IBM MQSeries Workflow for z/OS

Customization and Administration

Version 3.3

SC33-7030-05

IBM MQSeries Workflow for z/OS

Customization and Administration

Version 3.3

SC33-7030-05

Note!

Before using this information and the product it supports, be sure to read the general information under "Appendix O. Notices" on page 237.

Fifth Edition (March 2001)

This edition applies to version 3, release 3, modification 0 of IBM MQSeries Workflow for z/OS (product number 5655-BPM) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Figures

Tables

About this book

This book applies to MQSeries Workflow for z/OS and also for OS/390. All references to the operating system OS/390(R) in this book, therefore, also include z/OS.

This book provides information about customization and administration functions and practises within an IBM MQSeries Workflow for z/OS system. It explains the basic concepts of system administration and describes how to use the MQSeries Workflow administration server to administer and oversee an MQSeries Workflow for z/OS 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(R) 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 z/OS and its prerequisite and corequisite products.
- Administrates MQSeries Workflow for z/OS databases and the day-to-day operation of MQSeries Workflow for z/OS.

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

How this book is organized

- "Part 1. Customization" on page 1 describes how to customize MQSeries Workflow for z/OS. It contains the following chapters:
  - "Chapter 1. Understanding MQSeries Workflow for z/OS" on page 3 describes the MQSeries Workflow for z/OS 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 z/OS system.
  - "Chapter 5. Creating additional systems in an existing system group" on page 57 describes how to add additional Workflow systems to an existing system group.
  - "Chapter 6. Adding extra Workflow clients to an existing system" on page 69 describes how to add various types of client to an existing Workflow system.

- "Part 2. Administration" on page 77 introduces the concepts and components of system administration in an MQSeries Workflow system and explains how to start and use the MQSeries Workflow for z/OS administration server. Details regarding problem determination and using the trace facility are also given.
  - "Chapter 7. Introduction to system administration" on page 79 gives an overview of the objects and tasks involved in administrating this product.
  - "Chapter 8. Administration server tasks" on page 87 describes the command-driven administration interface that is used to start and stop the system and servers.
  - "Chapter 9. Buildtime administration tasks" on page 97 describes the administration tasks connected with the Buildtime tool, and process models.
  - "Chapter 10. Program execution" on page 105 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 119 describes how to add a Java Servelet to the WebSphere(R) 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 123 describes some specific ways to improve system performance.
  - "Chapter 13. Problem determination" on page 125 describes solutions to specific problems, and describes how to use the tracing facilities.
- "Part 3. Using OS/390 Workload Manager with Workflow" on page 143 describes how to apply OS/390 Workload Management (WLM) with MQSeries Workflow for z/OS services.
  - "Chapter 14. Introduction to WLM" on page 145 introduces workload management concepts.
  - "Chapter 15. Setting up WLM for MQSeries Workflow for z/OS" on page 155 describes how to create your service definitions, and how to migrate existing MQSeries Workflow for z/OS system groups to enable WLM support.
  - "Chapter 16. WLM problem determination" on page 167 provides help solving problems getting WLM working with MQSeries Workflow for z/OS.
- "Part 4" contains the following appendixes:
  - "Appendix A. Program Execution Server directory" on page 171 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 175 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 177 describes the database utility language used to modify the program mapper's database.
  - "Appendix D. Naming and code page restrictions" on page 181 describes the restrictions for naming OS/390 objects in the MQSeries Workflow Buildtime process model.
  - "Appendix E. FDL code page conversion tool" on page 183 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 185 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 191 describes the template files used during customization.
- "Appendix H. Configuration profiles" on page 201 describes contents of the configuration profile files that determines the behavior of servers, tools, and clients.
- "Appendix I. Environment variable files" on page 207 describes contents of the environment variable files for servers and clients.
- "Appendix J. WLM message classification" on page 209 describes the classification information for all Workflow messages sent to the execution and program execution servers.
- "Appendix K. Nesting WLM classification information" on page 215 describes how to specify qualifier names in WLM that are more than 8 characters long.
- "Appendix L. Error reporting" on page 217 describes the structure of the error log and system log records.
- "Appendix M. Audit Trail" on page 221 describes the audit trail, and provides example queries.
- 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 245.

