program mapping definitions which should be deleted. For more information, see “Appendix C. Program mapping import tool syntax” on page 169.
2. Customize a copy of *CustHLQ.SFMCNTL(FMCHJMPR)* so that the member created in step 1 is used in the FMCCTL DD-statement.
3. Submit your copy of FMCJMPR.

Note: The program mapping import tool checks whether the program mappings which should be deleted are no longer referenced anywhere in other program mapping definitions. If they are still used the deletion will fail.

Enabling a mapping type

To activate a new program mapping type, you must perform the following steps:

1. Create a mapping exit DLL as described in *MQSeries Workflow for OS/390: Programming*.
2. Provide the mapping exit DLL to MQSeries Workflow for OS/390. Either copy the DLL into the data set *CustHLQ.SFMCLOAD*, or concatenate the data set name, where the mapping exit DLL is stored, to SFMCLOAD.
3. Define the new mapping exit in the PES directory. Replicate the entry called (KEYTOMAPPING1) in the PES directory template, choose the next free number (for example KEYTOMAPPING2) and choose a new type name other than DEFAULT, insert the correct exit name (DLL name) and optionally provide exit initialization parameters (exit parameters). Definitions needed for sample mapping exit (See *CustHLQ.SFMCsrc(FMCHSMEX)*):

```
(KEYTOMAPPING2)
type           =SAMPLE
exitName       =SAMPEXT
exitParameters =LONG=L FLOAT=F
```

4. Import the new PES directory definitions as described in “Importing the PES directory” on page 107.
5. Enter the new mapping type name in the process model definition of the OS/390 program. Replacing the default mapping type DEFAULT as shown in Figure 8 on page 98.

6. Import the new process model into the Workflow database as described in “Importing and exporting process models” on page 101.

Disabling a mapping type

To deactivate a program mapping type:

1. Delete all references to the mapping type from all OS/390 program properties, as shown in Figure 8 on page 98.
2. Import the changes, as described in “Importing and exporting process models” on page 101.
3. Delete the mapping type from the PES directory.
4. Import the new PES directory definitions as described in “Importing the PES directory” on page 107.

Administering invocation types

If you have defined an invocation type in the PES directory, and in the process model, you can then enable the invocation type.

Enabling an invocation type

In order to make a new invocation type and its corresponding exit known to MQSeries Workflow for OS/390 the following steps are necessary:

1. The invocation type must be defined in the PES directory as described in “Adding a user-defined invocation type” on page 107.
2. The invocation exit DLL for MQSeries Workflow for OS/390 must be copied into *InstHLQ*.SMFCLOAD, or the data set containing that DLL must be concatenated to the DD statement FMCSVLIB in the JCL procedure *CustHLQ*.SFMCPROC (FMCHPSRV).
3. The service and its connection parameters must be defined in the PES directory as described in “Adding a new service definition and the related user resolution information” on page 106.

Note: If there is more than one asynchronous invocation exit recognizing the same kind of reply messages, it is unpredictable which of the exits will handle a reply message.

Disabling an invocation type

To disable an invocation type you should do the following:

1. Delete all references to the invocation type from all OS/390 program properties, as shown in Figure 8 on page 98.
2. Import the changes, as described in “Importing and exporting process models” on page 101.
3. Delete the invocation type from the PES directory.
4. Import the new PES directory definitions as described in “Importing the PES directory” on page 107.

Program execution security

The program execution server accepts requests for program invocations from MQSeries Workflow users on different operating system platforms. The programs may be defined with additional security checking.

The program to be invoked can either be run using the *ServerUserID* or the user ID of the request starter. This is determined by the program property **Execution user** that is shown in Figure 7 on page 97.

1. **Execution user=Agent** causes the program to be run using the *ServerUserID*.
2. **Execution user=Starter** causes the program to be run using the user ID request starter. In this case **Local user=Yes** is required; this is set on the Figure 8 on page 98. The MQSeries WorkflowID of the request starter must be resolved to a local execution user ID under which the program will be run.

Note: If **Local user** is set to **no**, runtime error FMC32203 (Local user ID is required to execute program) will be generated .

The program property **Security checking=Yes/No** determines whether a security check is to be performed for requests before they are executed. You must set this property in Figure 8 on page 98.

The following combinations of settings are meaningful:

Table 66. Meaningful security setting combinations in Buildtime

Buildtime settings		How the PES handles an invocation request
PES property: User support	program property: Execution user	
Program	Starter	<p>In this case Local user=Yes is mandatory, which means that the PES uses the starter's MQSeries Workflow user ID in the PES directory to obtain the execution user ID that the program should be run using.</p> <p>If Security checking=Yes then a security check will be performed on the execution user ID.</p> <p>If the security check is passed successfully (or if Security checking=No) then the program will be invoked using the request starter's execution user ID.</p>
Program	Agent	<p>If Security checking=Yes then a security check will be performed on the <i>ServerUserID</i>.</p> <p>If the security check is passed successfully (or if Security checking=No) then the program will be invoked using the <i>ServerUserID</i>.</p> <p>The setting of Local user has no effect.</p>

Information in the PES directory that is relevant to security

The program execution server directory is where user ID mappings are defined.

If you have set the program properties **Execution user=Starter** and **Local user=Yes** as described in point 2 on page 115 above, then the MQSeries Workflow ID of the user who caused the invocation request must be mapped to an execution user ID under which the program will be run. This mapping is defined in the service subsection of the PES directory, as described in “Adding a new service definition and the related user resolution information” on page 106.

Program security

If a program is defined with **Security checking=Yes**, then it must have a security profile defined for the executable. The profile name is *MQWFSsystemPrefix.service.invocationtype.executable*, for example, *FMC.IMSCPIC.CPIC.FMCH3IMT*. Where *service*, *invocationtype*, and *executable* are the values that are defined in the program properties screen that is shown in Figure 8 on page 98.

All user IDs that are to be able to invoke the service must be given read access to the program’s profile. A program defined with **Local user=No** will be executed under the *ServerUserID*. In this case the *ServerUserID* must be given read access to the program’s profile. If **Local user=Yes** then the program will be executed under the execution user ID defined in the PES directory.

Chapter 11. Administering Servlets on the WebSphere Application Server

You can add a Java Servlet to the WebSphere Application Server for OS/390 that will call the Java-API of MQSeries Workflow. This allows users to invoke MQSeries Workflow actions from their Internet Web browser. It is required that you have performed "Customize Java-API support" on page 34.

WebSphere Application Server includes the Application Server Manager, a graphical interface that makes it easy to perform servlet management tasks, for example, setting options for loading local and remote servlets, setting initialization parameters, specifying servlet aliases, creating servlet chains and filters, monitoring resources used by the Application Server, and monitoring loaded servlets and active servlet sessions.

For administering Servlets for MQSeries Workflow the following three steps need to be done:

1. "Placing servlet class files on the Application Server".
2. "Placing the HTML files on the Application Server" on page 118.
3. "Running a sample servlet, to log on MQSeries Workflow" on page 118.

Placing servlet class files on the Application Server

To make a compiled servlet or servlet-package available via the WebSphere Application Server, you only need to copy your compiled servlet class files to a servlet directory under the Application Server's root directory that is declared in the classpath.

By default, the Application Server looks for servlet class files in the directory `applicationserver_root/servlets` where *applicationserver_root* is the Application Server's root directory. It is also possible to specify a different servlet directory where your compiled servlet class files will be stored.

If necessary, create a new servlet sub-directory

You can create a new servlet directory and add it to WebSphere's classpath by performing the following:

1. Create the sub-directory under `applicationserver_root/servlets`
2. Add the directory to the WebSphere classpath variable:
 - a. Open the WebSphere Application Server Manager at `http://your.server.name:9090/`.
 - b. Log in.
 - c. In the list of **Services**, double-click the instance of the Application Server you want to manage.
 - d. On the next page, select the **Setup** button.
 - e. Click on **Basic**.
 - f. Add the new directory to the **classpath**
 - g. Save it.

Monitoring your servlet, or setting servlet initialization parameters

If you want to monitor your servlet, or set servlet initialization parameters, use the Application Server Manager as follows:

1. Open the WebSphere Application Server Manager at `http://your.server.name:9090/`.
2. Log in.
3. In the list of **Services**, double-click the instance of the Application Server you want to manage.
4. On the next page, select the **Servlets** button.
5. If the servlet is not already listed in the Servlets tree-view, select the **Add** option.

Placing the HTML files on the Application Server

To place the HTML files on the Application Server:

1. Copy the HTML files for the servlet to a sub-directory of the Web server's HTML document root directory, for example, in `server_root/HTML_directory/morefiles`.
2. Check the pass rules in the file `/etc/httpd.conf`.

Then, for example, if you copied the file `myhtml.html` to `server_root/HTML_directory/morefiles`, the following URL will access the file:
`http://your.server.name/morefiles/myhtml.html`

Running a sample servlet, to log on MQSeries Workflow

You can verify that the servlets work by performing the following test:

1. Start the MQSeries Workflow System.
2. Add the following DD statements for the user and machine profiles to your WebServer's started task:

```
//FMCHUPR DD  DISP=SHR,DSN=CustHLQ.SystemGroup.System.SFMCDATA(FMCHUPR)
//FMCHEMPR DD  DISP=SHR,DSN=CustHLQ.SystemGroup.System.SFMCDATA(FMCHEMPR)
```

using your values for *CustHLQ*, *SystemGroup*, and *System* from "Installation scope identifiers" on page 9, "System group scope identifiers" on page 10, and "System scope identifiers" on page 11 respectively.

If a RACF profile exist for these data sets, add the user ID of your WebServer, for example, *WEBSRV*, to this profile. This user ID needs read access to both profiles.

3. Add the following environment setting to `/etc/httpd.envvars`

```
_ICONV_UCS2_PREFIX=SYS1
LC_ALL=En_US.IBM-1047
FMC_SIMPLE_TRACE_ONLY=YES
FMC_CURRENT_CONFIG=<MQWFConfigurationKey>
FMC_DEFAULT_CONFIGURATION=<MQWFConfigurationKey>
FMC_ELAPSED_TIME=YES
FMC_IENV=1
```

4. If you have not already performed the servlet option in step 1.2 of "Customize Java-API support" on page 34, then you should perform steps 1 and 2 again.

This copies the `HelloServlet.html`, `HelloServlet.java`, and `HelloServlet.class` files to the required Java directories.

5. Edit `HelloServlet.html` in the `<HFSHTML>` directory to match your installation – especially the URL.
6. Add the MQSeries Workflow Java agent `fmcojagt.jar` to the WebServer's classpath, add the directory where the external links to `fmcojprf` and `fmcojloc` are defined to WebServers `libpath`:
 - a. Open the WebSphere Application Server Manager at `http://your.server.name:9090/`.
 - b. Log in.
 - c. In the list of **Services**, double-click the instance of the Application Server you want to manage.
 - d. On the next page, select the **Setup** button.
 - e. Click on **Basic**.
 - f. Add `fmcojagt.jar` to the **classpath**
 - g. Save it.

Remember that after changing the WebServer's `libpath` and `classpath` it is necessary to restart the WebServer.

7. Start the WebServer.
8. Call the `HelloServlet.html`. If `HelloServlet` is placed to sub-directory `workflow` and you use port 8000, call it in the following way:
`http://your.server.name:8000/workflow/HelloServlet.html`
9. To log on, enter values for system group, system, userid, and password on the panel.

Chapter 12. Performance tuning

You can tune the performance of your MQSeries Workflow for OS/390 system in the following ways:

- “Changing the number of running server instances”
- “Changing the number of server instances per address space”

Note: Some other performance issues are covered in “User response times are unacceptably long” on page 127.

Changing the number of running server instances

The number of instances of each server type that are started when you start the system is specified in the process model. By default, the execution server is started with two instances, and the PES is started with five instances.

You can start additional servers using the administration console, as described in “Starting servers” on page 86. You should avoid having too many as this may cause the servers to terminate abnormally.

Changing the number of server instances per address space

The maximum number of instances for each server type that can be started in one address space is defined in the machine profile, see “Appendix H. Machine profiles” on page 193.

This only applies to the multiple instance servers when they are not managed by WLM. To change the number of instances of WLM managed servers, you must change the value of the start parameter SRVN0, shown in Figure 23 on page 151.

For the non-WLM managed, multiple server types:

- The number of **Execution server** instances started per address space is set with the variable ExeSvrsPerAS. The initial value is two.
- The number of **Program execution server** instances started per address space is set with the variable PESvrsPerAS. The initial value is five.

This means that starting six program execution servers will start two address spaces. The optimum number of server instances that can run in one address space depends on your hardware and configuration.

By default, the maximum number of servers that can be started in one address space is limited to eight. If you set the number of server instances per address space to a number greater than eight, a dump will be written if you attempt to start more than eight instances. If you want more than eight execution or program execution server instances per address space, you must edit the Workflow server start job as described in step 4 of “Workflow server customization” on page 32.

Caching the PES directory

By default, program execution server instances do not cache the contents of the PES directory, but you may wish to enable PES caching to improve performance. When caching is enabled, changes made to the PES directory may require a PES restart to force a refresh of the cache. For more information about this option, see “Caching the PES directory at runtime” on page 108.

Chapter 13. Problem determination

This chapter describes how you can solve various problem situations involving MQSeries Workflow for OS/390:

1. If you have problems with servers or undelivered messages (on hold queues), see "Server problems".
2. If you have problems with resources or performance, see:
 - a. "Resource and performance problems" on page 127
 - b. "Chapter 12. Performance tuning" on page 121
3. If you have problems with WLM, see "Chapter 16. WLM problem determination" on page 157.
4. If none of the previous solutions apply, you may decide to use one of the following:
 - a. "The MQSeries Workflow for OS/390 system trace facility" on page 129
 - b. "Tracing in CICS" on page 131
5. If you got an SVC dump, see "What do I do if I get an SVC dump?" on page 131.

Server problems

Message catalog not available

If the following message is displayed on the console:

```
'Couldn't get message with <ID> from message catalog'
```

This can be caused if the message catalog is not available, deleted, or corrupted. It is also possible that the message with the specified ID is empty or the message doesn't exist at all. Verify that the language code xxx specified in the machine profile is correct (see "Appendix H. Machine profiles" on page 193), and that FMCHMxxx and MMSLSTxx were created correctly during "Create the MMS message catalogs" on page 20.

Servers cannot be started

Problems starting the administration server and program execution server are described in "The administration server cannot be started" on page 124, and "The program execution server cannot be started" on page 124.

Server terminates immediately

If a server, other than the administration server, terminates immediately after being started, this can be caused if MQSeries cannot connect any more servers to a queue. MQSeries limits the maximum number of servers that can be running concurrently, and be connected to the same queue manager. This number depends on the maximum number of concurrent DB2 connections defined in ZPARM, and the maximum number of channels XPARM. Increase these values if necessary.

All but one server instance terminates immediately after starting

This happens if you try to start more than one instance of a single instance server type.

No execution server instances start

Is it managed by WLM?: If it is managed by WLM, this is not a problem, WLM will start instances as soon as some work arrives. You can find out whether it is being managed by WLM by issuing the following command:

```
D WLM, APPLENV=EXEApplicationEnvironmentName
```

where *EXEApplicationEnvironmentName* is the name of the execution server's application environment.

The administration server cannot be started

Is the queue manager started?

The administration server requires that the queue manager is running. If the queue manager terminates, the administration server will terminate as well.

Is an administration server already running?

It is not possible to start an administration server on a Workflow system where there is already one running. Issue the display command to check if an administration server is already running.

```
MODIFY AdminServerID,DISPLAY ADM RUNINSTANCE
```

Are its queues inhibited?

The administration server requires three MQSeries alias queues to be operational otherwise it cannot be started. These queues are:

- Boot request queueBOOT.REQUEST
- Administration client queueADC
- Administration input queueADM

The administration server will terminate if these queues are in the state PUT_INHIBITED or GET_INHIBITED. Check the status of these queues, and enable them if necessary. If the simple trace is activated, these improper queue states will be recorded in the trace. These will show up as MQSeries reason codes MQRC_PUT_INHIBITED or MQRC_GET_INHIBITED.

The administration server does not respond to console commands

This may happen if you issue commands before the server indicates that it is ready. Wait for this indication, and then try again.

The program execution server cannot be started

Is the administration server running?

If the administration server terminates, normally or abnormally, while PES instances are booting, the PES address spaces will normally terminate. If the administration server is started within a short time, the PES instances will continue to boot.

Are its queues inhibited?

The PES requires five MQSeries alias queues to be operational, otherwise it cannot be started. These queues are:

- Boot request queueBOOT.REQUEST
- Boot reply queueBOOT.REPLY
- PES input queuePES.PESERVER

- Working queuesPES.PESERVER.COR andPES.PESERVER.RPL

The PES will terminate if these queues are in the state PUT_INHIBITED or GET_INHIBITED. Check the status of these queues, and enable them if necessary. If the simple trace is activated, these improper queue states will be recorded in the trace. These will show up as MQSeries reason codes MQRC_PUT_INHIBITED or MQRC_GET_INHIBITED.

Server instance terminates unexpectedly without reporting an error

Unexpected server termination can be caused by an MQSeries or DB2 problem. Turn tracing on, as described in “The MQSeries Workflow for OS/390 system trace facility” on page 129, and reproduce the problem. Search the trace for the last line that contains the keyword THROW. The reason code there will identify the cause of the termination.

One or more program execution server instances terminate, request goes in state error (FMC_ERROR_RETRY_LIMIT_REACHED).

During asynchronous request processing the PES correlation queue, . . .PES.PESERVER.COR, or the PES reply queue, . . . PES.PESERVER.RPL, could not be accessed. Try checking the state of both queues and then restart the request.

Program execution request stays in state running

The request could not be processed for one of the following reasons:

- While processing an asynchronous reply the PES did not find correlation data belonging to this reply. This could be caused by one of the following:
 - An inconsistent state of the correlation queue, for example, if the queue has eventually been emptied.
 - An invocation type that returns more than one reply for the same request.
 - A message that is unknown to the program execution server.

An internal error notification has been sent to the administration server and written to its error log. The notification contains the MQSeries message descriptor to identify the source of the message.

- While processing an asynchronous reply the specified invocation exit was not able to uniquely identify the reply. This could be caused by a programming error in the exits’ Recognize() interface. An internal error notification has been sent to the administration server and written to its error log. Try to correct the Recognize() interface and then restart the request.

A dump is written before all server instances are started

Have you exceeded the limit set in the server start job?

By default, the maximum number of servers that can be started in one address space is limited to eight. If you set the number of server instances per address space to a number greater than eight, a dump will be written if you attempt to start more than eight instances. If you want more than eight execution or program execution server instances per address space, you must edit the Workflow server start job as described in step 4 of “Workflow server customization” on page 32.

Is the number of server instances per address space too high?

If you get a dump before the specified number of server instances have started the server instances required more memory than the address space offers. You should reduce the value for the number of server instances started per address space. This is described in “Changing the number of server instances per address space” on page 121.

Cannot stop servers

For servers that are not managed by WLM, the stop server command works by disabling the server input queue for a given length of time, and then re-enabling the queue as soon as all the server instances have stopped, or after the queue disable period. When a server completes its current transaction, it will check its input queue. If the input queue is disabled, the server will shut itself down.

For servers that are managed by WLM, see “Chapter 16. WLM problem determination” on page 157.

Did you wait long enough?

1. Issue the stop server command.
2. Wait at least 30 seconds, this is the initial queue disable period.
3. Issue the display command to check how many server instances are running.
4. If there are still some instances running, you can try repeating from step 1 again.
5. If this does not work, use CANCEL. Any transactions being performed by server instances in the cancelled address space will be rolled back.

Do your transactions take longer than 30 seconds?

If a server’s current transaction takes longer than the queue disable time (initially 30 seconds), it is possible that the server never finds the queue disabled, and so does not shut down. Simply repeating the stop server command may work.

If this problem persists, you can try increasing the value for the **WaitBetweenQInhibitAndAllowed** setting in the machine profile *CustHLQ.SFMCDATA(FMCHEMPR)*, see “Appendix H. Machine profiles” on page 193 for more information. After changing the machine profile you must perform “Restarting the administration server” on page 88.

PES cannot be stopped

The following program execution server characteristics may affect attempts to stop it:

- The PES cannot be stopped within five minutes of it being started.
- While handling a synchronous invocation, the PES is blocked for the duration of the transaction. Check your service system.

Changes made to the machine profile are not activated

Have you restarted the administration server?

Changes to the machine profile will only become active after the administration server is restarted, as described in “Restarting the administration server” on page 88. All existing server instances will continue to use the old machine profile settings. After the restart all new server instances will use the new machine profile settings.

If you do not want any servers to continue using the old machine profile, you must also restart the whole MQSeries Workflow for OS/390 system.

Changes made to the PES directory are not activated

Is PES directory caching enabled?

If PES directory caching is enabled in the machine profile, changes to the PES directory will only take effect after the cache is refreshed. This is done by stopping and restarting all program execution server instances, as described in “Restarting the program execution server” on page 88.

Changes made to the program mapping definition are not activated

Have you restarted the program execution server?

Changes to a program mapping in the program mapping database may require a PES restart, as described in “Restarting the program execution server” on page 88.

Hold queue problems (undelivered messages)

While administering undelivered messages (as described in “Hold queue commands” on page 89), the following problems may occur:

DELETE or REPLAY affected fewer messages than expected

This may be caused by one of the following:

1. A message cannot be removed from the hold queue if it has not been displayed first.
2. If you issued the DELETE command while a REPLAY command was being processed, messages other than those you wanted to delete may have been removed. This is because the execution of the DELETE command can interfere with the execution of the REPLAY command. Avoid this situation by making sure that the REPLAY command has finished before issuing a DELETE command.

REPLAY is not executed, the hold queue contains less than the expected number of messages

This situation can arise if someone has manipulated the hold queue using commands other than those that are supported by the administration server, for example, using MQSeries queue handling commands.

Resource and performance problems

User response times are unacceptably long

Performance problems may be caused by one or more of the following:

Is tracing turned on?

Operation is significantly slower when tracing is active. Turning tracing off is described in “Turning tracing off” on page 131.

Are enough server instances running?

It is possible that there are not enough server instances to cope with the workload.

For the PES, this can happen because a PES instance is blocked while it processes a request that is synchronous. So having a high request rate, or having requests that cannot be completed quickly may require that you start more server instances. You can start additional execution server instances, see “Starting servers” on page 86.

Are too many server instances running?

In this case restart fewer servers instances, as described in “Restarting servers” on page 88.

Does the workload exceed your system’s capacity?

If you have eliminated the above possibilities, it may be that the workload exceeds your system’s capacity.

Invalid password**Are you using an old version of the runtime client?**

This can happen when an old runtime client tries to connect to a newer administration server. You should install the MQSeries Workflow Version 3.2.1 runtime client.

Running out of spool space**Is tracing turned on?**

This problem can be caused by the trace facility. Check which servers have been started with trace turned on. Trace entries are written even when the servers are idle. Reduce the trace level as described in “Turning tracing on” on page 129 (or turn it off) and restart the server type that was being traced.

The MQSeries Workflow for OS/390 system trace facility

Trace is used to diagnose reproducible problems by recording statements and instructions that will be executed within the MQSeries Workflow for OS/390 system in the sequence in which they occur. The trace facility records system events in data sets.

The trace facility provides two types of tracing:

1. **Simple trace** writes all trace entries directly to the data set, allocated as SYSOUT in the server JCL procedure SYS1.PROCLIB(*UniqueSystemKey*).
2. **Extended trace** is a powerful tool exploiting the OS/390 Component Trace system services. It works with wrap around buffers in storage, so that when several server types are running, and the extended trace has been activated, their trace entries will be written to each server's storage. When a buffer is full, the following trace information will be written to another buffer and the full buffer is passed to an asynchronous Component Trace external writer in order to be written to a data set.