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 z/OS documentation, choose one of the following methods:

- Send your comments by e-mail to: swsdid@de.ibm.com. Be sure to include the name of the book, the part number of the book, the version of MQSeries Workflow for z/OS, 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 (+49-(0)7031-16-4892), or by giving it to an IBM representative.

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.

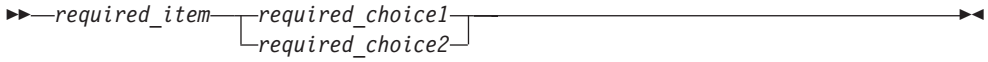
The following rules apply to the syntax diagrams used in this book:

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

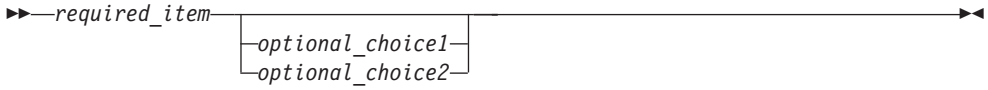
- Optional items normally appear below the main path.

  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.

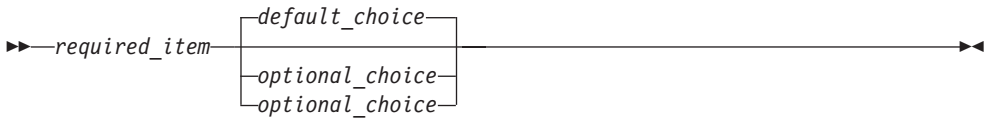
- If you can choose from two or more items, they appear vertically, in a stack.
  If you *must* choose one of the items, one item of the stack appears on the main path.



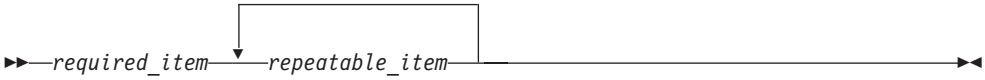
  If choosing one of the items is optional, the entire stack appears below the main path.



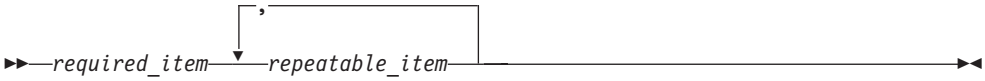
  If one of the items is the default, it appears above the main path and the remaining choices are shown below.



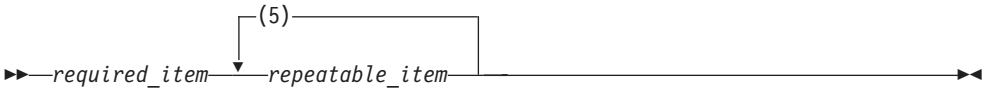
- An arrow returning to the left, above the main line, indicates an item that can be repeated.



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

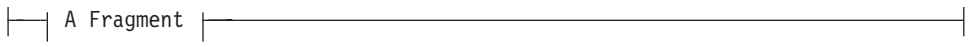


If the repeat arrow contains a number in brackets, the number represents the maximum number of times that item can appear.



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:

Summary of changes

Changes made to this book are summarized here.

Changes for this edition

This book contains information previously presented in *MQSeries Workflow for OS/390: Customization and Administration*, SC33-7030-04. It includes terminology, maintenance, corrections, and editorial changes to support MQSeries Workflow for z/OS Version 3 Release 3.

- This product has ben renamed. References to OS/390 also apply to the new version of this operating system, named z/OS.
- The DB2 customization process has been simplified, and now requires fewer jobs to be submitted.
- It is now possible to automate regular cleanups of the session records.
- "Appendix N. Migrating from a previous release" on page 229 describes how to migrate an existing MQSeries Workflow for z/OS installation.
- How to customize the distributed process sample using XML is described in "Customize the XML message API and distributed process sample using XML" on page 37.
- Support has been added for the MQSeries Workflow for z/OS Web Client using the WebSphere Application Server. How to customize the Web Client is described in "Customize the Web Client" on page 38.
- "The MQSeries Workflow for z/OS system trace facility" on page 133 describes how to use the new extended trace facility. Extended trace now uses MQSeries messages, it allows you to change the trace criteria and externalization dynamically, and you can use IPCS to analyze the trace or dump data.
- The customization parameter files have changed, they are listed in "Appendix G. Customization parameter files" on page 191.

Changes for the fourth edition

- "Hold queue commands" on page 94 describes how to display, replay, and delete undelivered messages intended for the program execution server.
- "Server properties that can be changed" on page 97 reflects that the start option for the cleanup server and scheduling server are now respected. Before, both servers were always started when the system was started, now you can determine the start up behavior by setting the start mode option for both servers in Buildtime. If you set the start mode to 'deferred', the server is not started when the system is started. You then can start the server from the system console when you want.
- "The MQSeries Workflow for z/OS system trace facility" on page 133 has been revised, and includes a full description of how extended tracing works.
- "Appendix M. Audit Trail" on page 221 describes the audit trail, and provides example queries.

Changes for the third edition

The following summarizes the most significant new information and functionality made to support MQSeries Workflow for z/OS Version 3 Release 2.1:

- "Chapter 1. Understanding MQSeries Workflow for z/OS" 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 57 describes how to add additional Workflow systems to an existing system group.
- "Chapter 6. Adding extra Workflow clients to an existing system" on page 69 describes how to add various types of client to an existing Workflow system.
- "Starting the administration server" on page 87 includes the option to register the administration server with the OS/390 Automatic Restart Manager (ARM).
- "Hold queue commands" on page 94 describes how to display, replay, and delete undelivered messages intended for the execution server.
- "Caching the PES directory at runtime" on page 110 describes how to activate and refresh the PES directory cache.
- "Chapter 11. Administering Servlets on the WebSphere Application Server" on page 119 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 143 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 z/OS

This chapter describes the architecture and customization tasks.