Note: Extended trace is only available on servers, it does not provide information on MQSeries Workflow for OS/390 tools.

To produce a trace of the MQSeries Workflow for OS/390 system, you must perform the following steps:

1. "Turning tracing on"
2. "Restarting the component and reproducing the problem" on page 130
3. "Viewing the trace" on page 130
4. "Turning tracing off" on page 131

Turning tracing on

To turn tracing on, select the trace level and type of trace you want:

1. Edit the machine profile file *CustHLQ.SFMCDATA(FMCHEMPR)*:
 - a. To get the most important trace entries, set the value:
`MQWorkflowMachine.MQWFSsystemPrefix.FMC_TRACE_CRITERIA:3,FFFF,FFFFFFFF`
 - b. To get a good overview, set the value:
`MQWorkflowMachine.MQWFSsystemPrefix.FMC_TRACE_CRITERIA:33,FFFF,FFFFFFFF`
 - c. To get maximum information, set the value:
`MQWorkflowMachine.MQWFSsystemPrefix.FMC_TRACE_CRITERIA:99,FFFF,FFFFFFFF`

Note: This trace level significantly reduces the performance of the system, and requires a large quantity of disk space.

2. Edit the environment variable file *CustHLQ.SFMCDATA(FMCHEENV)*
 - a. If you want extended trace, set
`FMC_SIMPLE_TRACE_ONLY=NO`

Note: This is the default, and is recommended because it does not create any synchronous input/output overhead.

- b. If you want simple trace, set
`FMC_SIMPLE_TRACE_ONLY=YES`

Now any servers or tools that you start will produce the trace information you specified.

Restarting the component and reproducing the problem

After starting the trace:

1. Run the component that is causing the problem.
2. Reproduce the problem.
3. If extended trace has been activated:
 - a. Issue the following command on the OS/390 system console:

```
TRACE CT,,COMP=CTComponent,PARM=CTIFMmm
```

where *CTComponent* and *mm* are your values for the CTRACE component name, and the two character CTRACE suffix *CTStartSuffix* — these values were defined in Table 3 on page 11.

- b. Run the JCL *CustHLQ.SFMCCNTL(FMCHJTRC)*. This converts the format of the extended trace data sets into the same format as the simple trace.

The extended trace data set format converter FMCHJTRC can return the following return codes:

Table 67. Extended trace format converter return codes

Value	Description	Explanation
0	Successful completion	The formatted results of the extended trace are available in the job output of FMCHJTRC.
4	Warning	No trace data was found. The input data set may be empty.
8	Error	Invalid trace buffer records were detected. Due to the use of the CTRACE external writer WRAP option, it is possible that the oldest records in the data set have been partially overwritten, and are unusable.
12	Severe error	Invalid trace buffer header records were detected. Either the trace data set has been corrupted, or the input data set was not created by a CTRACE external writer.

Viewing the trace

After reproducing the error, you can view the results of the trace in the following way:

1. The results of the simple trace is in the SYSOUT data sets of the started job.
 - a. The DD statement for tools is FMCTRC00.
 - b. The DD statements for servers are FMCTRCxx, ...01, ...02,
2. The formatted results of the extended trace will be available in the job output of FMCHJTRC. The fields in each line are: Date, time, filename, line number, current trace settings (level, category, component), process name, address space ID, server ID, function name, and description. The following is an example line from a log file:

```
1998-06-09, 10:27:47.94, FMC.DUMMY.CPP#(DUMMY1)( 421), (33,SC,Kr),  
Process Name(131-01), TestClass::Find(const TestString&), ifstream.close( )
```


Turning tracing off

To deactivate the trace:

- Edit data set *CustHLQ.SFMCDATA(FMCHEMPR)* and specify
`MQWorkflowMachine.MQWFSsystemPrefix.FMC_TRACE_CRITERIA:0,0000,00000000`

If you have been tracing a server, it will continue writing trace data until it is stopped or restarted, as described in “Stopping servers” on page 87 and “Restarting servers” on page 88.

If necessary, send the trace file to the appropriate IBM support personnel. Tracing should be turned off when it is no longer required because MQSeries Workflow for OS/390 operation can be significantly slower when tracing is active. Delete the trace file when the problem is solved and you do not need it any longer.

Tracing in CICS

All MQSeries Workflow for OS/390 components running in CICS use the CICS trace facilities to generate trace entries. Trace parameters are read from the machine and user profile and from the environment VSAM files that were generated during customization. The settings in the environment file overrule the machine profile settings. The settings affect all MQSeries Workflow for OS/390 programs running in the corresponding CICS region.

To print the contents of the CICS auxiliary trace data set, submit the JCL *CustHLQ.SFMCCNTL(FMCHJCTC)*.

The type of information that is provided, and the parameters for the MQSeries Workflow for OS/390 trace are described in “The MQSeries Workflow for OS/390 system trace facility” on page 129. For more information about the CICS trace facilities see *CICS/ESA: Problem Determination Guide* and *CICS Transaction Server for OS/390: CICS Problem Determination*.

What do I do if I get an SVC dump?

Create a problem summary

When a dump occurs various information is available for analysis. You can run the job *FMCHJDMP*, which calls an exec *FMCHKDMP* under *IPCS*. This analyses the SVC dump from a Workflow server. The output of this job presents various information about the system, the server instance that caused the problem, and also an analysis of the language environment. This information is presented in the same format as a *CEEDUMP*.

This allows you to create and submit a problem summary which is considerably reduced in size compared to the dump data set. This problem summary will be sufficient for the analysis of most problems.

Part 3. Using OS/390 Workload Manager with Workflow

Chapter 14. Introduction to WLM	135
What is OS/390 Workload Manager?	135
Overview of WLM	135
WLM queuing model.	135
Service definition	137
MQSeries Workflow and OS/390 Workload Management	138
Workflow administration server	139
Starting the system in WLM mode	140
Starting a WLM managed server	140
Stopping the system	140
OS/390 Workload Manager application environments	141
MQSeries for OS/390 workload management	143
Classification	144
Chapter 15. Setting up WLM for MQSeries	
Workflow for OS/390	145
Creating a WLM service definition	145
Service definition	145
Service policy	146
Workload.	147
Service class.	147
Classification rule	149
Application environment	150
Installing and activating a WLM service policy in a Parallel Sysplex environment	152
WLM administration	153
Switching servers between WLM and non-WLM mode by importing an FDL file	154
Switching servers to WLM mode.	154
Switching servers to non-WLM mode	154
Starting WLM-managed servers when WLM is in manual mode	155
Chapter 16. WLM problem determination	157
WLM setup problems	157
Unexpected runtime behavior of MQSeries	
Workflow with WLM.	157

Chapter 14. Introduction to WLM

Using OS/390 Workload Manager (WLM) with MQSeries Workflow for OS/390 is optional. You can use WLM to achieve your workflow service performance goals. It does this by automatically adjusting address space resources and the number of running server instances for the execution server and/or the program execution server. If you do not use WLM, you must start and stop server instances manually, using the administration commands described in “Chapter 8. Administration console tasks” on page 83.

This chapter introduces workload management concepts with emphasis on its use with MQSeries Workflow. Additional information about workload management can be found in the following documentation:

- *OS/390 MVS planning: Workload Management*
- *OS/390 MVS Programming: Workload Management Services*

What is OS/390 Workload Manager?

This chapter gives a high level description of OS/390 Workload Manager (WLM) and its influence on the mechanics and behavior of MQSeries Workflow for OS/390. The chapters following discuss these issues in more detail.

Overview of WLM

The OS/390 Workload Manager (WLM) manages the allocation of MVS resources to best meet your performance goals. Your service level administrator defines goals for each distinct work request in a single sysplex wide service definition. The service definition covers all types of MVS work. It is defined by a service level administrator using the ISPF panels described in *OS/390 MVS planning: Workload Management*. It allows you to define a performance goal and the business importance of achieving that goal for each MQSeries message. See “Service definition” on page 137 for more details. During workload execution, WLM periodically samples the work to see how well the goals in the service definition are being met. It then adjusts the MVS resources allocated to each address space or each TCB / SRB to best achieve all performance goals. It basically takes resources from overachievers and gives them to underachievers. WLM may also start or stop an address space if that helps achieve the performance goals.

All aspects of the OS/390 Workload Management facility, can be maintained using the standard ISPF WLM application. How you setup and customize WLM is described in “Chapter 15. Setting up WLM for MQSeries Workflow for OS/390” on page 145.

WLM queuing model

The queuing manager services of the OS/390 Workload manager are intended for queuing managers to manage execution server address spaces and the work requests they process to meet service class performance goals. In general, a queuing manager is a subsystem that queues work requests to workload management for execution in server address spaces. In the context of MQSeries Workflow, a queuing manager is the MQSeries Queue Manager.

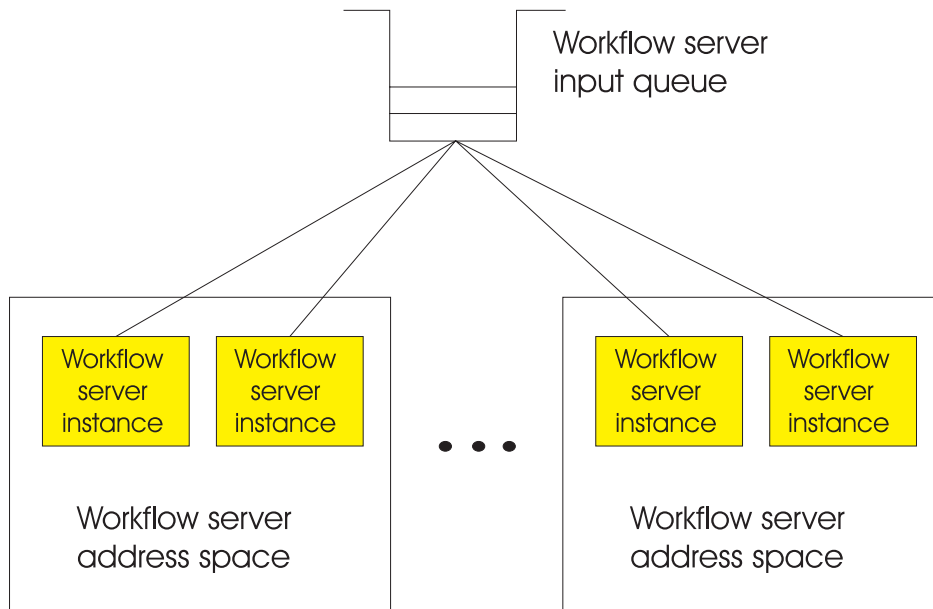


Figure 12. Server address spaces

Workload management dynamically starts and maintains MQSeries Workflow server address spaces as required to meet the performance goals based on response time evaluation and the workload in the queues used for the communication between MQSeries Workflow components. Consequently, there is no longer the need for manual intervention by the workflow administrator – starting or stopping of server instances – in case of changes in the system workload. Workload management spreads the work across multiple address spaces, providing workload isolation and greater scalability based on workload demands.

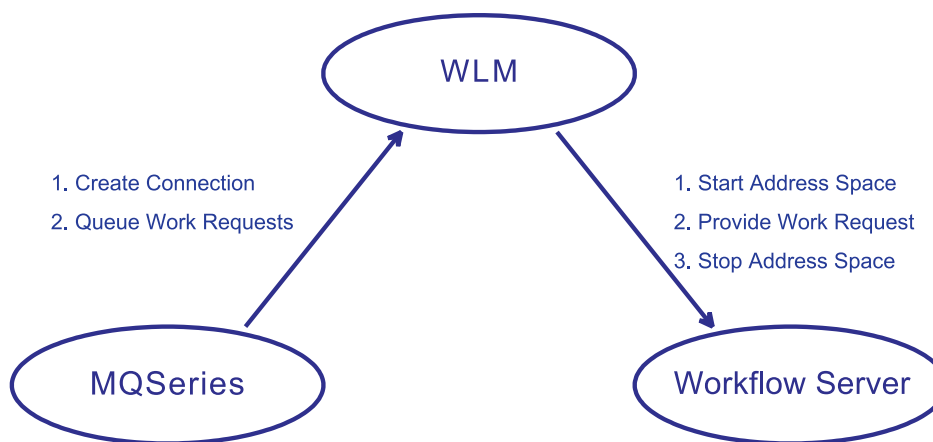


Figure 13. Interactions between MQSeries and WLM to manage execution server address spaces

OS/390 workload management and servers based on the workload management facility via a queuing manager interact as follows: when a queuing manager starts, it connects and queues work to workload management. The server address spaces are created on demand by workload management as defined in the service definition (see “Service definition” on page 137). In turn, when a server initializes, it connects to workload management, which allows it to select work from the

queues. The server address space will normally terminate only on request from workload management and after the current transaction is finished. The workflow server itself can also be the source of work requests that are intended for the same or another server type, these requests also go via the MQSeries queue manager.

Service definition

The service definition can be maintained using the standard ISPF WLM application as described in “Creating a WLM service definition” on page 145.

WLM collects data on how well the goals are being met in the same terms as those in the service definition. This information is available to reporting programs to help you refine your service definition. The elements of a service definition are:

Service policy

A named modification of your base service definition. It typically contains values for some of the goals associated with some of your service classes. You can have different service policies for different times of the day, week, or month. You can also activate a service policy using an operator command that deactivates any previously active service policy.

Service class

Each piece of WLM managed work is associated with a named service class through the classification process. You associate a performance goal and a business importance of achieving that goal with each service class.

Performance goals are normally defined as a response time goal for the transactional type work managed by MQSeries Workflow for OS/390. A response time goal specifies the desired time in milliseconds between acceptance of the work by a system, and its delivery of the result to the client. In an MQSeries environment this response time goal includes time spent waiting on MQSeries queues.

The **business importance** specifies five levels of importance, and is used when there is insufficient system capacity to enable everyone to achieve their performance goals.

Velocity goals are available for batch type work, which specify how fast the batch work should run when not held up by input/output. MQSeries will recognize this situation in the work qualifiers it passes to WLM for each message at classify time.

Discretionary goals are available for work to be run when the system has some unused resource.

You can further control the processing of longer running transactions by specifying up to 100 periods each with a different performance goal and a different importance. Each period defines an amount of consumed resources. Note that WLM queuing services only look at the actual initial performance statistics when deciding whether to start or stop a server address space. You can also associate a service class with a named resource group to limit the resource available to that service class, or to guarantee a minimum level of resource for that service class. This is particularly useful to ensure that discretionary work gets some service.

Classification rules

Classification is the way WLM associates an incoming work request with a named service class, and therefore with a performance goal and business importance. It also optionally associates an incoming work request with a

WLM reporting class for monitoring purposes. For more details see “Appendix J. WLM message classification” on page 199.

Application environment

You can define a number of named application environments to WLM, and associate each with the JCL procedure and start parameters to start an instance of a server address space for that environment. A later section will give details on the usage of application environments within MQSeries Workflow.

When you have classified a work request you can ask WLM to queue a reference to it in an internal WLM work list for a specified application environment. This WLM work list is split into one sub list per distinct service class, with the sub list entries in FIFO order. WLM will then start the optimum number of server address spaces for each service class to best meet performance goals. There will, of course, be at least one server address space for each distinct service class. Each server address space will thus be processing work with the same performance and business importance goals. WLM can then remove MVS resources from over achieving server address spaces and give them to under achieving server address spaces.

For each application environment you can optionally tell WLM to limit the number of server address spaces it will automatically start to one per MVS image or one per sysplex. It is strongly recommend that MQSeries Workflow users specify no limit. If you specify a maximum of one server per MVS image then the internal WLM work list will be FIFO ignoring service class.

MQSeries Workflow and OS/390 Workload Management

The following sections describe how the WLM queuing model is utilized in MQSeries Workflow. There are three relevant participants in the workflow system: MQSeries, MQSeries Workflow, and the OS/390 Workload Manager. When planning for performance goals, it is important to understand that the response time used by WLM is calculated as the sum of the time that the request spent on the MQSeries message queue, waiting to be forwarded to its intended receiver, and the time that the workflow server actually spent processing the request. This is illustrated in Figure 14 on page 139.

Response Time = Queue Time + Service Time.

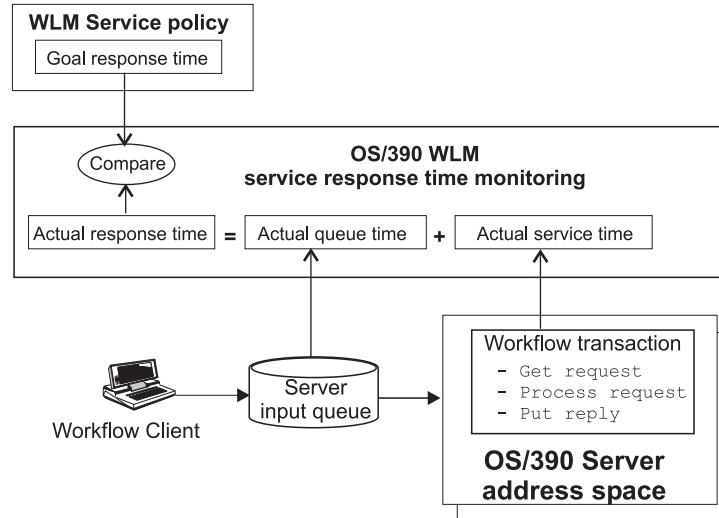


Figure 14. WLM service policy: response time goals

In contrast to the traditional way that applications use WLM, MQSeries Workflow uses asynchronous communication via MQSeries queues during the processing of process instances. WLM measures the queue time and service time for each request. Whenever a response time goal is not met, WLM may decide to adjust its management of system resources available to either the queue manager or the workflow servers. In addition, WLM may decide to adjust the number of workflow server instances. The reverse decisions may be made when a service class is over-achiever its goals, for example, when the workload is low.

The most important issue when setting up workload management is the determining of the performance goals. These goals do not depend just on the response time expectations of the workflow users, since the type of workload produced by the clients, and the available resources also have to be taken into account. There are two modes in which WLM can operate:

1. Goal mode
2. Compatibility mode

Compatibility mode is not supported by MQSeries Workflow.

Workflow administration server

The main difference between MQSeries Workflow with and without usage of the OS/390 Workload Management facility for a workflow administrator is the way that the number of server instances responsible for handling work requests is controlled. This parameter is crucial for the throughput and the overall performance of the workflow system.

It is important to note that the MQSeries Workflow administration server cannot effectively control this parameter on its own to maximize the system throughput. For example, by monitoring the queue depths: a deep queue in a system under high load indicates a need for more servers to handle the large amount of messages, which in turn would consume even more system resources and hence lower the throughput. On the other hand, MQSeries and MQSeries Workflow all cooperate with the OS/390 Workload Manager to provide the required services for system-wide or even sysplex-wide performance optimization.

Without OS/390 Workload Manager, the MQSeries Workflow administration server, supports commands for starting and stopping servers directly. With WLM, this responsibility can be transferred to the OS/390 Workload Manager. Then the MQSeries Workflow administration server delegates the administration of server instances to the OS/390 Workload Manager. MQSeries Workflow is then said to run in WLM mode, meaning that servers are started and stopped as indicated by the current overall workflow system performance. However, the workflow administrator can still use the same administration console commands which are then mapped internally to the appropriate WLM calls - only the behavior of these commands is different from the non-WLM mode.

Starting the system in WLM mode

The normal way to start an MQSeries Workflow system in WLM mode is to simply start the Workflow system with the standard administration console command, as described in “Starting the administration server and administration console” on page 83.

WLM can only manage the multiple instance server:

- Execution server
- Program execution server

Starting a WLM managed server

When starting a WLM-managed server, the administration server issues a RESUME for the respective application environment. The OS/390 system log will display a message for each resumed application environment. Any further attempts to start more server instances for WLM controlled server types will be rejected. Only the application environments are made available, and WLM decides when and how many server instances are started.

The application environments and their connections to the workflow system are specified in the WLM service definition (see “Service definition” on page 137 and “Creating a WLM service definition” on page 145). The OS/390 Workload Management facility will then start the necessary servers on demand based on this service definition.

Each server type under control of OS/390 Workload Management is associated with a WLM managed queue which is responsible to queue the work requests to the server instances of this type. WLM managed queues, in turn, are each associated with initiation queues which basically represent the state of the WLM managed queue: is the application environment of the WLM managed queue accepting work requests. Work requests are handled by WLM if and only if the initiation queue is open (MQOPEN for input count is greater than zero). The WLM managed queue is then said to be “trigger enabled”. If the queue is not trigger enabled, the work request is queued until its initiation queue is opened and the WLM managed queue transitions to the state “trigger enabled”. This means that during the startup of the server address space for an application environment, an MQOPEN for input is issued to the initiation queue, after this, WLM is able to process incoming work requests.

Stopping the system

To shut down the workflow system you issue the standard administration console commands. The stop commands are mapped to WLM calls for the server types (application environments) under WLM control. Hence, the request to terminate a server type under WLM control will actually result in a quiesce of the corresponding application environment and the server address spaces will terminate.

Since the administration server issues the WLM calls to control the various address spaces for WLM controlled server types and is also responsible for starting and stopping non-WLM controlled server types, the administration server is still the first server to be started and last to be stopped in a Workflow system. The administration console commands are described in “Chapter 8. Administration console tasks” on page 83.

OS/390 Workload Manager application environments

An application environment is defined as a group of application functions requested by client applications and executing in server address spaces. In the context of MQSeries Workflow, a server type is defined to the OS/390 Workload Manager as an application environment. Hence, the server types in MQSeries Workflow which may be under the control of WLM - the Execution Server and the PES - all define different application environments. The aforementioned clients are actually all MQSeries Workflow components putting work requests onto WLM managed queues.

Since in MQSeries Workflow each server type mentioned above listens to its own MQSeries queue, each application environment actually corresponds to a queue defined in the Workflow system. Note that this is no one-to-one correspondence, since there may be more queues than there are application environments.

All application environments have to be specified - and consequently also maintained - in three different places in a MQSeries Workflow system. The section troubleshooting discusses how to determine if application environment mismatches are the cause of unexpected system behavior.

1. The OS/390 WLM needs the information about all application environments in order to be able to start, control or terminate the server types associated with each application environment.
2. The proper MQSeries definitions must be in place in order to enable MQSeries to issue the appropriate WLM calls whenever requests are arriving on a queue.
3. MQSeries Workflow itself needs information about the application environments to be able to issue the correct OS/390 Workload Manager commands to start or terminate the server types under control of WLM. How to switch MQSeries Workflow to WLM mode is described in “Switching servers to WLM mode” on page 154. On the workflow side, the application environments have to be present in the MQSeries Workflow runtime database. Thus, information about the application environments have to be specified in the workflow system information given in the FDL (process model) containing the system definition either on the domain, system group, or system level.

You define your application environments, and make them available WLM using the ISPF WLM application, see “Chapter 15. Setting up WLM for MQSeries Workflow for OS/390” on page 145 for more details.

You manage the application environments using the administration console commands. The administration server, then issues the WLM commands `quiesce` and `resume`, which cause the application environment to transition between states. The possible application environment states are:

available

after initial startup or a `resume` command has completed.

quiescing

a `QUIESCE` command has been issued. When the command is completed the **quiesced** state is entered.

quiesced

a QUIESCE command has completed.

resuming

a RESUME command has been issued. When the command is completed the **available** state is entered.

refreshing

a REFRESH command has been issued. When the command is completed the **available** state is entered.

stopped

the maximum number of unexpected server terminations have been encountered. The default value is five.

stopping

the state during the transition to **stopped**.

The REFRESH command is in essence similar to a QUIESCE command issued to an application environment followed by a RESUME command. Refreshing an application environment is useful to restart the server instances, so that changes in the machine profile or environment variable file can take effect.