Understanding the MQSeries Workflow for z/OS architecture

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

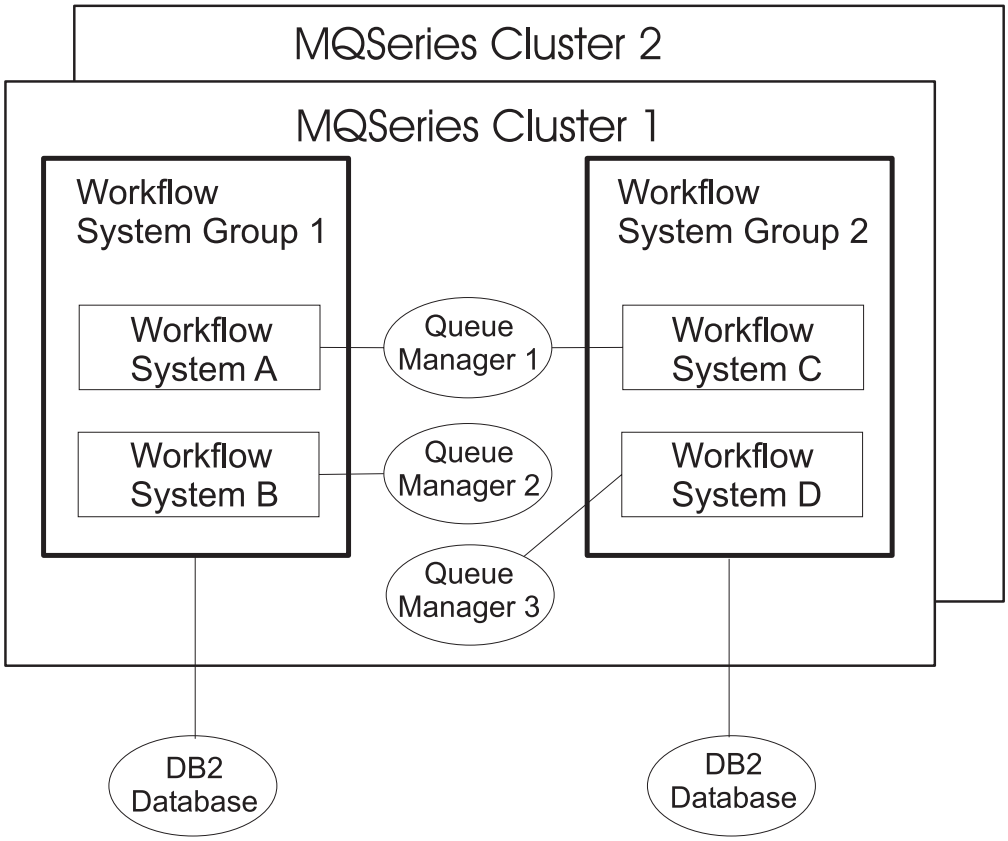


Figure 1. MQSeries Workflow for z/OS architecture

The architecture consists of the following components:

MQSeries(R) queue manager 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 Queue Manager Clusters*.

Workflow system group
A Workflow system group can contain one or more Workflow systems. All systems in a system group share the same workflow runtime database. The workflow servers in a system group are connected to MQSeries queue managers that belong to the same MQSeries queue manager cluster.

System
A Workflow system consists of servers for administration, program execution, execution, scheduling, and cleanup. All workflow servers in a system are monitored and controlled by the administration server. 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 z/OS 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 z/OS, 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 z/OS 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 57.

Understanding the customization process

The tasks required to customize MQSeries Workflow for z/OS 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 z/OS: Program Directory*.
2. After each installation, you must perform "After installing MQSeries Workflow for z/OS" 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 z/OS message catalog.
   - Modules which have to be loaded into the OS/390 Link Pack Area (LPA).

   Note: Note that the message catalog of OS/390 MVS(TM) Message Services (MMS) and the OS/390 LPA are resources that are shared, potentially by different release levels of MQSeries Workflow. Existing MQSeries Workflow installations can continue to be used with higher release levels of these resources.
3. To create each system group and a primary system, you must perform:
   a. "Deciding your MQSeries Workflow for z/OS 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 z/OS system prefix.
- MQSeries Workflow for z/OS 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 z/OS 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 57. 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 69. 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.