The states and their associated commands are listed here for completeness only, since they are almost always issued internally by the administration server. REFRESH is the only exception from this rule and should only be applied to activate profile changes. All other administration tasks are accomplished from the MQSeries Workflow administration console as described in “Chapter 8. Administration console tasks” on page 83.

Operators are strongly discouraged to issue any WLM commands related to the workflow system for normal administration purposes. Using WLM commands directly can result in inconsistent system behavior and the MQSeries Workflow console offers all necessary tools for the runtime management of the workflow system. However, the state of an application environment may be displayed with the corresponding DISPLAY WLM system command as long as no attempts of any modifications are made.

Work requests are accepted if and only if the application environment referenced in the work request is in the state available. Each application environment defined in the Workflow system is available after the initialization has completed. Work requests (messages in a MQSeries queue) issued to an unavailable application environment remain in the queue until the application environment is available again. Note that these messages may time-out while waiting for the application environment to become available. The state of the various application environments can be checked using standard OS/390 Workload Management console commands.

OS/390 Workload Management may start or stop server address spaces as required. This decision depends on previously gathered performance data (based on the response time behavior of earlier work requests), and the current state of the system. For example, the number of currently available server address spaces belonging to the same application environment as the work request.

It is important to note that WLM collects performance data across a sysplex. To do this, the WLM instances in each MVS image exchange performance data. WLM optimizes the overall performance of MQSeries workflow within the scope of a

sysplex. However, this may mean that a work request might not get enough resources in a particularly busy MVS image, if its associated service class meets its performance goals at the sysplex level.

MQSeries for OS/390 workload management

One technique for performance optimization is the prioritization of work requests. Moreover, WLM has a sysplex-wide view with regard of the fulfillment of the performance goals and uses these priorities to improve the overall performance. For this effect MQSeries Workflow, MQSeries and WLM in the queuing model interact closely in the handling of incoming messages.

The utilization of OS/390 Workload Management with the queuing model fundamentally changes the way messages put into a MQSeries queue are handled, this is illustrated in Figure 15. Based on the classification information passed to WLM, and the active classification rules, WLM assigns the messages to the servers active in the application environments. Conceptually, the messages are sorted into "virtual queues" according to the message priority given in the service class. The MQSeries Queue Manager delegates message priority management to the OS/390 Workload Manager.

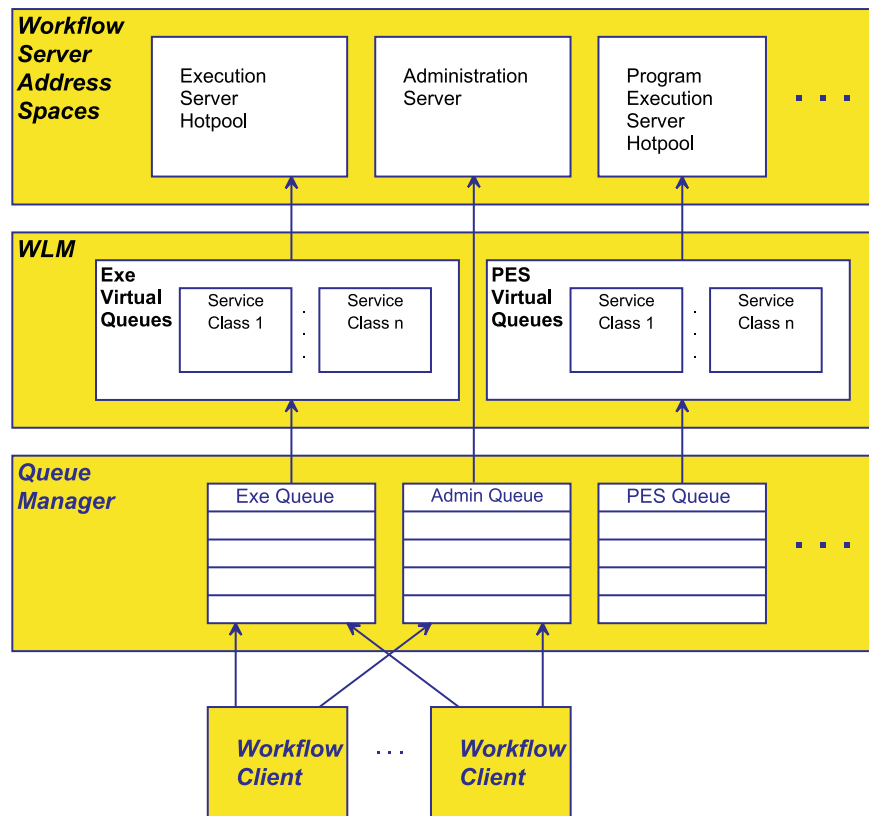


Figure 15. The WLM queuing model used by MQSeries Workflow

WLM decides which service class needs to be serviced in order to achieve the performance goals for this class and assigns a message from this service class to the next server ready to handle a new work request. The decision, which service class needs resources (which message is given to the requested server type), is not just based on the state of the service class with respect to its performance goals.

WLM also takes the states of the other service classes into account. Therefore, there is no guarantee that the queue manager handles messages in the order they are put into the queue.

When not using the OS/390 Workload Management facility, messages arriving in a MQSeries queue are retrieved according to a first-in-first-out (FIFO) strategy. In this case, no work request priorities are taken into account when servers get work requests from their queues. Note that MQSeries priorities are not used by MQSeries Workflow.

Classification

Work requests are classified using information provided by the originator of a message. This is either a client or a server that is communicating with another server. The information is specified in the **Work Information Header (WIH)** which is contained in the message with the message's other contents. Therefore each client or server that builds messages that will be put into a queue must also generate a WIH. Users writing client applications, however, do not have to concern themselves with this task, since the API handles the whole process of setting up the WIH.

Once the OS/390 Workload Manager receives a message via the queuing service, it interprets the WIH and can then assign the work request to a service class. From the service class, WLM obtains the performance goal, and can now manage the request towards this goal.

To give OS/390 Workload Management the ability to use the MQSeries WIH for the classification of messages, a new subsystem type MQ has been added to WLM. This subsystem type has to be used when defining the performance goals for MQSeries Workflow for OS/390.

For more information about message classification, see "Appendix J. WLM message classification" on page 199.

Chapter 15. Setting up WLM for MQSeries Workflow for OS/390

To enable WLM support in MQSeries Workflow for OS/390 you need to create definitions for MQSeries for OS/390, MQSeries Workflow for OS/390 and WLM itself. These definitions must be consistent with each other. In MQSeries Workflow for OS/390 you must edit the profiles and Buildtime settings. In MQSeries for OS/390 you will use the command language interfaces to enable queues as being managed by WLM. Finally in Workload Manager itself, you will make some definitions using the ISPF interface.

Creating a WLM service definition

Before you can set up your WLM definitions, you must identify the workloads, the service classes, the service class periods, and goals based on your performance objectives. Then you define classification rules. Together, this information provides the WLM service definition base.

The following procedure includes the minimum definitions required for Workload Management support for MQSeries Workflow for OS/390. You should use these definitions the first time you setup WLM to verify the functionality, without regard for performance tuning.

Note: These panels were made using OS/390 V2R6.0. If you are using a newer version of OS/390, the panels may look slightly different. The information that you enter on these panels is also provided in *CustHLQ.SFMCDATA(FMCHEWLM)*

Service definition

The service definition includes workloads, service classes, systems, resource groups, service policies and classification rules. Each service definition must be given a unique name.

```

File Utilities Notes Options Help
-----
Functionality LEVEL001          Definition Menu          WLM Appl LEVEL007
Command ==> _____

Definition data set . . . : none

Definition name . . . . . MQWFEDE (Required)
Description . . . . . Service Definition for MQWF

Select one of the
following options. . . . . ___ 1. Policies
                               2. Workloads
                               3. Resource Groups
                               4. Service Classes
                               5. Classification Groups
                               6. Classification Rules
                               7. Report Classes
                               8. Service Coefficients/Options
                               9. Application Environments
                              10. Scheduling Environments

```

Figure 16. WLM panel: Definition menu

Parameter	Required or Optional	Description	Examples
Definition Name	Required	Eight character identifier for the service definition.	MQWFEDE
Description	Optional	A description of the service definition (up to 32 characters.)	Service Definition for MQWF

Service policy

A service policy is a named collection of service class and resource group specification overrides. When a policy is put into effect, the overrides are merged with the service class and resource group specifications in the service definitions. A policy override is a way to change a goal or resource group capacity without having to redefine all of your service classes and resource groups. The initial setup described here requires only a single set of performance goals. Hence the policy you have to define contains no overrides. You must activate a service policy to take your definitions into effect, this is described later.

```

Service-Policy Notes Options Help
-----
                          Create a Service Policy
Command ==> _____

Enter or change the following information:

Service Policy Name . . . . . MQWFEPO (Required)
Description . . . . . Service Policy for MQWF

```

Figure 17. WLM panel: Create a service policy

Parameter	Required or Optional	Description	Examples
Service Policy Name	Required	Eight characters identifying the service policy. Every service policy name must be unique in a service definition.	MQWFSEPO
Description	Optional	A description of the service policy (up to 32 characters.)	Service Policy for MQWF

Workload

A workload is a named collection of work to be reported as a unit. You can arrange workloads by subsystem (MQ, CICS, IMS) or by major application (Production, Batch, Office). Logically, a workload is a collection of service classes. When you choose the Workload option for the first time, the application displays the *Create a Workload* panel shown in Figure 18.

```

Workload Notes Options Help
-----
                                Create a Workload
Command ==> _____

Enter or change the following information:

Workload Name . . . . . MQFWOLO (Required)
Description . . . . . Workload for MQWF

```

Figure 18. WLM panel: Create a workload

Parameter	Required or Optional	Description	Examples
Workload Name	Required	Eight characters identifying the workload. Every workload name must be unique for all defined workloads in a service definition	MQFWOLO
Description	Optional	A description of the workload (up to 32 characters.)	Workload for MQWF

Service class

A group of work with the same performance goals, resource requirements, or business importance. For workload management, you assign a service goal and optionally a resource group to a service class. You should define different service classes for each of the server types you want WLM to manage, plus a default service class. The two Workflow server types that WLM can manage are:

- The execution server — this processes runtime process instances.
- The program execution server — this handles requests for CICS and IMS program invocations.

```

Service-Class  Notes  Options  Help
-----
                                Create a Service Class                                Row 1 to 1 of 1
Command ==> _____

Service Class Name . . . . . DEF_SC    (Required)
Description . . . . . Default Service Class for MQWF
Workload Name . . . . . MQFWOLO    (name or ?)
Base Resource Group . . . . .          (name or ?)

Specify BASE GOAL information.  Action Codes: I=Insert new period,
E=Edit period, D=Delete period.

    ---Period---  -----Goal-----
Action # Duration Imp. Description
    1              Discretionary
***** Bottom of data *****

```

Figure 19. WLM panel: Create a service class

Parameter	Required or Optional	Description	Examples
Service Class Name	Required	Eight characters describing the service class. Service class names must be unique within a service definition	DEF_SC EXE_SC PES_SC
Description	Optional	A description of the service class (up to 32 characters.)	Default Service Class for MQWF Service Class for MQWF EXE Srv
Workload Name	Required	The name of the workload associated with the service class.	MQFWOLO
Base Resource Group	Optional	The resource group name associated with this service class.	
Base Goal information	Required	You can add, edit, or delete periods. At least one period must be defined.	Discretionary

Discretionary goals are available for work to be run when the system has some unused resources. We will use this type of goal here because we do not have a specific performance goal for the initial setup. If you enter the goal information, first the goal selection pop-up is displayed, as shown in Figure 20.

```

*****
* Choose a goal type for period 1          *
*                                          *
*                                          *
* 4_ 1. Average response time             *
*    2. Response time with percentile     *
*    3. Execution velocity                 *
*    4. Discretionary                     *
*                                          *
* F1=Help   F2=Split   F5=KeysHelp       *
* F9=Swap   F12=Cancel                      *
*****

```

Figure 20. WLM menu: Choose a goal type

Classification rule

The rules workload management and subsystems use to assign a service class and, optionally, a report class to a work request. A classification rule consists of one or more work qualifiers such as subsystem type, subsystem instance etc..

```

Subsystem-Type View Notes Options Help
-----
Subsystem Type Selection List for Rules      Row 1 to 14 of 14
Command ==> _____

Action Codes: 1=Create, 2=Copy, 3=Modify, 4=Browse, 5=Print, 6=Delete,
              /=Menu Bar

Action  Type      Description      -----Class-----
      ASCH      Use Modify to enter YOUR rules      Service Report
      CB        Use Modify to enter YOUR rules
      CICS      Use Modify to enter YOUR rules
      DB2       Use Modify to enter YOUR rules
      DDF       Use Modify to enter YOUR rules
      IMS       Use Modify to enter YOUR rules
      IWEB      Use Modify to enter YOUR rules
      JES       Use Modify to enter YOUR rules
      LSFM      Use Modify to enter YOUR rules
      3_ MQ        Use Modify to enter YOUR rules
      OMVS      Use Modify to enter YOUR rules
      SOM       Use Modify to enter YOUR rules
      STC       Use Modify to enter YOUR rules
      TSO       Use Modify to enter YOUR rules
  
```

Figure 21. WLM panel: Subsystem type selection list for rules

Specify action 3 next to type MQ to enter your rules.

```

Subsystem-Type Xref Notes Options Help
-----
Modify Rules for the Subsystem Type      Row 1 to 2 of 2
Command ==> _____      SCROLL ==> PAGE

Subsystem Type . : MQ      Fold qualifier names? Y (Y or N)
Description . . . Use Modify to enter YOUR rules

Action codes: A=After      C=Copy      M=Move      I=Insert rule
              B=Before      D=Delete row      R=Repeat      IS=Insert Sub-rule
                                          More ==>

Action  Type      Qualifier-----      -----Class-----
      1 TN      FMCIPGST      Start      Service Report
      1 TN      FMC*      Start      DEF_SC
                                          PES_SC
                                          EXE_SC
  
```

Figure 22. WLM panel: Modify rules for the subsystem type

Parameter	Required or Optional	Description	Examples
Subsystem Type	Required	You must use the IBM-supplied subsystem type MQ, which handles all MQSeries Workflow work.	MQ

Parameter	Required or Optional	Description	Examples
Qualifier Type	Required for each rule that you define	The work qualifier that identifies a work request to the system, such as a user ID or a transaction name.	The following qualifiers are valid for the subsystem type MQ: <ul style="list-style-type: none"> • PRI (priority) • PC (process name) • SI (subsystem instance) • SPM (subsystem parameter) • TC (transaction class) • TN (transaction name) • UI (user ID)
Qualifier Name		Name (value) of the work qualifier for the type you have selected. The qualifier name that you enter in this field must be from 1 to 8 characters long, if you need to specify a qualifier name that is longer than 8 characters, see “Appendix K. Nesting WLM classification information” on page 205. For a list of predefined MQSeries Workflow for OS/390 message types, see “Static Workflow message classification” on page 200.	Program execution server: FMCIPGST Execution server: FMC*
Service Class		The service class to be associated with this type of request.	Defaults : DEF_SC PES_SC EXE_SC

Application environment

An application environment is a group of application functions requested by a client that execute in server address spaces. A server type is defined to the OS/390 workload manager as an application environment.

You can define two application environments, one each for PES and execution servers, however, if you only want one of these server types to be controlled by WLM, you only need to define one unique application environment for the corresponding server type.

```

Application-Environment  Notes  Options  Help
-----
                                Create an Application Environment
Command ==> _____

Application Environment . . . MQWFEXEAE                                Required
Description . . . . . Appl Env for MQWF EXE Srv
Subsystem Type . . . . . MQ Required
Procedure Name . . . . . MQWFSRVP
Start Parameters . . . . . WLMAB=MQWFEXAE,WLMSN=&IWMSSNM,SRVEP=FMC
                                EMAIN,SRVNO=1_____

Limit on starting server address spaces for a subsystem instance:
2  1. No limit
   2. Single address space per system
   3. Single address space per sysplex

```

Figure 23. WLM panel: Create an application environment

Parameter	Required or Optional	Description	Examples
Application Environment	Required	Name identifying the application environment (up to 32 characters.) This must be unique in the parallel sysplex.	Execution server: MQWFEXEAE Program execution server: MQWFPESAE
Description	Optional	A description of the application environment (up to 32 characters.)	Appl Env for MQWF EXE Srv Appl Env for MQWF PES Srv
Subsystem Type	Required	Subsystem type is the name of the subsystem using application environments — in this case, always MQ.	MQ
Procedure Name	Required if you want WLM to operate in automatic mode.	Procedure name is the one to eight character name of the JCL procedure that workload management uses to start a server for the application environment work requests. For MQSeries Workflow, this is the value for the identifier <i>UniqueSystemKey</i> in Table 3 on page 11.	MQWFSRVP
	Required if you want WLM to operate in manual mode.	By leaving this field blank, WLM will operate in manual mode, and you will have to start WLM managed servers as described in “Starting WLM-managed servers when WLM is in manual mode” on page 155.	

Parameter	Required or Optional	Description	Examples
Start Parameters	Required	Start parameters are the parameters required by the JCL procedure defined in <i>Procedure Name</i> . These parameters define how workload management should start the server address spaces. For MQSeries Workflow, this parameter includes of the application environment , the subsystem instance name, the server task main program, and the number of server instances running in parallel in one address space (SRVNO). This value overrides the value specified in the machine profile – as described in “Changing the number of server instances per address space” on page 121. Note: If you specify the symbol &IWMSSNM inside the start parameters, WLM substitutes the subsystem instance name provided to WLM when the subsystem connected to it. For MQSeries Workflow for OS/390, the subsystem instance name is the name of the MQSeries started task procedure of the MQSeries queue manager that is to be used by MQSeries Workflow for OS/390.	For the execution server: WLMAE=MQWFEXEAE,WLMSN=&IWMSSNM,SRVEP=FMCEMAIN,SRVNO=1 For the program execution server: WLMAE=MQWFPESAE,WLMSN=&IWMSSNM,SRVEP=FMCXMAIN,SRVNO=1
Limit on starting server address space for a subsystem instance	Required	You can limit the numbers of servers per subsystem instance. This can be useful to limit the application environment’s server activity during testing.	Single address space per system

Installing and activating a WLM service policy in a Parallel Sysplex environment

To make your service definition have an effect in the sysplex environment you must install it on the WLM couple data set and activate a service policy. Installing the service definition overwrites any service definition previously installed on the WLM couple data set. You can install the service definition using the install function on the **Utilities** menu bar option of the **Definition** menu in the WLM ISPF application, as shown Figure 24.

```

File  Utilities  Notes  Options  Help
-----
* 1  1. Install definition          *
*   2. Extract definition          *
*   3. Activate service policy     *
*   4. Allocate couple data set    *
*   5. Allocate couple data set using CDS values *
*****

```

Figure 24. WLM menu: Install a service definition

To activate a service policy from the ISPF application, choose the **Utilities** option from the menu bar on the **Definition** menu of the WLM ISPF application, as shown in Figure 25 and Figure 26.

```
File  Utilities  Notes  Options  Help
----- *****
* 3 1. Install definition *
* 2. Extract definition *
* 3. Activate service policy *
* 4. Allocate couple data set *
* 5. Allocate couple data set using CDS values *
*****
```

Figure 25. WLM menu: Activate a service definition

```
Command ==> Policy Selection List Row 1 to 3 of 3

The following is the current Service Definition installed on the WLM
couple data set.

Name . . . . : WLMSEDE

Installed by : UID      from system SYSTEM
Installed on : 1999/08/12 at 15:11:33

Select the policy to be activated with "/"

Sel Name      Description
 /  MQWFSEPO  MQWF Service Policy
```

Figure 26. WLM panel: Policy selection list

MQSeries Workflow for OS/390 will only take full advantage of the services provided by WLM if WLM is in goal mode. To put WLM into goal mode, you can use the WLM console commands described in the WLM documentation, or more specifically enter in the OS/390 system console:

```
F WLM,MODE=GOAL
```

To IPL your OS/390 system in goal mode, you must remove the IPS= keyword from your IEASYS00 parmlib member as described in *MVS Planning: Workload Management*. Please note that the other WLM mode - the compatibility mode - is not supported in MQSeries Workflow for OS/390. To find out the current mode of WLM, enter one of the following commands:

- D WLM,SYSTEMS This version of the WLM display command returns the WLM related information (system name, mode, policy and status) for all systems in the sysplex.
- D WLM,SYSTEM=<SystemName> This version restricts the output to the specified system.

WLM administration

If you want to use WLM to administer server instances, there a few special administration tasks that you may require:

- The easiest way to switch between WLM and non-WLM mode is described in “Switching servers between WLM and non-WLM mode by importing an FDL file”.
- If you decide to use WLM in manual mode, see “Starting WLM-managed servers when WLM is in manual mode” on page 155.

Switching servers between WLM and non-WLM mode by importing an FDL file

An MQSeries Workflow for OS/390 system can either run with WLM-controlled server address spaces (for one or both of the execution server and the program execution server types), or under manual control by the workflow administrator. Switching between these modes is achieved by changing the server settings in the MQSeries Workflow runtime database. While this is rather straightforward to do, it also means that a mode change does not take effect until the administration server is restarted.

Switching servers to WLM mode

Figure 27 shows the FDL settings required to activate WLM for the execution server and the program execution server.

```
UPDATE SYSTEM '<MQWFSYSN>'
  RELATED_GROUP '<MQWFSGNM>'
  ...
  SERVER TYPE EXECUTION_SERVER
  EXTERNAL_CONTROL WLM
  EXTERNAL_CONTROL_CONTEXT '<WLMEXEAE>'
  SERVER TYPE PROGRAM_EXECUTION_SERVER
  EXTERNAL_CONTROL WLM
  EXTERNAL_CONTROL_CONTEXT '<WLMPEAE>'
END '<MQWFSYSN>'
```

Figure 27. FDL settings to switch to WLM mode

The MQSeries Workflow FDL to switch a system to WLM mode can be found in *CustHLQ.SFMCDATA(FMCHEFWM)*.

The server settings reside in the SYSTEM section of the FDL. For each server type, there is a SERVER subsection, into which additional or updated settings may be entered.

To activate the WLM settings in the MQSeries Workflow FDL, you can simply submit the ready-to-use JCL FMCHJWIB, which imports the FDL file into the MQSeries Workflow runtime database.

Switching servers to non-WLM mode

In order to deactivate WLM mode, it is sufficient to set the EXTERNAL_CONTROL parameter to NO for each WLM controlled server type and replace any previous settings of EXTERNAL_CONTROL and EXTERNAL_CONTROL_CONTEXT.

It is also possible to enable or disable WLM control separately for each server type. The example shown in Figure 28 on page 155 is an excerpt from the non-WLM reference, which can be found in *CustHLQ.SFMCDATA(FMCHEFNM)*.


```

UPDATE SYSTEM '<MQWFSYSN>'
  RELATED_GROUP '<MQWFSGNM>'
  ...
  SERVER TYPE EXECUTION_SERVER
  EXTERNAL_CONTROL NO
  SERVER TYPE PROGRAM_EXECUTION_SERVER
  EXTERNAL_CONTROL NO
END '<MQWFSYSN>'

```

Figure 28. FDL settings to switch to non-WLM mode

You can deactivate WLM control of the execution server and the program execution server by submitting the JCL FMCHJNIB. Then, the MQSeries Workflow administration server, and all other Workflow servers of the MQSeries Workflow system - has to be stopped and subsequently restarted.

Starting WLM-managed servers when WLM is in manual mode

In the OS/390 WLM application environment definition (shown in Figure 23 on page 151), the presence of a server JCL procedure in the field *Procedure Name* determines the control mode of the WLM application environment. If the procedure name (and the START command parameters) are present, then automatic control is in effect. If no JCL procedure is specified, WLM does not know how to start the address spaces, therefore, the server instances must be controlled manually.

If your WLM application environment is in manual control mode, you must create the server address spaces by issuing the appropriate START command. This START command is specified with the name of the server JCL procedure and optional parameters. To start the system, and manually start a given *NumberOfInstances* of the execution server, you would issue the following sequence of commands:

```

START UniqueSystemkey.AdminServerID
MODIFY AdminServerID,START
START UniqueSystemkey.ExeServerID,WLMAE=EXEApplicationEnvironment,
  WLMSN=QMGRSTPROC,SRVEP=FMCEMAIN,SRVNO=NumberOfInstances

```

The parameters of the second START command are exactly the same as the parameters you would specify for the Start Parameters field of the Application Environment definition shown in Figure 23 on page 151.

Note: Under manual control, the VARY WLM,APPLENV=EXEApplicationEnvironment,REFRESH system command will only terminate server address space, but it will not restart them.

Chapter 16. WLM problem determination

This chapter is intended to help you if you experiencing problems during the setup and execution of an MQSeries Workflow system with WLM controlled servers (PES and execution server).

Note: Many of the solutions described in the following sections require a shutdown of the workflow system if any changes to setup parameters are to take effect.

WLM setup problems

All WLM related settings on the MQSeries Workflow side are stored in the runtime database. Naturally, WLM itself has control over its own setup and therefore corresponding parameters have to match - this is one main source of possible setup and runtime problems. Most invalid settings (other than mismatches and typos) are found by the FDL import tool. They are listed in the table below.

Table 68. Problems importing FDL for WLM

Symptom	Possible cause	Solution
The FDL file containing system updates containing WLM-related settings fails to import.	Only partial information was provided with the keyword UPDATE.	Specify EXTERNAL_CONTROL WLM together with a valid external control context. Specify EXTERNAL_CONTROL NO without the context information and use the keyword REPLACE for the system.
	Invalid character in the context information.	The FDL syntax is more restrictive for application environment names than WLM is. Only use names without invalid characters (for example '_').

In any case, you should carefully examine the settings in both, the FDL file that contains the WLM-related parameters, and the ISPF WLM application. Settings entered in this application have to be installed and activated. The status of the current WLM settings can be checked with the appropriate WLM console commands.

Unexpected runtime behavior of MQSeries Workflow with WLM

Setup or resource problems in a dynamic and complex system like MQSeries Workflow may not always result in immediately recognizable error messages. Sometimes the system behavior is simply unexpected compared to the behavior expected from the intended setup. Symptoms hinting at setup problems that do not lead to error messages are listed below together with tips on how to diagnose and solve them. Here, we assume that the other components of MQSeries Workflow and the associated subsystems have been correctly set up and are running.

Table 69. Problems importing FDL for WLM

Symptom	Possible cause	Solution
WLM does not start any servers while work requests are waiting in the respective queues.	Is WLM in goal mode?	Put WLM into goal mode with the WLM console command: /MODIFY WLM,mode=goal
	Is there a valid WLM service definition?	Provide a WLM service definition for MQSeries Workflow using the ISPF WLM application.
	Has the service definition been installed and activated?	Install and activate the WLM service definition for MQSeries Workflow using the ISPF WLM application.
	Are the service definition settings correct?	Check the settings in the service definition for inconsistencies and mismatches with the other WLM related settings in MQSeries Workflow.
(Program) execution servers don't appear to run under the control of WLM (no application environment related messages appear in the OS/390 log).	Context information is missing or the corresponding server type(s) are set to EXTERNAL_CONTROL NO.	<ol style="list-style-type: none"> 1. Shutdown the Workflow for OS/390 system, as described in "Stopping the system" on page 85. 2. Shutdown the administration server, as described in "Stopping the administration server" on page 84. 3. Run the JCL <i>CustHLQ.SFMCNTL (FMCHJWIB)</i> that imports the FDL which sets EXTERNAL_CONTROL to WLM and provides the context information.
In spite of high load, WLM starts only one server.	Has WLM been restricted to "single server only" operation?	Remove the "single server" restriction in the WLM service definition for MQSeries Workflow using the ISPF WLM application.
Server start commands for one or all WLM controlled server types result in error messages from WLM (undefined application environment).	There may be a typo in the application environment name — either in the WLM definitions, or in the external control context of the FDL.	Make sure application environment names and the corresponding values in the external control contexts match.
Server start commands result in the wrong WLM controlled server type starting.	The application environment names for the execution server and the PES have been swapped.	Make sure the external control contexts and the application environments in the ISPF WLM application match.
After terminations of server address spaces WLM don't starts new server address spaces.	If five unexpected terminations of server address spaces occur within ten minutes, WLM will stop creating new server address spaces.	<ol style="list-style-type: none"> 1. Make any necessary changes to fix the condition that causes WLM to stop the application environment. 2. Stop the server type that was affected by the terminations using the administration console (see "Stopping servers" on page 87) in order to switch the corresponding application environment to the quiesce mode. 3. Wait until the application environment is in the state quiesced. 4. Start the server type as described in "Starting servers" on page 86.

Part 4. Appendixes

Appendix A. Program Execution Server directory

The program execution server directory contains information that is used by the program execution server. It contains the exit names, types and parameters for program invocations and program mappings. It also contains service definitions to connect to the CICS and IMS systems, and the user resolution information to execute a program under the correct user ID. You need to modify the PES directory whenever you want to add any of the following:

1. A new mapping type.
2. A new invocation type.
3. A new service.
4. A new user.

Using Buildtime, you can add OS/390 invocation and service definitions to your process model on a program's **OS/390** settings page. The PES directory provides the connection parameters for these invocations and services.

Note: Some key values in the PES directory must match the identifiers that are used in the process model in Buildtime; these dependencies are described in "PES directory dependencies on the process model's OS/390 program definitions" on page 164.

PES directory structure

This section describes the internal structure of the PES directory. You will need to understand the structure to be able to perform program execution customization and to add new services and new users.

The PES directory has a Key=Value structure that is similar to an OS/2 .ini file. The values specified for the primary and secondary keys can either define a final value, or a user-defined key. A user-defined key refers to another subsection of the current section.

User-defined keys are case-sensitive, and can be up to 32 characters long. Valid characters are: uppercase [A – Z], lowercase [a – z], and numerics [0 – 9]. Final values are case-sensitive, can consist of any characters, up to a maximum length of 254 characters.

The PES directory contains a root entry consisting of the primary key **directory**. A secondary key **programExecution** defines an area for the program execution server named **PESERVER**. The contents of the **PESERVER** section are described in the following:

- "Invocation section" on page 162
- "Mapping section" on page 162
- "Security section" on page 162

Invocation section

The **invocation** section defines each invocation type that is supported by the program execution server. For each invocation type, it defines the exit name, exit parameters, and a list of service subsections that can be accessed using that invocation type.

The “PES directory template” on page 163 already contains definitions for invocation sections for EXCI, CPIC, and MQSeries CICS and IMS Bridge invocation types.

Service subsection

The **service** subsections within the invocation section contains the following:

- Connection parameters necessary to connect to service systems using the given invocation type.
- User resolution information to translate the MQSeries Workflow user identification of the caller to a local OS/390 user ID that is known to the security system.

Connection parameters: The connection parameters provided depend on the invocation type:

- For **EXCI** invocation, the connection parameter is `applid`.
- For **CPIC** invocation, the connection parameters are `netid`, `luname`, and `mode`.
- For **MQ** invocation, the connection parameters are `queuemanager` and `inputqueuname`.
- MQ invocations using a queue manager belonging to the same MQSeries cluster as the Workflow system must not specify a queue manager. In this case, remove the string `"QUEUEMANAGER=<queuemanager>,"` from the template before importing the directory source file.

Note: Multiple parameter assignments are separated by a semicolon (;).

User resolution: User resolution information is only required if a program is to run under a local user ID associated with the MQSeries Workflow user starting the execution request. This only applies to programs that are defined in the process model with **Execution user=Yes** and **Local user=Yes**. In this case, you have to add **userID/executionUserID** pairs to provide a mapping from each MQSeries Workflow *userID* who may access that service, mapping on to the OS/390 *executionUserID* that the program is to run under.

The reason for having this mapping is that MQSeries Workflow user IDs may be up to 32 characters long, whereas OS/390 user IDs are restricted to 8 characters.

Mapping section

The **mapping** section defines the program mapping types that the program execution server supports. For each mapping type, it defines the DLL name of the exit that is used by the mapping type, and the initialization parameters. The standard program mapping type defines the default mapper that is provided with MQSeries Workflow for OS/390.

Security section

The **security** section is reserved for future use and must not be modified.

PES directory template

A PES directory template file is provided in *CustHLQ.SFMCDATA(FMCHEDTP)*. It contains definitions for the invocations types (EXCI, CPIC, MQCICS, and MQIMS) and the DEFAULT mapping type. It contains:

- A service section for each invocation type.
- A user section for each service section.

These have to be completed during program execution customization. The template contains the following:

```
;//*****
;//*
;//* Description: Program Execution Server Directory Template
;//*
;//*****
; Area of PES directory

(directory)

    programExecution      =keyToAreaOfPES
; Area of PES1

(keyToAreaOfPES1)
    pesName               =PESERVER
    invocation            =keyToInvocation
    security              =keyToSecurity
    mapping               =keyToMapping

; Invocation section of PES1

(keyToInvocation1)
    type                  =EXCI
    exitName              =FMCH0IEC
    exitParameters       =
    service               =invocation1Service

; Service section of PES1

(invocation1Service1)
    type                  =CICS
    name                  =CICSEHCI
    connectionParameters =APPLID=<applid>;TRANSID=CSMI
    user                  =invocation1Service1User

; User section of PES directory

(invocation1Service1User1)
    userID                =<user1>
    executionUserID       =<executionUser1>
(invocation1Service1User2)

    userID                =<user2>
    executionUserID       =<executionUser2>

(keyToInvocation2)

    type                  =CPIC
    exitName              =FMCH0ICI
    exitParameters       =
    service               =invocation2Service

(invocation2Service1)

    type                  =IMS
```

```

name =IMSCPIC
connectionParameters =NETID=<netid>;LUNAME=<luname>;MODE=#INTER
user =invocation2Service1User

(invocation2Service1User1)

userID =<user1>
executionUserID =<executionUser1>

(keyToInvocation3)

type =MQCICS
exitName =FMCH0ICM
exitParameters =
service =invocation3Service

(invocation3Service1)

type =CICS
name =CICSMQBR
connectionParameters =QUEUEMANAGER=<queuemanager>;INPUTQUEUE=<inputqueue>
user =invocation3Service1User

(invocation3Service1User1)

userID =<user1>
executionUserID =<executionUser1>

(keyToInvocation4)

type =MQIMS
exitName =FMCH0IIM
exitParameters =
service =invocation4Service

(invocation4Service1)

type =IMS
name =IMSMQBR
connectionParameters =QUEUEMANAGER=<queuemanager>;INPUTQUEUE=<inputqueue>
user =invocation4Service1User

(invocation4Service1User1)

userID =<user1>
executionUserID =<executionUser1>

(keyToMapping1)

type =DEFAULT
exitName =FMCH0XME
exitParameters =

(keyToSecurity1)

type =
exitName =
exitParameters =

```

PES directory dependencies on the process model's OS/390 program definitions

When you define a service in the PES directory, some of the key values you use must exactly match the following identifiers provided in the program's **OS/390** properties that are shown in Figure 8 on page 98. These identifiers are:

- **Service name**, for example CICSEXCI, or IMSCPIC.

- **Service type**, for example CICS, or IMS.
- **Invocation type**, for example EXCI, or CPIC.
- **Mapping type**, for example DEFAULT.

Note: The values are case-sensitive.

Appendix B. The PES directory import tool's syntax and semantics

You can start the import tool FMCH1PIT using the following options:

Table 70. PES directory import tool's options

Option	DD-names used	Description
c	FMCDIMP FMCDLOG	Creates new directory entries. If an entry already exists, an error is returned.
d	FMCDIMP FMCDLOG	Deletes existing directory entries. If an entry does not exist, an error is returned.
e	FMCDLOG	Erases everything in the directory database.
i	FMCDIMP FMCDLOG	Inserts directory entries. If an entry does not exist, it will be created. If an entry already exists, it will be replaced.
r	FMCDIMP FMCDLOG	Replaces existing directory entries. If an entry does not exist, an error is returned.

The PES directory source file containing the entries to be imported must be specified using the predefined DD-name FMCDIMP. The import tool writes information, warning, and error messages to the log file that is specified by the DD-name FMCDLOG. If you specify //FMCDLOG DD SYSOUT=*, the messages will be written to SYSOUT.

Return codes

The PES directory import tool can return the following return codes:

Table 71. PES directory import tool's return codes

Value	Description	Effect of modifications to the database
0	Successful execution	Any database modifications have been completed.
4	Warning	
12	Error	The tool has made a rollback of the transaction. The database remains unchanged.
16	Severe error	

PES directory import examples

The following JCL examples illustrate the use of the import options and DD statements.

Importing a PES directory source file

This example job imports the source file that is specified using the DD name FMCDIMP, creating the new entries in the directory.

```
//FMCHJPIF EXEC PROC=FMCHPBAT,PROGRAM=FMCH1PIT,  
//          PARM='ENVAR("_CEE_ENVFILE=DD:FMCEEENV")/  
//          c'
```

```

//*
//FMCDIMP DD DISP=SHR,DSN=CustHLQ.SFMCDATA(PES)
//FMCDLOG DD SYSOUT=*
//*

```

Importing a PES directory and writing a log file

This example job imports the source file specified by the DD name FMCDIMP by updating the contained entries in the directory. All information, warning, and error messages will be written to the log file specified by the DD name FMCDLOG.

```

//FMCHJPIF EXEC PROC=FMCHPBAT,PROGRAM=FMCH1PIT,
//          PARM='ENVAR("_CEE_ENVFILE=DD:FMCEENV")/'
//          r'
//*
//FMCDIMP DD DISP=SHR,DSN=CustHLQ.SFMCDATA(PES)
//FMCDLOG DD DISP=SHR,DSN=CustHLQ.SFMCDATA(LOG)
//*

```

Deleting the PES directory

This example job deletes the complete contents of the PES directory.

```

//FMCHJPIF EXEC PROC=FMCHPBAT,PROGRAM=FMCH1PIT,
//          PARM='ENVAR("_CEE_ENVFILE=DD:FMCEENV")/'
//          e'
//*
//FMCDLOG DD DISP=SHR,DSN=CustHLQ.SFMCDATA(LOG)
//*

```

Appendix C. Program mapping import tool syntax

The program mapping exit reads the mapping definitions from the mapping database. You must use control statements to perform the following updates to the mapping database:

- “Creating a new program mapping definition”
- “Replacing an existing program mapping definition”
- “Inserting a program mapping definition” on page 170
- “Deleting a program mapping definition” on page 170
- “Listing program mapping definitions” on page 170

All control statements have the same format: **keyword type element**

1. The first word is a **keyword** which defines the action.
2. The second word defines which **type** of program mapping definition should be processed.

Note: Valid types are: STRUCTURE, INTERFACE, USERTYPE, BACKWARDMAPPING, and FORWARDMAPPING.

3. The third word defines which **element** of this type should be processed.

Note: You can use the wildcard character '*'. You can combine wildcard control statements with non-wildcard control statements. The wildcard control statements can be used to select all elements of the mapping definition or mapping database without explicitly naming them.

C and C++ style comments are allowed. Single line comments may start with the characters '//', and multi-line comments begin with '/*', and end with '*/'.

Creating a new program mapping definition

To create a new entry, specify:

```
CREATE EntryType EntryName
```

To create all entries of a given type, specify:

```
CREATE EntryType *
```

Note: If an entry already exists, the activity is rolled back, and you will get an error message.

Replacing an existing program mapping definition

To replace a specific entry for *EntryType* and *EntryName*, use the control statement:

```
REPLACE EntryType EntryName
```

To replace all entries for *EntryType*, use the control statement:

```
REPLACE EntryType *
```

Note: If the entry does not exist, the database transaction is rolled back, and you will get an error message.

Inserting a program mapping definition

To insert a specific entry for *EntryType* and *EntryName*, use the control statement:

```
INSERT EntryType EntryName
```

For example, INSERT USERTYPE UT1

Note: If the entry already exists, it will be overwritten.

To insert all entries for *EntryType*, use the control statement:

```
INSERT EntryType *
```

Deleting a program mapping definition

To delete a specific entry for *EntryType* and *EntryName*, use the control statement:

```
DELETE EntryType EntryName
```

To delete all entries for *EntryType*, use the control statement:

```
DELETE EntryType *
```

Note: If the entry does not exist, the database transaction is rolled back, and you will get an error message.

You can delete the whole database with the control statements:

```
DELETE USERTYPE *  
DELETE FORWARDMAPPING *  
DELETE BACKWARDMAPPING *  
DELETE INTERFACE *  
DELETE STRUCTURE *
```

Listing program mapping definitions

You can list all entries for a given type, with the control statement:

```
LIST EntryType *
```

This statement lists the entries by name and type in alphabetical order.

Control statement execution

The control statements are executed on the program mapper's database as a transaction. If any of the statements fail, the whole transaction is rolled back, and an error is returned. The control statements are not necessarily executed in the order that they are defined in the control member. The control statements are executed in the following sequence:

1. Any command for **forward mapping** definitions, in alphabetical order of the forward mapping name, followed by forward mapping commands that use the wildcard.
2. Any command for **backward mapping** definitions, in alphabetical order of the backward mapping name, followed by backward mapping commands that use the wildcard.
3. Any command for **structure** definitions, in alphabetical order of the structure name, followed by structure commands that use the wildcard.
4. Any command for **interface** definitions, in alphabetical order of the interface name, followed by interface commands that use the wildcard.

5. Any command for **user type** definitions, in alphabetical order of the user type name, followed by user type commands that use the wildcard.
6. **List** commands in alphabetical order of definitions for forward mapping, backward mapping, structure, interface, and user type.

Example control statements

The following example creates all user types, mapping definitions, and replaces usertype UT1:

```
REPLACE USERTYPE UT1           // Replace existing UT1
CREATE USERTYPE *              // Create all other usertypes
```

Appendix D. Naming and code page restrictions

MQSeries Workflow for OS/390 exploits the `iconv` function set of the C/C++ Compiler on OS/390 by converting incoming messages to Unicode (UCS-2) and then to the local codepage. MQSeries Workflow for OS/390 relies on the converters available on the system and does not provide them as part of the product. Please see the *OS/390 C/C++ Programming Guide* for a list of supported unicode converters. There may be more converters available as PTF's.

You should verify that the code pages installed on all cooperating MQSeries Workflow platforms allow correct character conversion for all code points for a message round-trip.

MQSeries Workflow for OS/390 requires that certain naming restrictions are followed.

Naming Buildtime objects

The names given to MQSeries Workflow for OS/390 objects in the Buildtime should conform to the rules for naming MQSeries objects. If you follow these rules, no code page conversion problems should occur when you transfer the FDL file to the host. Names should only contain the following characters:

- Uppercase A-Z
- Lowercase a-z
- Numerics 0-9
- Period (.)
- Forward slash (/)
- Underscore (_)
- Percent sign (%)
- Parentheses (())

If you use object names which do not conform to the rules for naming MQSeries objects, your transfer method's code page conversion may corrupt the FDL data during the upload process. In this case you should upload your FDL file as a binary image, and then use the tool described in "Appendix E. FDL code page conversion tool" on page 175.

Restrictions for passwords in CICS

Passwords specified in the Logon API call from CICS programs must only include characters contained in codepage IBM-1047.

Appendix E. FDL code page conversion tool

If you have code page conversion problems when uploading your process model information, you can upload the FDL file as a binary image and then use the tool FMCH1CNV to convert the FDL file between particular source and target code pages.

Using the FDL code page conversion tool

In order to use this tool you have to transfer the FDL file as binary image to the host. The FDL file should be stored in a data set that has a variable record format. To use the fmch1cnv tool you should do the following:

1. Customize the JCL *CustHLQ.SFMCCNTL* (FMCHJCNV)
 - a. Specify the options the conversion tool should use, see "Options".
 - b. Specify the input and output files using the predefined DD-names FMCCIMP and FMCCEXP.
 - c. If you want to use a specific log file instead of SYSOUT, you should specify a data set for the DD-name FMCCLOG.
2. Submit the JCL *CustHLQ.SFMCCNTL* (FMCHJCNV)

Options

You can start the conversion tool FMCH1CNV using the following options:

Option	Argument	Description
s	source code page	Name of the code set in which the input data is encoded. If you omit this option, the code page used is taken from the input file.
t	target code page	Name of the code set to which the output data is to be converted. If you omit this option, the local code page of your system determined at runtime is used.

Notes:

1. An equal sign (=), a comma (,), a colon (:), or a blank character can be used as an option delimiter.
2. The predefined DD-names: FMCCIMP, FMCCEXP, and FMCCLOG must be used to specify the input file, output file, and log file.
3. The record length of the output data set must have at least the same length as the longest line of the FDL input file transferred to the host.
4. The FDL conversion tool writes information, warning, and error messages to the log file that is specified by the DD-name FMCCLOG. By specifying //FMCCLOG DD SYSOUT=* the messages are written to SYSOUT.
5. Please see the *OS/390 C/C++ Programming Guide* for a list of supported code set converters.

Return codes

The FDL code page conversion tool can return the following return codes:

Table 72. FDL code page conversion tool's return codes

Value	Description
0	Successful execution
4	Warning
12	Error
16	Severe error

Appendix F. FDL import/export tool

Process model information is created in the Buildtime tool, and exported in FDL file format. You must use the import/export tool to:

- Import process model information into the Workflow database.
- Translate and verify process model information that is stored in the Workflow database.
- Export process model information from the Workflow database.