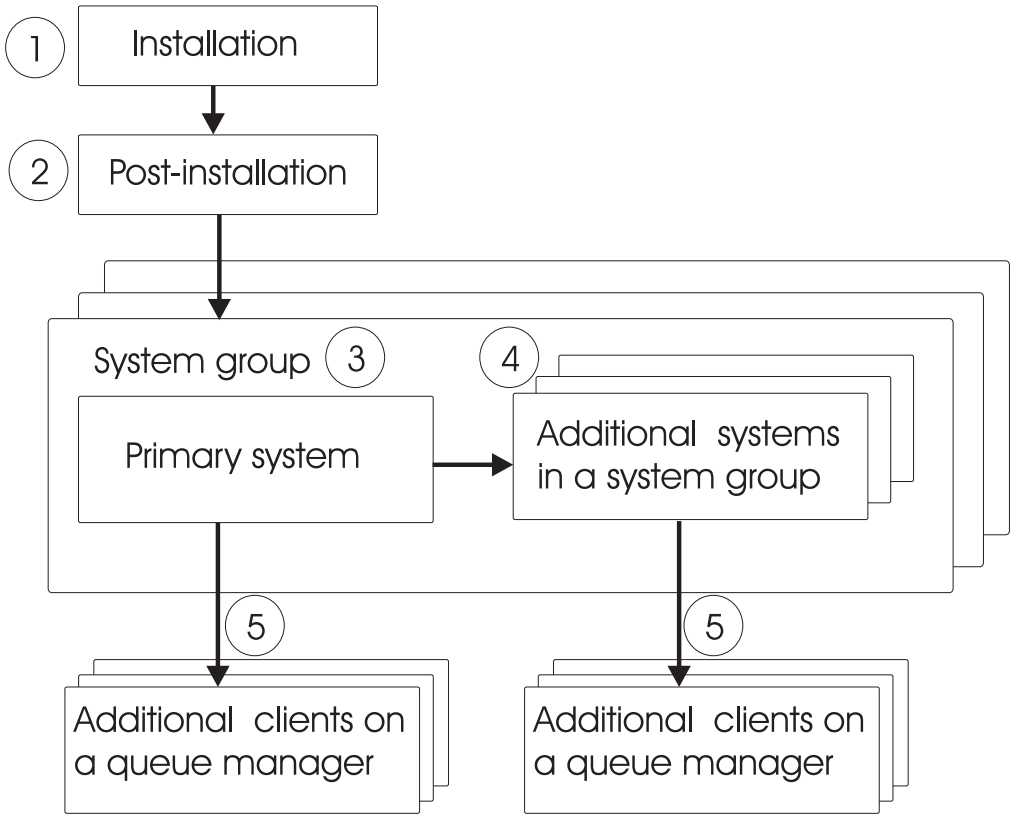


Figure 2. MQSeries Workflow for z/OS customization tasks

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 z/OS 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 z/OS 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. For high availability, 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 z/OS system, you should complete the following:

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

Note: If you want to migrate an existing MQSeries Workflow for z/OS system, you should follow the instructions described in "Appendix N. Migrating from a previous release" on page 229.