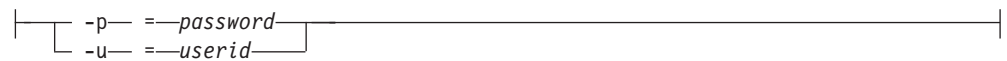
FDL import/export tool's syntax

The following syntax diagram shows how to use the FMCH0IBA tool:

Import tool FMCH0IBA syntax



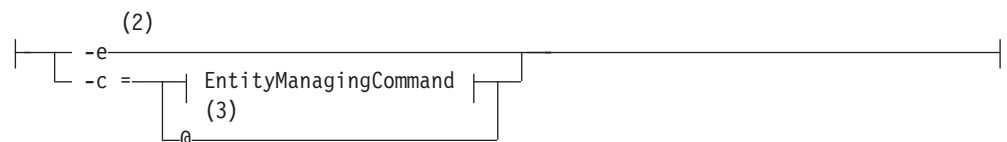
Logon:



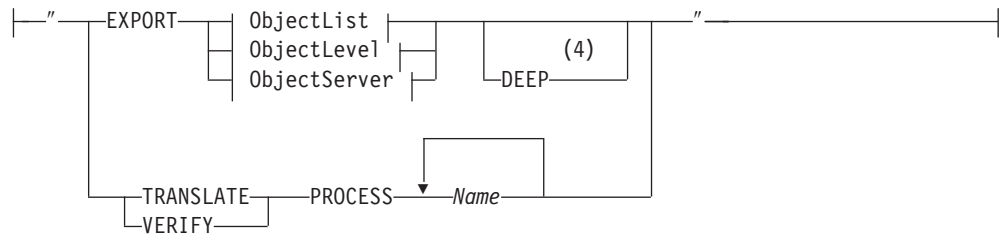
Import:



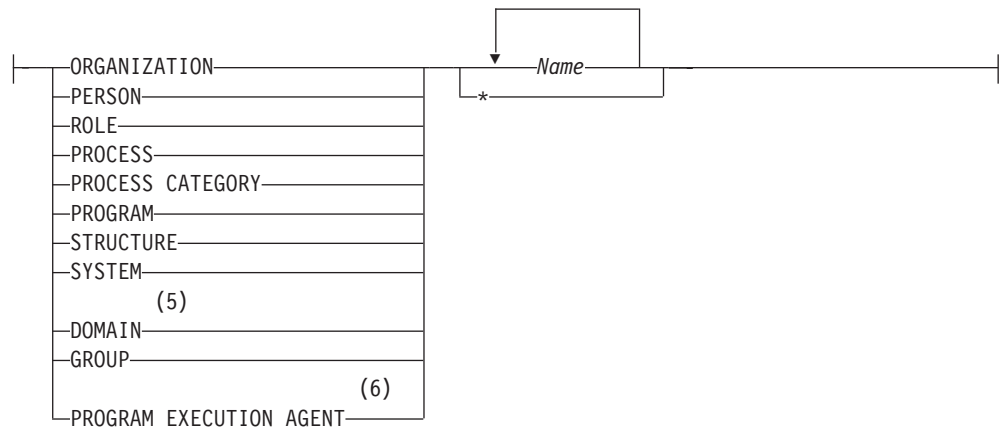
Export:



EntityManagingCommand:



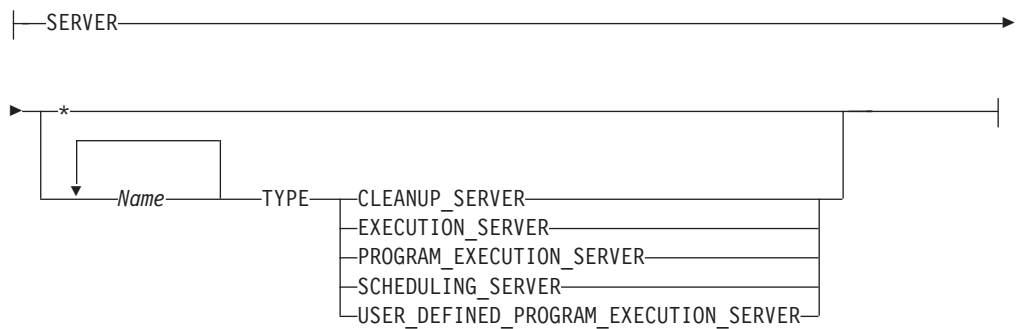
ObjectList:



ObjectLevel:



ObjectServer:



Notes:

- 1 When option **-i** is selected, the file specified by the **-DD** name **-FMCIMP** is imported into the Workflow database.

- 2 When option **-e** is selected, entities specified are exported from the Workflow database to the file specified by the DD name **-FMCIEXP**.
- 3 When a **-@** is specified after option **-c**, the commands contained in the file specified by the **-DD** name **-FMCICMD** are executed.
- 4 The **-DEEP** option is only valid for **-EXPORT PROCESS**. It means that all referenced objects, for example, nested subprocesses, are exported to the output file.
- 5 To export the domain you only have to specify the key word **-DOMAIN** without specifying the name of the entity.
- 6 The name of the **PROGRAM_EXECUTION_AGENT** is the *-PersonName* of the **RELATED_PERSON** attribute.
 - An equal sign (=), a comma (,), a colon (:), or a blank character can be used as an option delimiter.
 - You may specify any one of these options only once.
 - Either option **i** or option **c** may be specified, but not both.
 - You can use multiple words for option **c** enclosed in quotes, delimited by a blank character (space).
 - The predefined DD-names: **FMCIIMP**, **FMCIEXP**, **FMCICMD**, and **FMCILOG** must be used to specify the import file, export file, command file, and log file.
 - If you want to export entities with names that contain spaces, for example, for an entity named "Default Data Structure", you must enclose the name using two consecutive apostrophes — ' 'Default Data Structure' '.

Options for the import / export tool

You can start the import/export tool **FMCH0IBA** using the following options:

Option	Argument	Description
c	<i>"Command string"</i>	This accepts an entity command string in quotes ("). If you specify an at sign (@) the import tool executes the commands contained in the file specified by the DD-name FMCICMD .
e		Exports all workflow objects from your Workflow database to the output file specified by the DD-name FMCIEXP .
i		Imports into the Workflow database the entities from the import FDL file specified by the DD-name FMCIIMP .
o		Overwrites an existing database entity, however, only in import mode.
p	<i>password</i>	This is the password for the specified user ID.
t		Translates and verifies a process model, however, only in import mode.
u	<i>userid</i>	This is the logon user ID for the Workflow database.
v		Verifies a process model.

Log file and errors

The import tool writes information, warning, and error messages to the log file that is specified by the DD-name **FMCILOG**. By specifying **//FMCILOG DD SYSOUT=*** the messages are written to **SYSOUT**.

If the import tool detects any errors when importing a file, you will receive a non-zero return code.

Return codes

The import/export tool can return the following return codes:

Table 73. FDL import/export tool's return codes

Value	Description	Effect of modifications to the database
0	Successful execution	Any database modifications have been completed.
1	Information	
2	Warning	
4	Validation error	The tool has made a rollback of the transaction. The database remains unchanged.
8	Syntax error	
12	Error	
16	Input error	
20	Severe error	
24	Internal error	

Examples

FDL import examples

The following JCL examples show the use of the import options and DD statements. The example JCL jobs start the import tool, log on using the user ID *uid* and a password *pwd*:

To import an FDL file

This JCL job imports the FDL file that is specified by the DD name FMCIIMP.

```
//FMCHJRIF EXEC PROC=FMCHPBAT,PROGRAM=FMCH0IBA,  
//          PARM='ENVAR("_CEE_ENVFILE=DD:FMCHEENV")/  
//          -u uid -p pwd -i -o'  
//*  
//FMCIIMP DD  DISP=SHR,DSN=CustHLQ.SFMCDATA(FDL)  
//FMCILOG DD  SYSOUT=*  
//*
```

To import an FDL file and translate the contained process models

This job imports the FDL file that is specified by the DD name FMCIIMP and translates the imported process models.

```
//FMCHJRIF EXEC PROC=FMCHPBAT,PROGRAM=FMCH0IBA,  
//          PARM='ENVAR("_CEE_ENVFILE=DD:FMCHEENV")/  
//          -u uid -p pwd -i -o -t'  
//*  
//FMCIIMP DD  DISP=SHR,DSN=CustHLQ.SFMCDATA(FDL)  
//FMCILOG DD  SYSOUT=*  
//*
```

To import an FDL file and write messages in a separate log file

This job imports the FDL file that is specified by the DD name FMCIIMP. All information, warning, and error messages will be written to the log file that is specified by the DD name FMCILOG.

```
//FMCHJRIF EXEC PROC=FMCHPBAT,PROGRAM=FMCH0IBA,  
//          PARM='ENVAR("_CEE_ENVFILE=DD:FMCHEENV")/  
//          -u uid -p pwd -i -o'
```

```

//*
//FMCIIMP DD DISP=SHR,DSN=CustHLQ.SFMCDATA(FDL)
//FMCILOG DD DISP=SHR,DSN=CustHLQ.SFMCDATA(LOG)
//*
```

FDL export examples

The following JCL examples show the use of the export options and DD statements. The example JCL jobs start the import/export tool, log on using the user ID *uid* and a password *pwd*:

To export all workflow entities

This JCL job export the entities from the Workflow database to the output file specified by the DD name FMCIEXP.

```

//FMCHJRIF EXEC PROC=FMCHPBAT,PROGRAM=FMCH0IBA,
//              PARM='ENVAR("_CEE_ENVFILE=DD:FMCHEENV")/
//              -u uid -p pwd -e'
//*
```

```

//FMCIEXP DD DISP=SHR,DSN=CustHLQ.SFMCDATA(OUT)
//FMCILOG DD SYSOUT=*
//*
```

To export all people

This JCL job export the definitions for all persons from the Workflow database to the output file specified by the DD name FMCIEXP.

```

//FMCHJRIF EXEC PROC=FMCHPBAT,PROGRAM=FMCH0IBA,
//              PARM='ENVAR("_CEE_ENVFILE=DD:FMCHEENV")/
//              -u uid -p pwd -e -c"EXPORT PERSON*"'
//*
```

```

//FMCIEXP DD DISP=SHR,DSN=CustHLQ.SFMCDATA(OUT)
//FMCILOG DD SYSOUT=*
//*
```

To export individual people

This JCL job export the definitions for the people Eric and Tom from the Workflow database to the output file specified by the DD name FMCIEXP.

```

//FMCHJRIF EXEC PROC=FMCHPBAT,PROGRAM=FMCH0IBA,
//              PARM='ENVAR("_CEE_ENVFILE=DD:FMCHEENV")/
//              -u uid -p pwd -e -c"EXPORT PERSON 'ERIC' 'TOM'"
//*
```

```

//FMCIEXP DD DISP=SHR,DSN=CustHLQ.SFMCDATA(OUT)
//FMCILOG DD SYSOUT=*
//*
```

To export an individual process (deep)

This JCL job exports the definitions for the process process1 and all nested subprocesses of this process from the Workflow database to the output file specified by the DD name FMCIEXP.

```

//FMCHJRIF EXEC PROC=FMCHPBAT,PROGRAM=FMCH0IBA,
//              PARM='ENVAR("_CEE_ENVFILE=DD:FMCHEENV")/
//              -u uid -p pwd -e -c"EXPORT PROCESS process1 DEEP"'
//*
```

```

//FMCIEXP DD DISP=SHR,DSN=CustHLQ.SFMCDATA(OUT)
//FMCILOG DD SYSOUT=*
//*
```

To export Workflow entities using a command file

his job export entities from the Workflow database to the file specified by the DD name FMCIEXP, using the commands in the file that is specified by the DD name FMCI_CMD.

```

//FMCHJRIF EXEC PROC=FMCHPBAT,PROGRAM=FMCH0IBA,
//          PARM='ENVAR("_CEE_ENVFILE=DD:FMCHEENV")/'
//          -u uid -p pwd -e -c '@'
//*
//FMCICMD DD DISP=SHR,DSN=CustHLQ.SFMCDATA(CMD)
//FMCIEXP DD DISP=SHR,DSN=CustHLQ.SFMCDATA(OUT)
//FMCIOLOG DD SYSOUT=*
//*
```

Translate examples

The following JCL examples show how to translate existing process models that have already been imported into the database. The example JCL jobs start the import tool, log on using the user ID *uid* and a password *pwd*:

To translate existing models

This job translates an existing process model in the MQSeries Workflow for OS/390 runtime database with the process name *process1*.

```

//FMCHJRIF EXEC PROC=FMCHPBAT,PROGRAM=FMCH0IBA,
//          PARM='ENVAR("_CEE_ENVFILE=DD:FMCHEENV")/'
//          -u uid -p pwd
//          -c "TRANSLATE PROCESS process1"
//*
//FMCIOLOG DD SYSOUT=*
//*
```

To translate existing process models using a command file

This job translates the existing process models using the commands in the file that is specified by the DD name FMCICMD.

```

//FMCHJRIF EXEC PROC=FMCHPBAT,PROGRAM=FMCH0IBA,
//          PARM='ENVAR("_CEE_ENVFILE=DD:FMCHEENV")/'
//          -u uid -p pwd -c '@'
//*
//FMCICMD DD DISP=SHR,DSN=CustHLQ.SFMCDATA(CMD)
//FMCIOLOG DD SYSOUT=*
//*
```

Appendix G. Customization parameter files

There are three customization parameter files. During customization, you must insert your own values for identifiers that are then automatically substituted in customization jobs and templates that you require. The customization parameter files are:

- “Customization parameter file for a primary system”
- “Customization parameter file for adding a system to a system group” on page 186
- “Customization parameter file for a client on a queue manager” on page 190

Customization parameter file for a primary system

All the customization parameters for an MQSeries Workflow for OS/390 primary system are entered into the customization parameter file. Each time that you create a new MQSeries Workflow for OS/390 primary system in a new system group, you must copy and complete this file. This is done during pre-customization.

The customization parameter file template is *CustHLQ.SFMCDATA(FMCHCIF)*. During pre-customization (in step 2 of the task “Create input files for customization” on page 24) you will to enter your system’s customization parameters from the tables in “Chapter 2. Planning your configuration” on page 7. The following steps then generate the JCLs necessary to customize the Workflow system that is defined in this file.

The template file contains the following:

```
*****
**** PARAMETERS FOR CUSTOMIZATION ****
*****

**** CICS ****

*** (Used to control whether to include a CICS installation library
*** into the steplib concatenation of the jcl procedures. One of the
*** following two lines must be commented out with a '*'. If CICS
*** is not installed the parameter CICS LPFX must not be customized.)

*** if CICS is Installed
CICSFL = ' '

*** if CICS is NOT Installed
*CICSFL = '* '

*** CICS Installation High Level Qualifier
CICSLPFX = 'CICSLPFX'

**** High Level Qualifiers ****

*** DB2 Installation High Level Qualifier
DB2INHLQ = 'DB2INHLQ'

*** MQSeries Installation High Level Qualifier
MQPREFIX = 'MQPREFIX'

*** Language Environment High Level Qualifier
LELIBPFX = 'LELIBPFX'
```

```

*** C/C++ Installation High Level Qualifier
CLIBRPFX ='CLIBRPFX'

*** Cobol Installation High Level Qualifier
CBLIBPFX ='CBLIBPFX'

*** IMS Installation High Level Qualifier
IMSLIBPX ='IMSLIBPX'

*** ICONV Installation High Level Qualifier
ICONVPFX ='ICONVPFX'

*** IPCS Installation High Level Qualifier
IPCSPRFX ='IPCSPRFX'

**** Subsystems ****

*** CICS Group Name
CICSGRPN ='CICSGRPN'

*****
**** PARAMETERS FOR SCOPE DOMAIN ****
*****

*** MQWF System Prefix
*** (e.g. part of MQSeries object naming
*** and profile keys)
MQWFSYSP ='MQWFSYSP'

*****
**** PARAMETERS FOR SCOPE SYSTEM GROUP ****
*****

*** MQWF System Group Name
MQWFSGNM ='MQWFSGNM'

*** MQWF System Group Locale Setting
*** (is used to set the locale for
*** all servers and tools, use
*** 'C' for the default on your machine
*** or a specific locale setting, e.g.
*** De_DE.IBM-273 for german)
MQWFSGLC ='MQWFSGLC'

**** MQ-Series Data with System Group Scope

*** MQ-Series Cluster Name
MQWFCLST ='MQWFCLST'

*** DB2 System Group Object Qualifier
*** (is used for the DB2 objects created
*** for this System Group e.g.
*** 'DB2SGPRE'.TABLENAME)
DB2SGPRE ='DB2SGPRE'

**** DB2 Storage Group for Workflow Data ****

*** DB2 Storage Group Name
DB2STGNW ='DB2STGNW'

*** DB2 Storage Group Dataset Name Prefix
DB2STGPW ='DB2STGPW'

*** DB2 Storage Group Volume
*** (VOLUME for volume name or
*** '*' for SMS managed volumes)

```

```

DB2STGVW ='DB2STGVW'

**** DB2 Storage Group for Audit Trail Data ****

*** DB2 Storage Group Name
DB2STGNA ='DB2STGNA'

*** DB2 Storage Group Dataset Name Prefix
DB2STGPA ='DB2STGPA'

*** DB2 Storage Group Volume
***      (VOLUME for volume name or
***      '*' for SMS managed volumes)
DB2STGVA ='DB2STGVA'

**** DB2 Collections ****

*** DB2 Database Collection Name
DB2DBCOL ='DB2DBCOL'

*** DB2 PES Mapping Database Collection Name
DB2MDCOL ='DB2MDCOL'

*** DB2 PES Directory Database Collection Name
DB2PDCOL ='DB2PDCOL'

*** DB2 Plan Name
DB2PLANN ='DB2PLANN'

**** DB2 Databases ****

*** DB2 Database Name
DB2DBNAM ='DB2DBNAM'

*** DB2 PES Mapping Database Name
DB2MDBNM ='DB2MDBNM'

*** DB2 PES Directory Database Name
DB2PDBNM ='DB2PDBNM'

**** Location of DB2 Sample DSNTEP2 ****

*** Load library where DSNTEP2 is located
*** (e.g. <DB2inh1q>.RUNLIB.LOAD)
DB2SMPRL ='DB2SMPRL'

*** Library where the DSNTEP2 DBRM is located
*** (e.g. <db2inh1q>.DBRMLIB.DATA)
DB2SMPDL ='DB2SMPDL'

*****
**** PARAMETERS FOR SCOPE SYSTEM ****
*****

*** MQWF Installation High Level Qualifier
MQWFIHLQ ='MQWFIHLQ'

*** MQWF Customization High Level Qualifier
MQWFCFLQ ='MQWFCFLQ'

*** MQWF Configuration Key
MQWFCFGK ='MQWFCFGK'

*** MQWF Unique System Key
***      (e.g. for started task name)
MQWUFUKEY ='MQWUFUKEY'

```

```

*** MQWF System Name
MQWFSYSN = 'MQWFSYSN'

*** MQWF Server Started Task RACF UserId
STTSKUID = 'STTSKUID'

*** MQWF Server Started Task RACF GroupId
STTSKGRP = 'STTSKGRP'

*** CTRACE Component Name
CTRCNAME = 'CTRCNAME'

*** CTRACE Parmlib Member Suffix (Start Writer)
CTRCPMS1 = 'CTRCPMS1'

*** CTRACE Parmlib Member Suffix (Stop Writer)
CTRCPMS2 = 'CTRCPMS2'

*** CTRACE Writer Procedure Name (max. 7 chars)
CTRCWPRC = 'CTRCWPRC'

*** ARM Restart Policy Name (max. 8 chars)
ARMPOLNM = 'ARMPOLNM'

*** ARM Restart Element Name Suffix for
***   MQWF Administration Server (ADM)
***   (max. 8 chars)
ARMRESFX = 'ARMRESFX'

*** MQ-Series Cluster Namelist Name
MQCLNAME = 'MQCLNAME'

*** MQSeries Queue Manager Name
MQQMNAME = 'MQQMNAME'

*** WLM Application Environment Name for
***   MQWF Execution Server (EXE)
***   (max. 32 chars)
WLMAEEXE = 'WLMAEEXE'

*** WLM Application Environment Name for
***   MQWF Program Execution Server (PES)
***   (max. 32 chars)
WLMAEPES = 'WLMAEPES'

*** TCP/IP Address of the OS/390 Subsystem
***   (MQSeries Queue Manager)
STCPADDR = 'STCPADDR'

*** TCP/IP Port of the MQSeries Queue Manager
***   (default is port 1414)
STCPPORT = 'STCPPORT'

*** DB2 Subsystem Name
DB2SSYSN = 'DB2SSYSN'

```

Customization parameter file for adding a system to a system group

All the customization parameters for adding a new system to an existing system group are entered into the customization parameter file *CustHLQ.SFMCDATA(FMCHECSY)*, it contains the following:

```

*****
***** LIST OF PARAMETERS TO CUSTOMIZE *****
*****

```



```

*****
**** PARAMETERS FOR CUSTOMIZATION ****
*****

**** CICS ****

*** (Used to control whether to include a CICS installation library
*** into the steplib concatenation of the jcl procedures. One of the
*** following two lines must be commented out with a '*'. If CICS
*** is not installed the parameter CICSLPFX must not be customized.)

CICSLFL   ='          '   *** if CICS is Installed
*CICSLFL  ='*'         '   *** if CICS is NOT Installed

***CICS Installation High Level Qualifier
* CICSLPFX ='<CICSLPFX>'
  CICSLPFX ='CICSLPFX'

**** High Level Qualifiers ****

*** DB2 Installation High Level Qualifier
* DB2INHLQ ='<DB2INHLQ>'
  DB2INHLQ ='DB2INHLQ'

*** MQSeries Installation High Level Qualifier
* MQPREFIX ='<MQPREFIX>'
  MQPREFIX ='MQPREFIX'

*** Language Environment High Level Qualifier
* LELIBPFX ='<LELIBPFX>'
  LELIBPFX ='LELIBPFX'

*** C/C++ Installation High Level Qualifier
* CLIBRPFX ='<CLIBRPFX>'
  CLIBRPFX ='CLIBRPFX'

*** Cobol Installation High Level Qualifier
* CBLIBPFX ='<CBLIBPFX>'
  CBLIBPFX ='CBLIBPFX'

*** IMS Installation High Level Qualifier
* IMSLIBPX ='<IMSLIBPX>'
  IMSLIBPX ='IMSLIBPX'

*** ICONV Installation High Level Qualifier
* ICONVPFX ='<ICONVPFX>'
  ICONVPFX ='ICONVPFX'

*** IPCS Installation High Level Qualifier
* IPCSPRFX ='<IPCSPRFX>'
  IPCSPRFX ='IPCSPRFX'

**** Subsystems ****

*** CICS Group Name
* CICSGRPN ='<CICSGRPN>'
  CICSGRPN ='CICSGRPN'

*****
**** PARAMETERS FOR SCOPE DOMAIN ****
*****

*** MQWF System Prefix
***      (e.g. part of MQSeries object naming
***      and profile keys)

```

```

MQWFSYSP = '<MQWFSYSP>'

*****
**** PARAMETERS FOR SCOPE SYSTEM GROUP ****
*****

*** MQWF System Group Name
MQWFSGNM = '<MQWFSGNM>'

*** MQWF System Group Locale Setting
*** (is used to set the locale for
*** all servers and tools, use
*** 'C' for the default on your machine
*** or a specific locale setting, e.g.
*** De_DE.IBM-273 for german)
MQWFSGLC = '<MQWFSGLC>'

**** MQ-Series Data with System Group Scope
*** MQ-Series Cluster Name
MQWFCLST = '<MQWFCLST>'

*** DB2 System Group Object Qualifier
*** (is used for the DB2 objects created
*** for this System Group e.g.
*** 'DB2SGPRE'.TABLENAME)
DB2SGPRE = '<DB2SGPRE>'

**** DB2 Storage Group for Workflow Data ****
*** DB2 Storage Group Name
DB2STGNW = '<DB2STGNW>'

*** DB2 Storage Group Dataset Name Prefix
DB2STGPW = '<DB2STGPW>'

*** DB2 Storage Group Volume
*** (VOLUME for volume name or
*** '*' for SMS managed volumes)
DB2STGVW = '<DB2STGVW>'

**** DB2 Storage Group for Audit Trail Data ****

*** DB2 Storage Group Name
DB2STGNA = '<DB2STGNA>'

*** DB2 Storage Group Dataset Name Prefix
DB2STGPA = '<DB2STGPA>'

*** DB2 Storage Group Volume
*** (VOLUME for volume name or
*** '*' for SMS managed volumes)
DB2STGVA = '<DB2STGVA>'

**** DB2 Collections ****

*** DB2 Database Collection Name
DB2DBCOL = '<DB2DBCOL>'

*** DB2 PES Mapping Database Collection Name
DB2MDCOL = '<DB2MDCOL>'

*** DB2 PES Directory Database Collection Name
DB2PDCOL = '<DB2PDCOL>'

*** DB2 Plan Name
DB2PLANN = '<DB2PLANN>'

```

```

**** DB2 Databases ****
*** DB2 Database Name
DB2DBNAM = '<DB2DBNAM>'

*** DB2 PES Mapping Database Name
DB2MDBNM = '<DB2MDBNM>'

*** DB2 PES Directory Database Name
DB2PDBNM = '<DB2PDBNM>'

*** Location of DB2 Sample DSNTEP2 ****

*** Load library where DSNTEP2 is located
*** (e.g. <DB2INHLQ>.RUNLIB.LOAD)
DB2SMPRL = 'DB2SMPRL'

*** Library where the DSNTEP2 DBRM is located
*** (e.g. <DB2INHLQ>.DBRMLIB.DATA)
DB2SMPDL = 'DB2SMPDL'

*****
**** PARAMETERS FOR SCOPE SYSTEM ****
*****

*** MQWF Installation High Level Qualifier
* MQWFIHLQ = '<MQWFIHLQ>'
MQWFIHLQ = 'MQWFIHLQ'

*** MQWF Customization High Level Qualifier
MQWFCHLQ = 'MQWFCHLQ'

*** MQWF Configuration Key
MQWFCFGK = 'MQWFCFGK'

*** MQWF Unique System Key
*** (e.g. for started task name)
MQWFKUKEY = 'MQWFKUKEY'

*** MQWF System Name
MQWFSYSN = 'MQWFSYSN'

*** MQWF System Identifier
*** This key corresponds to the
*** FDL keyword SYSTEM_IDENTIFIER and
*** must be higher than 1 !!! 1 is reserved for
*** the primary system in a system group
MQWFSYID = 'MQWFSYID'

*** MQWF Server Started Task RACF UserId
STTSKUID = 'STTSKUID'

*** MQWF Server Started Task RACF GroupId
STTSKGRP = 'STTSKGRP'

*** TRACE Component Name
CTRCNAME = 'CTRCNAME'

*** TRACE Parmlib Member Suffix (Start Writer)
CTRCPMS1 = 'CTRCPMS1'

*** TRACE Parmlib Member Suffix (Stop Writer)
CTRCPMS2 = 'CTRCPMS2'

*** TRACE Writer Procedure Name (max. 7 chars)
CTRCWPRC = 'CTRCWPRC'

```

```

*** ARM Restart Policy Name (max. 8 chars)
    ARMPOLNM ='ARMPOLNM'

*** ARM Restart Element Name Suffix for
***     MQWF Administration Server (ADM)
***     (max. 8 chars)
    ARMRESFX ='ARMRESFX'

*** MQ-Series Cluster Namelist Name
    MQCLNAME ='MQCLNAME'

*** MQSeries Queue Manager Name
    MQQMNAME ='MQQMNAME'

*** WLM Application Environment Name for
***     MQWF Execution Server (EXE)
***     (max. 32 chars)
    WLMAEEXE ='WLMAEEXE'

*** WLM Application Environment Name for
***     MQWF Program Execution Server (PES)
***     (max. 32 chars)
    WLMAEPES ='WLMAEPES'

*** TCP/IP Address of the OS/390 Subsystem
***     (MQSeries Queue Manager)
    STCPADDR ='STCPADDR'

*** TCP/IP Port of the MQSeries Queue Manager
***     (default is port 1414)
    STCPPORT ='STCPPORT'

*** DB2 Subsystem Name
    DB2SSYSN ='DB2SSYSN'

```

Customization parameter file for a client on a queue manager

All the customization parameters for adding a new client on a queue manager are entered into the customization parameter file *CustHLQ.SFMCDATA(FMCHCCCL)*, it contains the following:

```

*****
***** LIST OF PARAMETERS TO CUSTOMIZE CLIENTS *****
*****

*****
**** PARAMETERS FOR CUSTOMIZATION ****
*****

**** CICS ****

*** CICS Installation High Level Qualifier
* CICSLPFX ='<CICSLPFX>'
  CICSLPFX ='CICSLPFX'

*****
**** PARAMETERS FOR SCOPE CLIENT ****
*****

*** MQWF Installation High Level Qualifier
  MQWFIHLQ ='<MQWFIHLQ>'

*** MQWF Customization High Level Qualifier
  MQWFCHLQ ='<MQWFCHLQ>'

```

```
*** MQWF Client Customization
*** High Level Qualifier
    CLNTCHLQ = 'CLNTCHLQ'

*** MQWF Client Configuration Key
    CLNTCFGK = 'CLNTCFGK'

*** MQSeries Client Queue Manager Name
    CLQMNAME = 'CLQMNAME'

*** TCP/IP Address of the OS/390 Subsystem
*** (MQ-Series Client Queue Manager)
    CTCPPADDR = 'CTCPPADDR'

*** TCP/IP Port of MQ-Series Queue Manager
*** (Default Port is 1414)
    CTCPPORT = 'CTCPPORT'

*** CICS Group Name
    CICSGRPC = 'CICSGRPC'
```


Appendix H. Machine profiles

There are two machine profiles, these are:

- “Server machine profile”
- “Client machine profile” on page 195

Server machine profile

Each MQSeries Workflow for OS/390 system has a machine profile in *CustHLQ.SFMCDATA(FMCHEMPR)*. This profile contains system settings that affect the operation of MQSeries Workflow for OS/390 servers and tools. Some of the values are substituted automatically during customization, these must not be changed. Changes made to the machine profile will affect new server instances and tools that are started. If you want the changes to affect all running server instances then you must restart the system as described in “Restarting the system” on page 85.

Table 74. Server machine profile settings

Variable	Value may be changed?	Description
System	No	This value should be your value for <i>System</i> in Table 3 on page 11. This value is substituted from the customization parameter file, see “Customization parameter file for a primary system” on page 183.
SystemGroup	No	This value should be your value for <i>SystemGroup</i> in Table 2 on page 10. This value is substituted from the customization parameter file, see “Customization parameter file for a primary system” on page 183.
DatabaseName	No	This value should be your value for <i>WorkflowDatabaseName</i> in Table 2 on page 10. This value is substituted from the customization parameter file, see “Customization parameter file for a primary system” on page 183.
DbPlan	No	This value should be your value for <i>DB2Plan</i> in Table 2 on page 10. This value is substituted from the customization parameter file, see “Customization parameter file for a primary system” on page 183.
DbSubSystem	No	This value should be your value for <i>DB2SubSystem</i> in Table 5 on page 14. This value is substituted from the customization parameter file, see “Customization parameter file for a primary system” on page 183.
ExecutionServer OperationMode	No	For future use.
APITimeOut	Tune carefully	API time-out in milliseconds.
FMLConnectName	No	<i>QueueManager</i> and Workflow context.
FMLConnect DelayTime	Tune carefully	Interval in milliseconds to wait between consecutive retries to reconnect to the <i>QueueManager</i> .
FMC_TRACE_ CRITERIA	Yes	Determines the level of trace detail provided by newly started servers or tools, as described in “Turning tracing on” on page 129 and “Turning tracing off” on page 131. Valid values are between 0,0000,000000000 (no trace) and 99,FFFF,FFFFFFF (full trace).

Table 74. Server machine profile settings (continued)

Variable	Value may be changed?	Description
Language	Yes	The three letter language code selects which language version of the MMS messages the servers will send to the OS/390 system console. Valid values are: ENU For mixed-case U.S. English. This is the default value. ENP For uppercase U.S. English. This option may be required if you are using a double-byte character set. If other languages become available in the future, they will be found as <i>InstHLQ</i> .SFMCMSG(FMCHMxxx), where xxx is the language code.
AdminSvrsPerAS	No	The maximum number of administration servers that will be started per address space is one.
ClnupSvrsPerAS	No	The maximum number of cleanup servers that will be started per address space is one.
DistSvrsPerAS	No	For future use.
ExeSvrsPerAS	Tune carefully	The maximum number of execution servers that will be started per address space. For more information, see "Changing the number of server instances per address space" on page 121.
GwySvrsPerAS	No	For future use.
ModelSvrsPerAS	No	For future use.
PESvrsPerAS	Tune carefully	The maximum number of program execution servers that will be started per address space. For more information, see "Changing the number of server instances per address space" on page 121.
SchedSvrsPerAS	No	The maximum number of scheduling servers that will be started per address space is one.
ServerStartProc	No	This identifies the server start procedure.
WaitBetweenQ InhibitAnd Allowed	Tune carefully	Determines how many seconds a server queue is disabled for by the server stop command. "Do your transactions take longer than 30 seconds?" on page 126 describes a situation when you may wish to change this value.
PESDirectory InCache	Yes	Determines whether the PES directory contents are cached at runtime. Valid values are 0 (zero) for no caching, or 1 (one) to enable caching. The default is no caching. For more details about this option, see "Caching the PES directory at runtime" on page 108.

After customization, your machine profile will look like the following, with your values from "Chapter 2. Planning your configuration" on page 7 automatically substituted for the identifiers shown in italics:

```
*****
*
* Description: MQ WorkFlow machine profile.
*
*****
*
Configuration.<MQWFCFGK>.System:<MQWFSYSN>
Configuration.<MQWFCFGK>.SystemGroup:<MQWFSGNM>
Configuration.<MQWFCFGK>.RTDatabase:<DB2DBNAM>
Configuration.<MQWFCFGK>.DbPlan:<DB2PLANN>
Configuration.<MQWFCFGK>.DbSubSystem:<DB2SSYSN>
Configuration.<MQWFCFGK>.RTExecutionServerOperationMode:Standalone
Configuration.<MQWFCFGK>.FMLConnectName:<MQWFSYSP>.<MQWFSGNM>.<MQWFSYSN>,<MQQMNAME>
Configuration.<MQWFCFGK>.FMC_TRACE_CRITERIA:00,0000,00000000
Configuration.<MQWFCFGK>.Language:ENU
Configuration.<MQWFCFGK>.AdminSvrsPerAS:1
Configuration.<MQWFCFGK>.ClnupSvrsPerAS:1
Configuration.<MQWFCFGK>.DistSvrsPerAS:1
```


Configuration.<MQWFCFGK>.ExeSvrsPerAS:5
 Configuration.<MQWFCFGK>.GwySvrsPerAS:1
 Configuration.<MQWFCFGK>.ModelSvrsPerAS:1
 Configuration.<MQWFCFGK>.PESvrsPerAS:5
 Configuration.<MQWFCFGK>.SchedSvrsPerAS:1
 Configuration.<MQWFCFGK>.ServerStartProc:<MQWFKUKEY>
 Configuration.<MQWFCFGK>.WaitBetweenQInhibitAndAllowed:30

Client machine profile

Each MQSeries Workflow for OS/390 system has a client machine profile in *CustHLQ.SFMCDATA(FMCHCPR)*. Some of the values are substituted automatically during customization, these must not be changed.

Table 75. Client machine profile settings

Variable	Value may be changed?	Description
System	No	This value should be your value for <i>System</i> in Table 3 on page 11. This value is substituted from the customization parameter file, see “Customization parameter file for a primary system” on page 183.
SystemGroup	No	This value should be your value for <i>SystemGroup</i> in Table 2 on page 10. This value is substituted from the customization parameter file, see “Customization parameter file for a primary system” on page 183.
DatabaseName	No	This value should be your value for <i>WorkflowDatabaseName</i> in Table 2 on page 10. This value is substituted from the customization parameter file, see “Customization parameter file for a primary system” on page 183.
DbPlan	No	This value should be your value for <i>DB2Plan</i> in Table 2 on page 10. This value is substituted from the customization parameter file, see “Customization parameter file for a primary system” on page 183.
DbSubSystem	No	This value should be your value for <i>DB2SubSystem</i> in Table 5 on page 14. This value is substituted from the customization parameter file, see “Customization parameter file for a primary system” on page 183.
ExecutionServerOperationMode	No	For future use.
APITimeOut	Tune carefully	API time-out in milliseconds.
FMLConnectName	No	<i>QueueManager</i> and Workflow context.
FMLConnectDelayTime	Tune carefully	Interval in milliseconds to wait between consecutive retries to reconnect to the <i>QueueManager</i> .
FMC_TRACE_CRITERIA	Yes	Determines the level of trace detail provided by newly started servers or tools, as described in “Turning tracing on” on page 129 and “Turning tracing off” on page 131. Valid values are between 0,0000,00000000 (no trace) and 99,FFFF,FFFFFFFF (full trace).
Language	Yes	The three letter language code selects which language version of the MMS messages the servers will send to the OS/390 system console. Valid values are: ENU For mixed-case U.S. English. This is the default value. ENP For uppercase U.S. English. This option may be required if you are using a double-byte character set. If other languages become available in the future, they will be found as <i>InstHLQ.SFMCMMSG(FMCHMxxx)</i> , where xxx is the language code.
AdminSvrsPerAS	No	The maximum number of administration servers that will be started per address space is one.

Table 75. Client machine profile settings (continued)

Variable	Value may be changed?	Description
ClnupSvrsPerAS	No	The maximum number of cleanup servers that will be started per address space is one.
DistSvrsPerAS	No	For future use.
ExeSvrsPerAS	Tune carefully	The maximum number of execution servers that will be started per address space. For more information, see “Changing the number of server instances per address space” on page 121.
GwySvrsPerAS	No	For future use.
ModelSvrsPerAS	No	For future use.
PESvrsPerAS	Tune carefully	The maximum number of program execution servers that will be started per address space. For more information, see “Changing the number of server instances per address space” on page 121.
SchedSvrsPerAS	No	The maximum number of scheduling servers that will be started per address space is one.
ServerStartProc	No	This identifies the server start procedure.
WaitBetweenQ InhibitAnd Allowed	Tune carefully	Determines how many seconds a server queue is disabled for by the server stop command. “Do your transactions take longer than 30 seconds?” on page 126 describes a situation when you may wish to change this value.
PESDirectory InCache	Yes	Determines whether the PES directory contents are cached at runtime. Valid values are 0 (zero) for no caching, or 1 (one) to enable caching. The default is no caching. For more details about this option, see “Caching the PES directory at runtime” on page 108.

The client machine profile contains the following before your customization values are automatically substituted:

```

*****
*
* Description: MQ WorkFlow client profile.
*
*****
*
Configuration.<MQWFCFGK>.System:<MQWFSYSN>
Configuration.<MQWFCFGK>.SystemGroup:<MQWFSGNM>
Configuration.<MQWFCFGK>.APITimeOut:180000
Configuration.<MQWFCFGK>.FMLConnectName:<MQWFSYSP>.<MQWFSGNM>.<MQWFSYSN>,<MQQMNAME>
Configuration.<MQWFCFGK>.FMLConnectDelayTime:30
Configuration.<MQWFCFGK>.FMLSegmentation:0
Configuration.<MQWFCFGK>.FMC_TRACE_CRITERIA:00,0000,00000000
Configuration.<MQWFCFGK>.Language:ENU

```

Appendix I. Environment variable files

There are two environment profiles, these are:

- “Server environment variable file”
- “Client environment file”

Server environment variable file

Each MQSeries Workflow for OS/390 system has a server environment variable file in *CustHLQ.SFMCDATA(FMCHEENV)*. This file contains system settings that affect the operation of MQSeries Workflow for OS/390 servers and tools. Some of the values are substituted automatically during customization, these must not be changed. Changes made to the environment variable file will affect new server instances and tools that are started. If you want the changes to affect all running server instances then you must restart the system as described in “Restarting the system” on page 85.

Table 76. Server environment variable file settings

Variable	Value may be changed?	Description
_ICONV_UCS2_PREFIX	No	Your value for <i>ICONVInstHLQ</i> in Table 4 on page 13.
LC_ALL	No	Selects the codepage to be used by the Workflow servers and tools. This is set to your value for <i>SystemGroupLocale</i> , see Table 2 on page 10 for more details.
FMC_SIMPLE_TRACE_ONLY	Yes	Activates simple tracing in newly started servers and tools. Valid values are YES or NO. For more information, see “Turning tracing on” on page 129.
FMC_CURRENT_CONFIG	No	
FMC_DEFAULT_CONFIGURATION	No	
FMC_ELAPSED_TIME	No	This must be set to YES.
FMC_IENV	No	This must be set to 1.

Before customization, your server customization file contains the following:

```
_ICONV_UCS2_PREFIX=<ICONVPFX>
LC_ALL=<MQWFSGLC>
FMC_SIMPLE_TRACE_ONLY=NO
FMC_CURRENT_CONFIG=<MQWFCFGK>
FMC_DEFAULT_CONFIGURATION=<MQWFCFGK>
FMC_ELAPSED_TIME=YES
FMC_IENV=1
```

Client environment file

Each MQSeries Workflow for OS/390 system has a client environment variable file in *CustHLQ.SFMCDATA(FMCHECEV)*. This file contains system settings that affect the operation of MQSeries Workflow for OS/390 clients. Some of the values are substituted automatically during customization, these must not be changed.

Table 77. Client environment variable file settings

Variable	Value may be changed?	Description
_ICONV_UCS2_PREFIX	No	Your value for <i>ICONVInstHLQ</i> in Table 4 on page 13.
LC_ALL	No	Selects the codepage to be used by the Workflow servers and tools. This is set to your value for <i>SystemGroupLocale</i> , see Table 2 on page 10 for more details.
FMC_SIMPLE_TRACE_ONLY	Yes	Activates simple tracing in newly started servers and tools. Valid values are YES or NO. For more information, see “Turning tracing on” on page 129.
FMC_CURRENT_CONFIG	No	
FMC_DEFAULT_CONFIGURATION	No	
FMC_ELAPSED_TIME	No	This must be set to YES.
FMC_IENV	No	This must be set to 1.

Before customization, your client customization file contains the following:

```

_ICONV_UCS2_PREFIX=<ICONVPFX>
LC_ALL=<MQWFSGLC>
FMC_SIMPLE_TRACE_ONLY=NO
FMC_CURRENT_CONFIG=<CLNTCFGK>
FMC_DEFAULT_CONFIGURATION=<CLNTCFGK>
FMC_ELAPSED_TIME=YES
FMC_IENV=1

```

Appendix J. WLM message classification

Each Workflow message contains a header, known as the **Work Information Header (WIH)**. This header contains the classification information for WLM. When work arrives on the Workflow server input queue, some information about this request is communicated to WLM. WLM assigns the request to a service class. At that point in time, it has a specific response time goal, and can be managed towards this goal. The WIH header is only created by workflow clients or servers that send messages to other workflow servers and it is only read by the MQSeries queue manager in order to classify the work for WLM. WIH-fields will be empty in messages that are sent to clients. The fields of the WIH that contain the relevant classification information are MQWIH_ServiceStep (described in Table 78) and MQWIH_ServiceName (described in Table 79).

Message classification namespace

The MQWIH_ServiceStep field contains the static message classification information this identifies the message type. The message type is structured in three hierarchy layers. This hierarchy allows you to specify both generic groupings (using a prefix plus '*') and single message types.

Table 78. MQWIH_ServiceStep field definition

Name	(Position, Length)	Description	Example
Prefix	(1 , 3)	Each message type starts with a fixed prefix (FMC).	FMCEPICS
Class Type	(4 , 1)	The class type specifies the kind of request, e.g. Process Execution (E), or a Query (Q).	FMCEPICS
Object Type	(5 , 2)	The object type classifies the Workflow object type, e.g. Process Model (PM), Process Template (PT) or Process Instance (PI).	FMCEPICS
Action Type	(7 , 2)	The action type specifies the specific action executed to satisfy the Workflow request, e.g. CreateAndStart (CS) or TErminate (TE).	FMCEPICS

The information in the MQWIH_ServiceStep field is passed to WLM in the 8-byte work qualifier TRANSACTION NAME (**TN**). For a complete list of all MQSeries Workflow Server Messages and their classifications see "Static Workflow message classification" on page 200.

Program Execution Server invocation information

The 32 byte MQWIH_ServiceName field only contains information for an InvokeProgram message sent to the Program Execution Server. Its value is passed to WLM in a new 32-byte work qualifier PROCESS NAME (**PC**).

Table 79. MQWIH_ServiceName field definition

Name	(Position, Length)	Description	Example
------	--------------------	-------------	---------

Table 79. MQWIH_ServiceName field definition (continued)

PES Name	(1 , 8)	The PES name is the name of the PES server instance. In the current release this is the fixed value PESERVER .	" PESERVER "
Reserved	(9 , 4)	These four bytes are reserved.	
Priority	(13 , 4)	The priority value contains to the activity extensions (except block activities). Valid entries are the numbers 0000 thru 9999. The value can be specified explicitly or obtained from a member of the input container of the activity. The value is always represented by four characters. In the FDL file, this value is represented by the keyword PRIORITY.	" 0001 "
PES Invocation Type	(17 , 8)	This defines the logical name of the invocation type that is used to invoke the program. In the FDL file, this value is represented by the keyword INVOCATION_TYPE in the OS/390 EXTERNAL section.	" EXCI "
PES Service	(25 , 8)	The service name is the logical name of the service system where the program is executed. This value is defined by the FDL keyword SERVICE in the OS/390 EXTERNAL section.	" CICSEHCI "

Static Workflow message classification

The tables below contains all MQSeries Workflow Server Messages sent to the MQSeries Workflow for OS/390 Execution Server and Program Execution Server, and the related message classification information in the MQWIH_ServiceStep field. This information is used by WLM to identify the service class for each request:

Message Type	Described in
Process Template	Table 80 on page 201
Process Template List	Table 81 on page 201
Process	Table 82 on page 201
Process InstList	Table 83 on page 201
Work Item	Table 84 on page 202
Activity	Table 85 on page 202
User Information	Table 86 on page 202
Process Monitor	Table 87 on page 203
WorkList	Table 88 on page 203
PEA/PES-Server	Table 89 on page 203
PEA/PES Reply	Table 90 on page 203
Scheduling	Table 91 on page 203
SubProcess	Table 92 on page 204
Internal Server	Table 93 on page 204

For each message type there is a unique MQWIH_ServiceStep field value constructed from the product prefix **FMC** plus the characters marked in bold typeface which indicate the class type, object type, and action type.