Deciding your MQSeries Workflow for z/OS identifiers

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

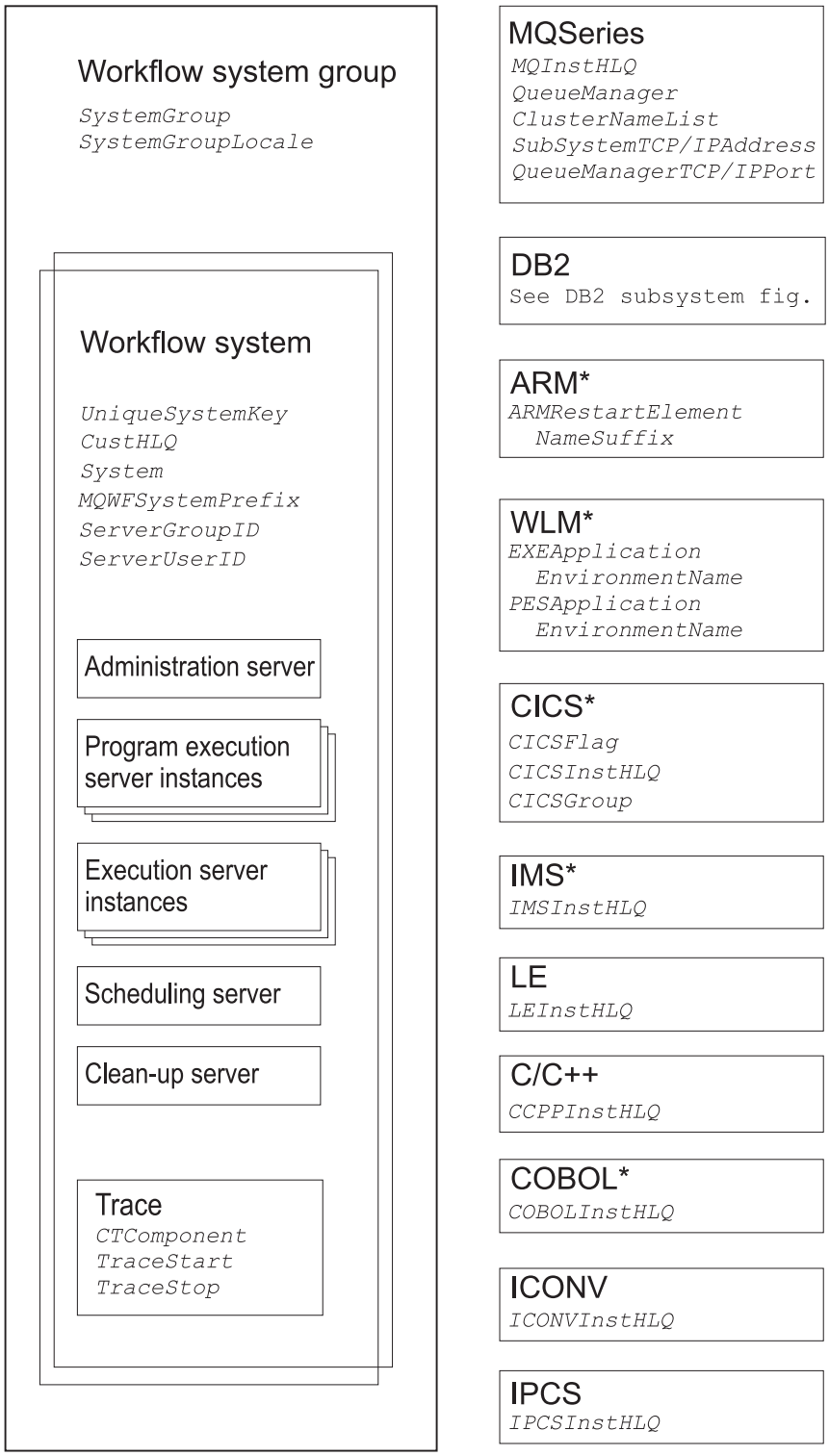
We recommend that you copy and complete the following tables for each MQSeries Workflow for z/OS system you want to plan. The identifiers decided here will be entered into the customization parameter file during "Before starting customization" on page 23. During "Create input files for customization" 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 191.