Process Template messages

Table 80. Process Template messages

Message Type	MQWIH_ServiceStep	Class Type	Object Type	Action Type
DelProcTempl	FMCXPTDL	X (other action)	Process Template	DeLete
QryProcTempl	FMC G PTPR	Get (property access)	Process Template	PRoperties
QryProcessTemplates	FMC Q PTMU	Query	Process Template	MUltiple
QryProcTemplInpCtrn	FMC Q PTCT	Query	Process Template	ConTainer

Process Template List messages

Table 81. Process Template List messages

Message Type	MQWIH_ServiceStep	Class Type	Object Type	Action Type
QryProcTemplLists	FMC Q LTMU	Query	List Template	MUltiple
QryProcTemplListData	FMC G LTPR	Get (property access)	List Template	PRoperties
CreateProcTemplList	FMC X LTCR	X (other action)	List Template	CReate
ChgProcTemplList	FMC X LTCR	X (other action)	List Template	CHange
DelProcTemplList	FMC X LTDL	X (other action)	List Template	DeLete

Process messages

Table 82. Process messages

Message Type	MQWIH_ServiceStep	Class Type	Object Type	Action Type
ChgProcInstDesc	FMC X PICD	X (other action)	Process Instance	Change Description
ChgProcInstName	FMC X PICN	X (other action)	Process Instance	Change Name
CreateProcInst	FMC E PICR	Execute process	Process Instance	CReate
CreateStartProcInst	FMC E PICS	Execute process	Process Instance	Create and Start
DelProcInst	FMC X PIDL	X (other action)	Process Instance	DeLete
QryProcInst	FMC G PIPR	Get (property access)	Process Instance	PRoperties
QryProcInstCtrn	FMC G PICT	Get (property access)	Process Instance	ConTainer
QryProcessInstances	FMC Q PIMU	Query	Process Instance	MUltiple
ResumeProcInst	FMC X PIRE	X (other action)	Process Instance	REsume
StartProcInst	FMC E PIST	Execute process	Process Instance	STart
SuspendProcInst	FMC X PISU	X (other action)	Process Instance	SUspend
TermProcInst	FMC X PITE	X (other action)	Process Instance	TErminate

Process InstList messages

Table 83. Process InstList messages

Message Type	MQWIH_ServiceStep	Class Type	Object Type	Action Type
QryProcInstLists	FMC Q LPMU	Query	List Process	MUltiple
QryProcInstListData	FMC G LPPR	Get (property access)	List Process	PRoperties
CreateProcInstList	FMC X LPCR	X (other action)	List Process	CReate

Table 83. Process InstList messages (continued)

Message Type	MQWIH_ServiceStep	Class Type	Object Type	Action Type
ChgProcInstList	FMCXLPCH	X (other action)	List Process	CHange
DelProcInstList	FMCXLPDL	X (other action)	List Process	DeLete

Work Item messages

Table 84. Work Item messages

Message Type	MQWIH_ServiceStep	Class Type	Object Type	Action Type
CancelCheckoutWorkItem	FMCXWICC	X (other action)	Work Item	Cancel Checkout
CheckinWorkItem	FMCEWICI	Execute process	Work Item	CheckIn
CheckoutWorkItem	FMCEWICO	Execute process	Work Item	CheckOut
ChgWorkItemDesc	FMCXWICD	X (other action)	Work Item	Change Description
ChgWorkItemName	FMCXWICN	X (other action)	Work Item	Change Name
DelWorkItem	FMCXWIDL	X (other action)	Work Item	DeLete
FinishWorkItem	FMCXWIMF	X (other action)	Work Item	Manual Finish
ForceFinishWorkItem	FMCXWIFF	X (other action)	Work Item	Force Finish
ForceRestartWorkItem	FMCXWIFR	X (other action)	Work Item	Force Restart
PutWorkItemOffHold	FMCXWIPF	X (other action)	Work Item	Put oFF hold
PutWorkItemOnHold	FMCXWIPN	X (other action)	Work Item	Put oN hold
QryWorkItem	FMCGWIPR	Query	Work Item	PRoperties
QryWorkItemCtnr	FMCGWICT	Query	Work Item	ConTainer
QryWorkItems	FMCQWIMU	Query	Work Item	MULTiple
RestartWorkItem	FMCXWIMR	X (other action)	Work Item	Manual Restart
StartSupportTool	FMCXWISS	X (other action)	Work Item	Start Support tool
StartWorkItem	FMCEWIST	Execute process	Work Item	STart
TerminateWorkItem	FMCXWITE	X (other action)	Work Item	TERminate
TransferWorkItem	FMCXWITR	X (other action)	Work Item	TRansfer

Activity messages

Table 85. Activity messages

Message Type	MQWIH_ServiceStep	Class Type	Object Type	Action Type
CancelCheckOutAct	FMCXAIACC	X (other action)	Activity Instance	Cancel Checkout
ForceFinishAct	FMCXAIFF	X (other action)	Activity Instance	Force Finish
ForceRestartAct	FMCXAIFR	X (other action)	Activity Instance	Force Restart
RescheduleAct	FMCXAIIRS	X (other action)	Activity Instance	ReSchedule
TransferAct	FMCXAITR	X (other action)	Activity Instance	TRansfer

User Information messages

Table 86. User Information messages

Message Type	MQWIH_ServiceStep	Class Type	Object Type	Action Type
ChgSubstitute	FMCXUSCU	X (other action)	USer	Change sUbstitute

Table 86. User Information messages (continued)

Message Type	MQWIH_ServiceStep	Class Type	Object Type	Action Type
QryUserDetails	FMCGUSPR	Query	USer	PRoperties
ChgAbsence	FMCXUSCA	X (other action)	USer	Change Absence

Process Monitor messages

Table 87. Process Monitor messages

Message Type	MQWIH_ServiceStep	Class Type	Object Type	Action Type
QryProcInstStatus	FMCQPISA	Query	Process Instance	StAtus
QryActCtnr	FMCQAICT	Query	Activity Instance	ConTainer

WorkList messages

Table 88. WorkList messages

Message Type	MQWIH_ServiceStep	Class Type	Object Type	Action Type
QryWorkLists	FMCQLWMU	Query	List Work	MUltiple
QryWorkListData	FMCGLWPR	Get (property access)	List Work	PRoperties
CreateWorkList	FMCXLWCR	X (other action)	List Work	CReate
ChgWorkList	FMCXLWCH	X (other action)	List Work	CHange
DelWorkList	FMCXLWDL	X (other action)	List Work	DeLete
CheckoutNext WorkItem	FMCELWCO	Execute process	List Work	CheckOut
CheckinNext WorkItem	FMCELWCI	Execute process	List Work	CheckIn
StartNextWorkItem	FMCELWST	Execute process	List Work	STart

PEA/PES Server messages

Table 89. PEA-Server messages

Message Type	MQWIH_ServiceStep	Class Type	Object Type	Action Type
InvokeProgram	FMCIPGST	Invoke program	ProGram	STart

PEA/PES Reply messages

Table 90. PEA Reply messages

Message Type	MQWIH_ServiceStep	Class Type	Object Type	Action Type
ProgramError	FMCEPGER	Execute Process	ProGram	ERorr
ProgramFinished	FMCEPGFI	Execute Process	ProGram	FIshed

Scheduling messages

Table 91. Scheduling messages

Message Type	MQWIH_ServiceStep	Class Type	Object Type	Action Type
CleanupProcInst	FMCSPICL	Intra-Server	Process Instance	CLeanup
CleanupWIs	FMCSWICL	Intra-Server	Work Item	CLeanup
AutoResumeProcess	FMCSPIAR	Intra-Server	Process Instance	Auto Resume
CheckNotifItems	FMCSXXCE	Intra-Server	XX (other)	ChEck notifItems

Table 91. Scheduling messages (continued)

Message Type	MQWIH_ServiceStep	Class Type	Object Type	Action Type
Create Notifications	FMCSXXCF	Intra-Server	XX (other)	Create notiFications

SubProcess messages

Table 92. SubProcess messages

Message Type	MQWIH_ServiceStep	Class Type	Object Type	Action Type
CreateStartSubProcInst	FMCSPPCS	Intra-Server	ProcessSub	Create and Start
SuspendSubProcInst	FMCSPPSU	Intra-Server	ProcessSub	SUspend
ResumeSubProcInst	FMCSPPRE	Intra-Server	ProcessSub	REsume
TermSubProcInst	FMCSPPSTE	Intra-Server	ProcessSub	TErminate
DelSubProcInst	FMCSPPDL	Intra-Server	ProcessSub	DeLete
SubProcInstError	FMCSPPSER	Intra-Server	ProcessSub	ERorr
SubProcInstFinished	FMCSPPFI	Intra-Server	ProcessSub	FIinished

Internal Server messages

Table 93. Internal Server messages

Message Type	MQWIH_ServiceStep	Class Type	Object Type	Action Type
ChndTxnUpdateWorkItems	FMCSWIUD	Intra-Server	Work Item	UpDate
PhysDelProcInst	FMCSPIPD	Intra-Server	Process Instance	Physical Delete
ChndTxnDelWorkItems	FMCSWIDL	Intra-Server	Work Item	DeLete

Appendix K. Nesting WLM classification information

All work qualifiers must be from one to 8 characters long. They are treated as 8 character names in the classification rules. For 32-byte MQWIH_ServiceName value passed in the 32-byte work qualifier PROCESS NAME (PC) you can use a start position to indicate how far to index into the character string. Because WLM allows only eight characters per rule, you can nest the PC work qualifier within themselves.

The example below shows the classification rules for the MQ subsystem using the PC work qualifier. You can classify with more than the allowed 8 characters by nesting service name information. In the example, all MQSeries Workflow work with service name information '0009EXCI' starting in position 13 for 8 characters and 'CICSEXCI' starting in position 25 for 8 characters is associated with service class PES_SC.

```

Subsystem-Type Xref Notes Options Help
-----
Modify Rules for the Subsystem Type Row 1 to 2 of 2
Command ==> _____ SCROLL ==> PAGE

Subsystem Type . : MQ          Fold qualifier names? Y (Y or N)
Description . . . Use Modify to enter YOUR rules

Action codes: A=After      C=Copy      M=Move      I=Insert rule
               B=Before    D=Delete row R=Repeat    IS=Insert Sub-rule
                                           More ==>

-----Qualifier-----
Action  Type      Name      Start      Service      Report
-----
_____ 1  PC          %%%0009 9
_____ 2  PC          EXCI*    17
_____ 3  PC          CICSEXCI 25          PES_SC

DEFAULTS: DEF_SC

```

Figure 29. Example of nesting WLM classification information

Using the masking notation percent character (%) allows you to replace a single character within the qualifier. This allows any character to match the position in the rule. Using the wild card notation asterisk character (*) allows you to replace multiple characters in a character string. The asterisk used in the example indicates a match for all characters starting in position 21 for 4 characters.

Glossary

This glossary defines terms and abbreviations used in this and other MQSeries Workflow for OS/390 publications. If you do not find the term you are looking for, refer to the index or the *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994.

A

administration console. The MQSeries Workflow component that accepts commands for starting and stopping systems and servers. It also allows the number of server instances to be queried.

administration server. The MQSeries Workflow component that performs administration functions within an MQSeries Workflow system.

administration server ID. This name is used when issuing administration console commands. It identifies which administration server will execute a given command. This identifier must be unique within the OS/390 image, and not more than 8 characters long.

activity. One of the steps that make up a process model. This can be a program activity, process activity, or block activity.

API. See *application programming interface*.

application environment. A WLM term for a group of application functions that are requested by clients, and execute in server address spaces. In the context of *MQSeries Workflow for OS/390*, each of the multiple instance server types (*execution server* and *program execution server*) requires a different application environment. For more details see "OS/390 Workload Manager application environments" on page 141.

application programming interface. An interface provided by the MQSeries Workflow workflow manager that enables programs to request services from the MQSeries Workflow workflow manager. The services are provided synchronously.

B

backward mapping. Conversion of output data created by an OS/390 legacy application into an MQSeries Workflow container. This conversion is performed by the *program execution server's program mapper*.

backward mapping definition. Part of the *MDL* which connects an *interface* definition and *structure* definition.

bridge. See *MQSeries bridges*.

Buildtime. An MQSeries Workflow component with a graphical user interface for creating and maintaining workflow process models, administering resources, and the system network definitions.

business importance. Defines the importance of achieving the *performance goal* associated with a *WLM service class*. There are five levels of importance. This is used to prioritize WLM managed work when there is insufficient system capacity to satisfy all *performance goals*.

C

CICS bridge. See *MQSeries bridges*.

classification rules. The way that WLM associates a work request with a *service class*. For more details see "Appendix J. WLM message classification" on page 199.

clean-up server. The MQSeries Workflow component that physically deletes information in the MQSeries Workflow run-time database, which had only been deleted logically.

compatibility mode. A WLM mode, in which you must start servers manually. See also: *goal mode*.

container API. An MQSeries Workflow API that allows programs executing under the control of MQSeries Workflow to obtain data from the input and output container of the activity and to store data in the output container of the activity. See also *data container*.

CPIC. An *invocation type* that allows the *program execution server* to run an application synchronously on an IMS service. CPIC is based on IMS/APPC.

D

data container. Storage for the input and output data of an activity or process. See also *container API*.

data structure. A named entity that consists of a set of data structure members. Input and output containers are defined by reference to a data structure, and adopt the layout of the referenced data structure type.

data structure member. One of the variables of which a *data structure* is composed.

'DEFAULT' mapping. The *mapping type* provided by IBM.

discretionary goal. Associated with a *WLM service class* that is to be run when there are unused resources.

domain. A set of MQSeries Workflow system groups which have the same meta-model, share the same staff information, and topology information. Communication between the components in the domain is via message queuing.

E

EXCI. An *invocation type* that allows the *program execution server* to run an application synchronously on a CICS service. EXCI is based on the CICS External CICS Interface provided by CICS Version 4.1 and higher to allow non-CICS applications to call programs running under CICS.

executable. The name of the program as defined to the service system.

execution user ID. The OS/390 user ID used by the *program execution server* to execute a program request.

execution server. The MQSeries Workflow component that performs the processing of process instances at runtime.

export tool. A utility program for retrieving information from the Workflow database and making it available in MQSeries Workflow Definition Language (FDL) format.

F

FDL. The MQSeries Workflow definition language used to exchange MQSeries Workflow information between MQSeries Workflow system groups. The language is used by the import and export function of MQSeries Workflow and contains the workflow definitions for staff, programs, data structures, and topology. This allows non-MQSeries Workflow components to interact with MQSeries Workflow. See also *export tool* and *import tool*.

forward mapping. Conversion of MQSeries Workflow containers into a format accepted by an OS/390 legacy application. This conversion is performed by the *program execution server's program mapper*.

forward mapping definition. Part of the *MDL* which connects a *structure* definition and *interface* definition.

G

goal mode. The *WLM* mode in which *WLM* seeks to achieve the goals associated with the *service class*. See also: *compatibility mode*.

I

import tool. A utility program that accepts information in the Workflow definition language (FDL) format and places it in a Workflow database.

IMS bridge. See *MQSeries bridges*.

interface. The definition of the data structure accepted by an OS/390 CICS or IMS legacy application. This definition is used by the *'DEFAULT' mapping* exit to convert the data to (and from) an MQSeries Workflow program's *structure*.

interface definition. Part of the *MDL* which defines the interface used by a legacy application.

interface element. Part of an *interface* definition. An interface element has a name, a *type*, and a cardinality. It is mapped on to a *structure element* by a *mapping rule*.

invocation exit. The DLL specified by the *invocation type*. The exit is based on an invocation protocol like CICS EXCI or the MQSeries CICS and IMS bridges.

invocation protocol. The way the PES connects to a service like CICS or IMS in order to invoke a program on that service system.

invocation type. The name used to identify the invocation exit to use. An invocation type must be defined in the program execution server directory, where it is associated with one or more services. In the process model, an invocation type must also be associated with each program that the PES is to be able to invoke.

L

local user. The RACF user under which the program is executed.

M

mapping definition language. The language used to define *mapping rules* for the *'DEFAULT' mapping* exit.

mapping exit. Used by the PES to convert data between MQSeries Workflow and legacy applications. The exit is identified by a mapping type defined in the *PES directory* and in *Buildtime*. The exit is only called if mapping has been enabled in *Buildtime*.

mapping rules. Part of a *forward mapping* or *backward mapping* definition that defines the mapping between individual *interface elements* and *structure elements*. Mapping rules are defined using the *mapper definition language*.

mapping type. The name used to identify which mapping exit to use. The mapping type is defined in

the *PES directory* and must match the *Buildtime* definitions for the legacy application. The mapping type provided with MQSeries Workflow for OS/390 is named 'DEFAULT'.

MDL. See *mapping definition language*.

message queuing. A communication technique provided by MQSeries that uses asynchronous messages for communication between software components.

'MQCICS'. An invocation type that allows the program execution server to run an application asynchronously on a CICS service. The corresponding invocation exit uses the MQSeries CICS Bridge invocation protocol.

'MQIMS'. An invocation type that allows the program execution server to run an application asynchronously on an IMS service. The corresponding invocation exit uses the invocation protocol MQSeries IMS Bridge.

MQSeries. The cross-platform, reliable message passing system on which the MQSeries Workflow product family is built.

MQSeries bridges. The *program execution server* supports two *asynchronous invocation types*: the MQSeries CICS bridge and the MQSeries IMS bridge.

MQSeries Workflow. The IBM product for business process automation. In this manual this term is used when referring to the MQSeries Workflow product family.

MQSeries Workflow for OS/390. This product; extending IBM's business process automation to the OS/390 platform. This term is always used to distinguish it from MQSeries Workflow for other platforms.

P

performance goal. See *response time goal*.

PES. See *program execution server*.

PES directory. See *program execution server directory*.

process activity. An activity that is part of a process model. When a process activity is executed, an instance of the process model is created and executed.

process definition. See *process model*.

process model. A set of processes represented in a process model. The processes are represented in graphical form in the process diagram. The process model contains the definitions for staff, programs, and data structures associated with the activities of the process. After having translated the process model into

a process template, the process template can be executed over and over again.

program execution server. The MQSeries Workflow for OS/390 component that manages the invocation of programs running on OS/390.

program execution server directory. The PES directory defines *invocation types*, *mapping types*, and the services where MQSeries Workflow program activities can be executed. It also contains information to map an MQSeries Workflow user ID to an OS/390 execution user ID. The PES directory must be updated when you add services and users.

program mapping import tool. Component of the MQSeries Workflow program mapping exit which reads the result of the program mapping parser and inputs the compiled program mapping definitions into the program mapping DB.

program mapping parser. Component of the MQSeries Workflow for OS/390 program mapping exit which parses the MDL and creates an intermediate file which is used by the program mapping import tool.

Q

quiesce. The WLM action that deactivates an *application environment*. This is equivalent to stopping the WLM managed servers.

R

response time goal. The desired response time for a *WLM service class*. This is defined as the number of milliseconds between a work request arriving, and the response being delivered to the client.

resume. The WLM action that activates an *application environment*. This is equivalent to starting the WLM managed servers.

S

safe application. An application that is guaranteed to execute once and only once, or not at all. A safe application is invoked in the same transactional context as the program execution request. This requires the specification of a transactional *invocation type*. MQSeries Workflow program execution normally guarantees execution at least once.

scheduling server. The MQSeries Workflow component that schedules actions based on time events, such as resuming suspended work items, or detecting overdue processes.

security routine. The routine to check whether a *local user* is allowed to access an executable on a service system with a given *invocation type*.

server. The servers that make up an MQSeries Workflow system are called *Program Execution Server*, *Execution Server*, *Administration Server*, *Scheduling Server*, and *Cleanup Server*.

service. The name of a CICS or IMS system that the *program execution server* accesses to execute programs.

service class. Part of a *WLM service definition*, each piece of WLM managed work is associated with a service class using *classification rules*. The service class defines the *performance goals* and *business importance*.

service definition. The *WLM* definition that includes the *service policy*, *service class*, *classification rules*, *performance goals*, and *business importance*. For more details see "Service definition" on page 137.

service policy. A named modification of your base *WLM service definition* that typically contains values for *service class* goals. Different service policies can be active at different times of the day, week, or month.

structure. The definition of the MQSeries Workflow structure passed into or out of an *activity* implementation. This definition is used by the 'DEFAULT' *mapping* exit to convert the data to (and from) a legacy application's *interface*.

structure element. Part of a *structure* definition. A structure element has a name, a type, and a cardinality. It is mapped on to an *interface element* by a *mapping rule*.

system. The smallest MQSeries Workflow unit within an MQSeries Workflow domain. It consists of a set of the MQSeries Workflow servers: one administration server, one or more execution server instances, and zero or more program execution server instances, and optionally, one scheduling server and/or one clean-up server.

system group. Each system group needs its own database, and contains one *system*. Multiple system groups can share the same DB2 subsystem.

T

translate. The action that converts a process model into a run-time process template.

U

user ID. An alphanumeric string that uniquely identifies an MQSeries Workflow user. MQSeries Workflow for OS/390 handles two types of user IDs, (1) MQSeries Workflow user IDs. (2) *Execution user IDs*.

user type definition. A user defined interface type. If you need to map a data type that is not supported by the default mapper type, you can define a user type,

and write a type conversion program which handles the conversion of that particular data type. This must use the user type exit.

user type interface. A user defined interface type. If you need to map a data type that is not supported by the default mapper type, you can define a user type, and write a type conversion program which handles the conversion of the particular data type. This must use the user type exit.

V

velocity goal. Specifies how fast a WLM managed piece of batch-type work should run when it is not held up by input/output. This is part of the *service class* definition.

W

WIH. See *work information header*.

WLM. See *Workload Manager*.

workflow. The sequence of activities performed in accordance with the business processes of an enterprise.

Workflow Management Coalition. A non-profit organization of vendors and users of workflow management systems. The coalition's mission is to promote workflow standards for workflow management systems to allow interoperability between different implementations.

workflow model. Synonym for *process model*.

Workflow system. See *system*.

Workload Manager (WLM). The OS/390 *Workload Manager* can be used to manage resources and the number of running server instances to achieve the *performance goals* in your *service definition*. For more details see "What is OS/390 Workload Manager?" on page 135.

work information header (WIH). Information that is used by *WLM* to determine which *service class* a request belongs to. The WIH is added to messages by the Workflow APIs. For application programmers, this is done transparently within the API calls.

Bibliography

To order any of the following publications, contact your IBM representative or IBM branch office.

MQSeries Workflow for OS/390 publications

This section lists the publications included in the MQSeries Workflow for OS/390 library.

- *MQSeries Workflow for OS/390: Customization and Administration*, SC33-7030, explains how to customize and administer an MQSeries Workflow for OS/390 system.
- *MQSeries Workflow for OS/390: Programming*, SC33-7031, explains the C, C++, Java, and Cobol application programming interfaces (APIs), and the program exits.
- *MQSeries Workflow for OS/390: Messages and Codes*, SC33-7032, explains the MQSeries Workflow for OS/390 system messages and codes.
- *MQSeries Workflow for OS/390: Program Directory*, GI10-0483, explains how to install MQSeries Workflow for OS/390.

MQSeries Workflow publications

This section lists the publications included in the MQSeries Workflow library.

- *IBM MQSeries Workflow: List of Workstation Server Processor Groups*, GH12-6357, lists the processor groups for MQSeries Workflow.
- *IBM MQSeries Workflow: Concepts and Architecture*, GH12-6285, explains the basic concepts of MQSeries Workflow. It also describes the architecture of MQSeries Workflow and how the components fit together.
- *IBM MQSeries Workflow: Getting Started with Buildtime*, SH12-6286, describes how to use Buildtime of MQSeries Workflow.
- *IBM MQSeries Workflow: Getting Started with Runtime*, SH12-6287, describes how to get started with the Client.
- *IBM MQSeries Workflow: Programming Guide*, SH12-6291, explains the application programming interfaces (APIs).

- *IBM MQSeries Workflow: Installation Guide*, SH12-6288, contains information and procedures for installing and customizing MQSeries Workflow.
- *IBM MQSeries Workflow: Administration Guide*, SH12-6289, explains how to administer an MQSeries Workflow system.

MQSeries publications

- *MQSeries for OS/390 V2R1 System Management Guide*, SC34-5374.
- *MQSeries: Cluster Support*, SC34-5349.
- *MQSeries Clients*, GC22-1632.

Workflow publications

- *IBM Systems Journal*, Vol. 36. No. 1, 1997 by Frank Leymann, Dieter Roller, you can also refer to the Internet:
<http://www.almaden.ibm.com/journal/sj361/leymann.html>
- *Workflow Handbook 1997*, published in association with WfMC, edited by Peter Lawrence

Other useful publications

- *DB2 for OS/390 Administration Guide*, SC26-8957.
- *DB2 for OS/390 SQL Reference*, SC26-8966.
- *DB2 for OS/390 Application Programming and SQL Guide*, SC26-8958.
- *DB2 for OS/390 Command Reference*, SC26-8960.
- *DB2 for OS/390 Utility Guide and Reference*, SC26-8967.
- *OS/390 MVS planning: Workload Management*, GC28-1761.
- *OS/390 MVS Programming: Workload Management Services*, GC28-1773.

Licensed books

The licensed books that were declassified in OS/390 Version 2 Release 4 appear on the OS/390 Online Library Collection, SK2T-6700. The remaining licensed books for OS/390 Version 2 appear on the OS/390 Licensed Product library, LK2T-2499, in unencrypted form.