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

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

Overview of the customization parameters

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



* Using these features is optional.

Figure 3. Customization parameters for a Workflow system

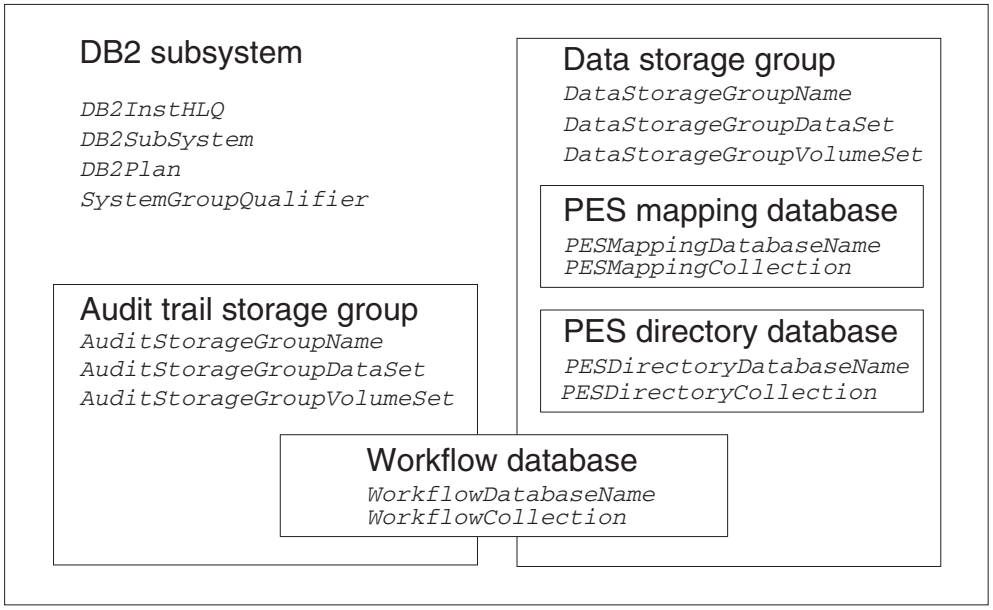


Figure 4. Customization parameters for DB2

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
InstHLQ		MQWFIHLQ	MQSeries Workflow for z/OS installation high level qualifier. This qualifier is determined when the product image is installed from tape, this is described in *MQSeries Workflow for z/OS: Program Directory*.
CustHLQ		MQWFCHLQ	The high level qualifier for the MQSeries Workflow for z/OS system you want to customize. This HLQ is required for "Chapter 4. Creating a system group and a primary system" on page 23 and "Chapter 5. Creating additional systems in an existing system group" on page 57.
ClientCustHLQ		CLNTCHLQ	Client customization high level qualifier for MQSeries Workflow. This HLQ is only required for "Chapter 6. Adding extra Workflow clients to an existing system" on page 69.

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
SystemGroup		MQWFSGNM	MQSeries Workflow for z/OS system group name. This name must be unique within your MQSeries Workflow domain.
MQWFSystemPrefix		MQWFSYSP	MQSeries Workflow for z/OS MQSeries queue prefix is used to prefix MQSeries object names, for example queue names. This identifier may be up to 8 uppercase characters long.
MQClusterName		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.
SystemGroupLocale		MQWFSGLC	MQSeries Workflow for z/OS 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 z/OS 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.
DB2SampleLoad Library		DB2SMPRL	The load library where the DB2 sample DSNTEP2 is located for example, <DB2InstHLQ>.RUNLIB.LOAD
DB2SampleDBRM Library		DB2SMPDL	The library where the DSNTEP2 DBRM is located, for example, <DB2InstHLQ>.DBRMLIB.DATA
SystemGroupPrefix		DB2SGPRE	DB2 system group object qualifier. This is used to prefix all DB2 objects created for this *SystemGroup*.
DataStorageGroup Name		DB2STGNW	DB2 storage group name where your MQSeries Workflow for z/OS data will be stored.
DataStorageGroup DataSetPrefix		DB2STGPW	DB2 storage group data set prefix for runtime data.
DataStorageGroup VolumeSet		DB2STGVW	DB2 storage group volume set for runtime data. Specify the volume name or '*' for SMS managed volumes.
AuditStorageGroup Name		DB2STGNA	DB2 storage group name for the audit trail data.
AuditStorageGroup DataSetPrefix		DB2STGPA	DB2 storage group data set prefix for audit trail data.
AuditStorageGroup VolumeSet		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 *DataStorageGroupVolumeSet*.
WorkflowDatabaseName		DB2DBNAM	DB2 Workflow database name.

Table 2. System group identifiers (continued)

Parameter	Your value	Name in customization parameter file	Description
PESMapping DatabaseName		DB2MDBNM	DB2 program execution server (PES) mapping database name.
PESDirectory DatabaseName		DB2PDBNM	DB2 PES directory database name.
WorkflowCollection		DB2DBCOL	DB2 Workflow database collection name.
PESMappingCollection		DB2MDCOL	DB2 PES mapping database collection name.
PESDirectoryCollection		DB2PDCOL	DB2 PES directory database collection name.
DB2Plan		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 57.

Table 3. System scope identifiers

Parameter	Your value	Name in customization parameter file	Description
UniqueSystemKey		MQWFUKEY	Unique key for an MQSeries Workflow for z/OS 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 your PROCLIB. This key is used in the START command to start an administration server on the *System* associated with this key.
SystemName		MQWFSYSN	MQSeries Workflow for z/OS system name. This name may be up to eight uppercase characters long, and 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 *UniqueSystemKey.AdminServerID*.
MQWFConfiguration Key		MQWFCFGK	This key must be unique for all MQSeries Workflow for z/OS 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.
ServerUserID		STTSKUID	The server started task user ID used by all MQSeries Workflow for z/OS 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 z/OS program invocations. This user ID requires EXECUTE rights on *DB2Plan*
ServerGroupID		STTSKGRP	The server started task RACF group ID for all MQSeries Workflow for z/OS servers.
CTComponent		CTRCNAME	CTRACE component name.
TraceStart		TRCWPRC	Trace writer start procedure name.
TraceStop		TRCSPRC	Trace writer stop procedure name.
ARMRestartPolicy		ARMPOLNM	The name of the ARM restart policy.