Index

Special Characters

_ICONV_UCS2_PREFIX 197, 198

A

activity 207
 process activity 209
 program properties 100
address space
 administration 77
 changing the number of server instances per 121
 too many server instances per 126
administering program mappings 110
administration console 75, 78, 80, 81, 83, 207
administration server 8, 75, 78, 79, 86, 207
 cannot be started 124
 commands 83
 ID 207
 registering with ARM 84
 starting the 83
 stopping the 84
administration tasks 80, 81
administration tasks using Buildtime 93
AdminServerID 37, 50, 83, 86
AdminSvrsPerAS 194, 195
alias queues, MQSeries 124
API 207
APITimeOut 193, 195
APPC LU for VTAM 44
APPC/MVS transaction scheduler 44
APPCPM 44
application environment 138, 141, 207
application programming interface 207
applid 14, 106
applId, EXCIC connection
 parameter 162
ARM 76, 84
ARMRestartElementNameSuffix 84
ASCHPMxx 44
asynchronous invocation types 105
attach mode 94
audit trail storage group 9
AuditStorageGroupDataSet 9
AuditStorageGroupDataSetPrefix 10
AuditStorageGroupName 9, 10
AuditStorageGroupVolumeSet 9, 10
authorizing a user 109

B

backward mapping 170, 207
backward mapping format 99
backward mapping parameters 99
BACKWARDMAPPING 169
bibliography 211
boot queues 79
bridge 207

bridge invocation, customize MQSeries
 IMS 46
buffer pool 15
buffer pools 17
Buildtime 76, 81, 93, 101, 161, 173, 207
business importance 137, 207

C

CANCEL 126
capacity, system's 128
CCPPInstHLQ 8, 13
CEECCSD 31, 59, 69
CICS 8, 77
 API support customization 31, 59
 customize EXCI invocation 40
 customize MQSeries bridge invocation 41
 flag 13
 MQSeries bridge 39
 program types 105
 restrictions for passwords 173
 tracing in 131
CICS bridge 207
CICS stubs, MQSeries 31, 60, 69
CICSBridgeInputQueue 14, 41, 43
CICSContainer, sample program 50
CICSFlag 8, 13
CICSGroup 8, 14
CICSInstHLQ 8, 13
CICSMapping, sample program 50
CKBR 43
classification 144
classification rules 137, 207
CLB3YCSD 31, 59, 69
clean-up server 86, 207
Cleanliest 9
cleanup server 8
CLEANUP_SERVER export 178
client
 connection, customize the MQSeries 33, 62
 customization, LAN 33, 62
 runtime 128
 sample application, verify 37
 tier 78
client environment variable file 197
client identifiers 65
client machine profile 195
client request concentrator 71
ClnupSvrsPerAS 194, 196
cluster, MQSeries 3
code page conversion tool 175
command file 182
commands
 server 86
 system 84
compatibility mode 139, 207
component trace 8
concentrator, client request 71
configuration, planning your 7
connection parameters 48

connection parameters in PES
 directory 162
CONNECTION resource definition 40
ConnectionName 40
connectionParameters 106, 107, 163, 164
container API 207
CPIC 39, 48, 105, 207
 connection parameters 162
 invocation, customize IMS 44
creating a program mapping 108
CSD file 31, 60, 69
CSQ6SYSP macro 46
CSQCKB group 41
CTComponent 8, 11, 54, 130
CTIFMCxx 30, 58, 59
CTStartSuffix 8, 11, 30, 54, 58
CTStopSuffix 8, 11, 30, 54, 58, 130
CTWriter 8, 11, 30, 54, 58
CustHLQ 7, 8, 9, 65
customization
 identifiers 14
 parameter files 183
 verification 37

D

data
 container 207
 storage group 9
data structure 207
database
 requirements 15
 utility, program mapping 169
DatabaseName 193, 195
DataStorageGroup DataSetPrefix 10
DataStorageGroupDataSet 9
DataStorageGroupName 10
DataStorageGroupVolumeSet 9, 10
DatatStorageGroupName 9
DB2 77
 customization 25, 27, 28
 customization parameters 9
 data sharing 57
 prerequisite 4
 requirements 15
DB2InstHLQ 9, 13
DB2Plan 9, 11, 33, 62
DB2SubSystem 9, 14
DbPlan 193, 195
DbSubSystem 193, 195
DEEP export 178, 181
DEFAULT
 mapping 208
DEFAULT mapping type 105
defining a security profile 109
defining process models 93
defining program properties 96
defining server properties 93
definition, process 209
deleting a program mapping
 definition 113

- deleting the PES directory 168
- DFHCSDUP resource definition
 - utility 41
- DFHRPL 31, 59, 69
- DFLTUSER 42
- directory
 - database 104
 - database, PES 9
 - PES 209
 - program execution server 209
 - routine 104
- disabling
 - a program 109
 - a program mapping 110, 113
 - a program mapping type 114
- disabling an invocation type 114
- discretionary goal 208
- discretionary goals 137
- displaying
 - the system 86
- displaying server instances 89
- DistSvrsPerAS 194, 196
- domain 78, 208
- DOMAIN export 178
- DSNTEP2 25
- dump analyzer 131
- duration 99

E

- EDSALIM 31, 59, 69
- enabling a mapping type 113
- enabling a program mapping 110, 112
- EXCI 39, 48, 105, 208
- EXCI connection parameters 162
- EXCI invocation, customize CICS 40
- executable 99, 208
- executable type 99
- execution
 - invocation types, customizing 39
 - mode 109
 - samples, verify program 50
 - server 8, 75, 86, 208
 - starting and execution server 87
 - user ID 208
- EXECUTION_SERVER export 178
- execution user 115
- ExecutionServerOperationMode 193, 195
- executionUserID 106, 163
- ExeSvrsPerAS 121, 194, 196
- exitName 107, 163
- exitParameters 107, 163
- EXPORT database 178
- export tool 179, 208
- export tool return codes 180
- exporting FDL 181
- exporting process models 101
- extended trace 129, 130

F

- FDL 76, 93, 101, 179, 180, 181, 182, 208
- FDL code page conversion tool 175
- FMC_CURRENT_CONFIG 197, 198
- FMC_DEFAULT_CONFIGURATION 197, 198
- FMC_ELAPSED_TIME 197, 198

- FMC_IENV 197, 198
- FMC_SIMPLE_TRACE_ONLY 129, 197, 198
- FMC_TRACE_CRITERIA 129, 131, 193, 195
- FMCCCTL 112
- FMCDIMP 108, 167
- FMCDLOG 108
- FMCH0IBA 101, 179, 180, 182
- FMCH0IBA import/export tool
 - syntax 177
- FMCH0XME default mapper 164
- FMCH1PIT PES directory import
 - tool 107
- FMCH2CCT sample CICS program 49, 50
- FMCH2CMT sample CICS program 49, 50
- FMCH3ICS 49
- FMCH3ICS sample IMS program 49, 50
- FMCH3ICT sample IMS transaction 49
- FMCH3IMS 49
- FMCH3IMS sample IMS program 49, 50
- FMCH3IMT sample IMS transaction 49
- FMCH3xxx 32, 61, 70
- FMCHDDDB 16, 25
- FMCHDDMD 16, 28
- FMCHDDMS 16, 29
- FMCHDDMT 16, 29
- FMCHDDPD 16, 27
- FMCHDDPS 16, 27
- FMCHDDPT 16, 27
- FMCHDDST 16, 25
- FMCHDDTB 16, 26
- FMCHDDTS 16, 26
- FMCHECCL customization parameter file
 - for a client 190
- FMCHCEV client environment variable
 - file 197
- FMCHECIF customization parameter file
 - for primary system 183
- FMCHECPR client machine profile 195
- FMCHESY customization parameter file
 - for a adding a system 186
- FMCHEDTP 48, 106
- FMCHEDTP PES directory template 163
- FMCHENV 129, 180, 182
- FMCHENV server environment variable
 - file 197
- FMCHEFNM, non-WLM mode FDL 154
- FMCHEFWM, WLM mode FDL 154
- FMCHEMCT sample control
 - statements 112
- FMCHEMDL 112
- FMCHEMDL sample mapping
 - definition 112
- FMCHEMPR 126, 129, 131
- FMCHEMPR server machine profile 193
- FMCHepro 31, 60
- FMCHepRT 69
- FMCHeuPR 32, 60, 70
- FMCHJBDB 26, 27
- FMCHJBMA 29
- FMCHJBPD 28
- FMCHJBTE 25
- FMCHJCCH 27
- FMCHJCMQ 67

- FMCHJCPD 28
- FMCHJCPR 31, 60
- FMCHJCPT, CICS client customization
 - job 69
- FMCHJCTC 131
- FMCHJCTR 30, 59
- FMCHJCUT 69
- FMCHJDBP 16, 25
- FMCHJDDB 25
- FMCHJDMD 28
- FMCHJDMP SVC dump analyzer 131
- FMCHJDMQ 29, 58
- FMCHJDMS 29
- FMCHJDMT 29
- FMCHJDPD 27
- FMCHJDPS 27
- FMCHJDPT 27
- FMCHJDSC 32, 61
- FMCHJDST 25
- FMCHJDTB 26
- FMCHJDTS 26
- FMCHJEDB 26
- FMCHJEDC 26
- FMCHJEMD 29
- FMCHJEMS 29
- FMCHJEPD 27
- FMCHJEPS 27
- FMCHJETS 26
- FMCHJFDL 37
- FMCHJMPR 49, 111
- FMCHJNIB, imports non-WLM FDL 155
- FMCHJDDL 49
- FMCHJPIB 28
- FMCHJPIC 28, 48
- FMCHJPIF PES directory import
 - tool 107
- FMCHJPMQ 29, 58
- FMCHJRBS 26
- FMCHJRIB 26
- FMCHJRIF 101
- FMCHJRST 27
- FMCHJSPD 28
- FMCHJTRC 130
- FMCHJWIB, imports WLM mode
 - FDL 154
- FMCHSMEX sample mapping exit
 - definitions 113
- FMCIcmd 101, 178, 179, 182
- FMCIEXP 101, 179
- FMCIIMP 101, 178, 179, 180
- FMCILOG 101, 179, 180
- FMCIIN 111
- FMCTRC00, tool trace DD
 - statement 130
- FMCTRCxx, server trace DD
 - statement 130
- FMLConnectDelayTime 193, 195
- FMLConnectName 193, 195
- forward
 - mapping 170
- forward mapping 208
 - definition 208
 - format 99
 - parameters 99
- FORWARDMAPPING 169
- FTP 101

G

GET_INHIBITED 124, 125
goal mode 139, 208
GwySvrsPerAS 194, 196

H

high level qualifiers 13
hold queue 89

I

ICONVInstHLQ 13
import 178
import / export tool 101
import/export tool syntax 177
import tool 179, 180, 182
 PES directory 107
 return codes 180
importing process models 101
IMQB23IC 31, 60, 69
IMQS23IC 31, 60, 69
IMS 8, 77
 API support customization 32, 61
 bridge 208
 bridge invocation, customize
 MQSeries 46
 CPIC invocation, customize 44
 MQSeries bridge 39
 program types 105
IMS conversations 105
IMSBridgeInputQueue 14, 47
IMSContainer, sample program 50
IMSInstHLQ 8, 13
IMSMapping, sample program 50
input queues 79
inputQueue, MQ invocation connection
 parameters 162
installation scope identifiers 9
instances
 changing the number of running
 server 121
 per address space, changing the
 number of server 121
 per address space, too many
 server 126
 too few server 127
 too many server 128
InstHLQ 7, 9, 65, 66
interface 208
 element 208
INTERFACE 169
interface, mapping 170
invalid password 128
invocation
 customize CICS EXCI 40
 customize IMS CPIC 44
 customize MQSeries CICS bridge 41
 customize MQSeries IMS bridge 46
 exit 76, 208
 protocol 208
 routine 104
 section in PES directory 162
 type 208
invocation type 76, 98, 106, 165
 adding a new 107

invocation type 76, 98, 106, 165
 (continued)
 disabling an 114
 new 161
 user-defined 104
invocation types 105
 customizing 39
IPCS 8
IPCSInstHLQ 8, 13
IRC 40

J

Java support, customizing 34

L

LAN client customization 33, 62
Language 194, 195
language environment 8
LC_ALL 197, 198
legacy program mapping 76
LEInstHLQ 13
local user 99, 115, 208
logon 178
luname 14, 48
LUName, CPIC connection
 parameter 162

M

machine profile 121, 126
machine profile changes not
 activated 126
mapper
 program mapper 81
 user-defined mapper 104
mapping
 database 9, 104
 database utility 169
 exit 76, 208
 import tool return codes 112
 interface 170
 PES directory section 162
 program mapping 76
 properties 112
 routine 104
 routine call 99
 rules 208
 user IDs 162
mapping, backward 170
mapping, forward 170
mapping definition
 creating a new program mapping
 definition 169
 deleting a 113
 deleting a program mapping
 definition 170
 inserting a program mapping
 definition 170
 language 208
 replacing a program mapping
 definition 169
 sample 112
mapping definitions
 listing program mapping
 definitions 170

mapping structure 170
mapping type 76, 99, 165, 208
 adding a new 107
 defining a new 161
 disabling a 114
 enabling a 113
mapping types 105
mapping user type 171
MDL 209
message queuing 209
mode, CPIC connection parameter 162
model, process 209
ModelSvrsPerAS 194, 196
MQ invocation connection
 parameters 162
MQCICS 105
MQHostName 14, 15
MQIMS 105
MQInstHLQ 8, 13
MQRC_GET_INHIBITED 124, 125
MQRC_PUT_INHIBITED 124, 125
MQSeries 8, 33, 62, 77, 79, 209
 alias queues 124
 bridges 48, 105, 209
 CICS bridge 39
 customization 41
 invocation customization 41
 CICS stubs 31, 60, 69
 client connection customization 33,
 62
 customization 29, 58
 IMS bridge 39
 IMS bridge invocation
 customization 46
 Workflow 209
 Workflow client customization 34,
 35, 63
 Workflow Definition Language,
 (FDL) 208
 Workflow for OS/390 209
MQSeries cluster 3
MQSeries queue manager 4
mqwf_uid 41
mqwf_userid 42
MQWFSystemPrefix 8, 129
MQWIH_ServiceName 199
MQWIH_ServiceStep 199
MQWorkflowMachine 129

N

naming Buildtime objects 173
netid 14, 48
netId, CPIC connection parameter 162
new invocation type 161
new mapping type 107, 161
new service 161
new user 161
Notices xiii
NumberOfInstances 86

O

ORGANIZATION export 178
OTMACON 46

P

parameter files, customization 183
PARMLIB 19

- password, invalid 128
- passwords in CICS, restrictions for 173
- performance goal 137
- performance goals 209
- performance problems 127
- performance tuning 121
- PERSON export 178
- PES 75, 86, 209
- PES, cannot stop the 126
- PES directory 76, 81, 161, 209
 - administration 105
 - caching 108, 122
 - changes not activated 127
 - connection parameters 162
 - customization 27
 - database 9
 - deleting the 168
 - dependencies 164
 - import tool 107
 - import tool examples 167
 - import tool return codes 167
 - invocation section 162
 - mapping section 162
 - populating 28
 - refreshing the cache 108
 - routine 104
 - security 116
 - security section 162
 - service section 162
 - structure of 161
 - template 163
- PES mapping
 - database 9
 - DB2 customization 28
- PESDirectoryCollection 9, 11
- PESDirectoryDatabaseName 9, 11
- PESDirectoryInCache 108, 194, 196
- PESDirectorySourceFile 15
- PESERVER 94, 161, 163
- PESMappingCollection 9, 11
- PESMappingDatabaseName 9, 10
- pesName 163
- PESvrsPerAS 121, 194, 196
- PGMLIB 49
- planning your configuration 7
- PRIQTY 26, 27, 29
- problem determination 123
 - WLM 157
- problems
 - resource and performance
 - problems 127
 - server problems 123
- PROCESS 178
- process activity 209
- PROCESS CATEGORY export 178
- process definition 209
- PROCESS export 178
- process models 179, 180, 181, 182, 209
 - defining 93
 - importing and exporting 101
 - uploading 101
- PROCLIB 19
- program
 - activity 108
 - activity properties 100
 - administration tasks 81
 - disabling a 109

- program (*continued*)
 - mapper 81
 - security 116
- program execution
 - invocation types, customizing 39
 - samples 49
 - security 114
 - verifying samples 50
- PROGRAM_EXECUTION_AGENT 178
- program execution server 8, 75, 86, 209
 - component structure 104
 - directory 76, 161
 - directory administration 105
 - directory customization 48
 - properties 94
 - starting the 87
 - WLM information 199
- program execution server directory 209
 - connection parameters 162
 - invocation section 162
 - mapping section 162
 - service section 162
 - template 163
- PROGRAM_EXECUTION_SERVER
 - export 178
- PROGRAM export 178
- program mapping 76
 - changes not activated 126, 127
 - creating a 108
 - creating a definition 169
 - database 104
 - database utility 169
 - deleting a definition 113, 170
 - disabling 113
 - disabling a mapping type 114
 - enabling a 112
 - import tool return codes 112
 - inserting a definition 170
 - listing definitions 170
 - properties 112
 - replacing a definition 169
- program mappings, administering 110
- program properties 98
 - defining 96
 - in Buildtime 98
- programExecution 163
- programExecution PES directory
 - key 161
- properties
 - program 98
 - server 94
- properties, program activity 100
- PUT_INHIBITED 124, 125

Q

- queue manager 4, 19
- QUEUEMANAGER 164
- queueManager, MQ invocation
 - connection parameters 162
- queues 79
- queues, MQSeries alias 124
- queuing model in WLM 135
- quiesce 209

R

- RACF 19, 77, 81, 104, 109
- RACF profile 116

- refresh 209
- REGION 31, 59, 69
- request concentrator, client 71
- resource problems 127
- resource recovery service 109
- response time goal 209
- restarting servers 88
- return codes
 - extended trace format converter 130
 - import/export tool 180
 - PES directory import tool 167
 - program mapping import tool 112
- revoking a user 110
- ROLE export 178
- rollback 179, 180
- RRS 26, 109
- rules, mapping 208
- RUNSTATS 27, 28
- runtime client 128

S

- safe applications 109, 209
- samples, verify program execution 50
- SchedSvrsPerAS 194, 196
- scheduling server 8, 86, 209
- SCHEDULING_SERVER export 178
- security
 - checking 99
 - defining a profile 109
 - PES directory 116
 - program 116
 - program execution 114
 - routine 104
 - routine call 109
 - section in PES directory 162
- security routine 209
- SELECT 27
- server 210
 - administration commands 83
 - administration server cannot be started 124
 - administration tasks 80
 - changing the number of instances per address space 121
 - changing the number of running instances 121
 - commands 86
 - customization 32, 61
 - defining server properties 93
 - displaying server instances 89
 - problems 123
 - program execution 209
 - properties 94
 - restarting servers 88
 - scheduling 209
 - setting restrictions 96
 - starting servers 86
 - stopping servers 87
 - tier 78
 - too few instances 127
 - too many instances 128
 - too many instances per address space 126
 - trace 130
- server environment variable file 197
- server machine profile 193
- ServerGroupID 8, 11, 54

- servers
 - cannot stop 126
- ServerStartProc 194, 196
- ServerType 86
- ServerUserID 8, 11, 40, 42, 49, 54, 115
- service 98, 210
 - adding a new definition 106
 - name 107, 164
 - new 161
 - PES directory section 162
 - properties in Buildtime 98
 - type 99, 107, 165
- service class 137, 210
- service definition 137, 210
- service policy 137, 210
- ServiceName field in WIH 199
- ServiceStep field in WIH 199
- SESSIONS resource definition 40
- SFMCEMCT 112
- simple trace 129
- spool space, running out 128
- starting
 - execution servers 87
 - servers 86
 - the administration server 83
 - the program execution server 87
 - the system 84
- stopping
 - cannot stop servers 126
 - cannot stop the PES 126
 - servers 87
 - the administration server 84
 - the system 85
- structure
 - definition 210
 - element 210
 - mapping 170
- STRUCTURE 169
- STRUCTURE export 178
- subsystem identifiers 14
- support mode 94
- surrogate_id 41
- SVC dump analyzer 131
- synchronous invocation types 105
- syntax, import/export tool 177
- SYSOUT 101, 179, 182
- system 4, 8, 11, 53, 78, 193, 195, 210
 - administration tasks 80
 - capacity 128
 - commands 84
 - console 75, 80
 - customization verification 37
 - displaying the 86
 - group 3, 78, 210
 - group scope identifiers 10
 - scope identifiers 11
 - starting the system 84
 - stopping the system 85
 - trace 77, 129
- SYSTEM export 178
- SystemGroup 193, 195
- SystemGroupPrefix 10
- SystemGroupQualifier 9

T

- table space 15, 17
- tasks
 - administration 80, 81

- tasks (*continued*)
 - server administration 80
 - system administration 80
- tool trace 130
- trace
 - criteria 129
 - customization 30, 58
 - system trace 77
 - viewing the trace 130
- tracing 127, 128, 129
- tracing in CICS 131
- trademarks xv
- transaction scheduler, APPC/MVS 44
- transactions 126
- transferring files to the host 101
- TRANSID 106
- transId, EXCIC connection
 - parameter 162
- translate 178
- TRANSLATE 178, 182
- translating FDL process models 179, 180, 182
- translating process models 101
- tuning, performance 121

U

- undelivered messages 89
- UniqueSystemKey 8, 11, 12, 37, 50, 53, 83
- updating a program mapping 110
- uploading files to the host 101
- user
 - administration tasks 81
 - authorizing a user 109
 - ID 109, 115, 210
 - new 161
 - resolution 162
 - resolution information 106
 - revoking a user 110
- user-defined
 - invocation type 104
 - mapper 104
- USER_DEFINED_PROGRAM
 - _EXECUTION_SERVER 178
- user type
 - definition 210
 - mapping 171
- userID 106, 163
- USERTYPE 169

V

- velocity goal 210
- velocity goals 137
- VERIFY 178
- volumes 17
- VOLUMES 25
- VTAM, APPC LU for 44

W

- WaitBetweenQInhibitAndAllowed 126, 194, 196
- WIH (work information header) 144
- Windows NT 34, 63
- WLM 210

- WLM 210 (*continued*)
 - application environment 138, 141
 - business importance 137
 - classification 144
 - classification rules 137
 - compatibility mode 139
 - discretionary goals 137
 - goal mode 139
 - manual mode 151, 155
 - message classification 199
 - overview 135
 - parallel sysplex, in a 152
 - performance goal 137
 - problem determination 157
 - queuing model 135
 - service class 137
 - service definition 137
 - service definition, creating 145
 - service policy 137
 - ServiceName field in WIH 199
 - ServiceStep field in WIH 199
 - setting up 145
 - setup problems 157
 - switching servers between WLM and non-WLM mode by importing an FDL file 154
 - switching servers between WLM and non-WLM mode using Buildtime 94
 - unexpected runtime behavior 157
 - velocity goals 137
 - work information header 199
- work information header (WIH) 144, 199, 210
- Workflow
 - customizing the Workflow client 34, 35, 63
 - database 9
 - server customization 32, 61
 - system 75
 - customization parameters 8
 - system group 8
 - verifying the Workflow client sample application 37
- Workflow Definition Language (FDL) 208
- Workflow system 210
- WorkflowCollection 9, 11
- WorkflowDatabaseName 9, 10
- Workload Manager 210
- workload manager (WLM) 135

X

- XCFGroupName 14, 47
- XCFMemberIMS 47
- XCFMemberMQ 46
- XCFMemberName 14

Readers' Comments — We'd Like to Hear from You

IBM MQSeries Workflow for OS/390
Customization and Administration

Publication No. SC33-7030-02

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Complete	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Easy to find	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Easy to understand	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Well organized	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Applicable to your tasks	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? Yes No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name

Address

Company or Organization

Phone No.



Fold and Tape

Please do not staple

Fold and Tape



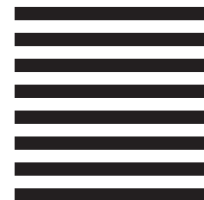
NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Deutschland Entwicklung GmbH
Information Development
Department 3248
Schoenaicher Strasse 220
71032 Boeblingen
Germany 71032-0000



Fold and Tape

Please do not staple

Fold and Tape



Program Number: 5655-A96



Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC33-7030-02



Spine information:



IBM MQSeries Workflow for
OS/390

Customization and Administration

Version 3 Release 2