Table 3. System scope identifiers (continued)

Parameter	Your value	Name in customization parameter file	Description
ARMRestartElement NameSuffix		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*Application EnvironmentName*		WLMAE**EXE**	WLM Application Environment for the MQSeries Workflow for z/OS **execution server**. It must be unique in the parallel sysplex environment.
PES*Application EnvironmentName*		WLMAE**PES**	WLM Application Environment for the MQSeries Workflow for z/OS **program execution server**. It must be unique in the parallel sysplex environment.
ClusterNamelist		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.
QueueManager		MQQMNAME	Name of the MQSeries queue manager that is to be used by MQSeries Workflow for z/OS. 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 z/OS 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.
OS/390System TCP/IPAddress		STCPADDR	This is the TCP/IP address of the OS/390 system.
QueueManager TCP/IPPort		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.

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
CICSFlag	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 191.
CICSInstHLQ	*	CICSLPFX	CICS installation high level qualifier.
DB2InstHLQ		DB2INHLQ	DB2 installation high level qualifier.
MQInstHLQ		MQPREFIX	MQSeries installation high level qualifier.
LEInstHLQ		LELIBPFX	Language Environment installation high level qualifier.
CCPPInstHLQ		CLIBRPFX	C/C++ installation high level qualifier.
COBOLInstHLQ	*	CBLIBPFX	COBOL installation high level qualifier
IMSInstHLQ	*	IMSLIBPX	IMS(TM) installation high level qualifier.
ICONVInstHLQ		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 (*LEInstHLQ*). The value is substituted in the environment variable file, see "Appendix I. Environment variable files" on page 207. In the following example, *ICONVInstHLQ* should be set to SYS1: SYS1.SCEEUCS2 SYS1.SCEEUCS2.UCMAP SYS1.SCEEUCS2.UCONVTBL
IPCSInstHLQ		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
DB2SubSystem		DB2SSYSN	Name of the DB2 subsystem that is to be used by MQSeries Workflow for z/OS.
CICSGroup		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
DB2AdminUserID		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 GRANT SYSADM TO *DB2AdminUserID*
MQWFAdminUserID		The user ID of the MQSeries Workflow for z/OS administrator. This user ID requires EXECUTE rights on *DB2Plan* to be able to execute the tools that update the runtime databases.
applid		This value is required during "Customize CICS EXCI invocation" on page 42 and "Customize program execution server directory" on page 49.
netid		This value is required during "Customize IMS CPIC invocation" on page 46 and "Customize program execution server directory" on page 49.
luname		This value is required during "Customize IMS CPIC invocation" on page 46 and "Customize program execution server directory" on page 49.
CICSBridge InputQueue		This value is required during "Customize MQSeries CICS bridge invocation" on page 44 and "Customize program execution server directory" on page 49.
IMSBridge InputQueue		This value is required during "Customize MQSeries IMS bridge invocation" on page 48 and "Customize program execution server directory" on page 49.
XCFGroupName		This value is required during "Customize MQSeries IMS bridge invocation" on page 48. The MQSeries instance and the target IMS system must belong to the same XCF group.
XCFMemberIMS		This value is required during "Customize MQSeries IMS bridge invocation" on page 48. It represents the IMS system as a member in the XCF group *XCFGroupName*.
XCFMemberMQ		This value is required during "Customize MQSeries IMS bridge invocation" on page 48. It represents the MQSeries instance as a member in the XCF group *XCFGroupName*.

Table 6. Customization identifiers (continued)

Parameter	Your value	Description
PESDirectory SourceFile		Your PES directory is based on the skeleton provided in *CustHLQ*.SFMCDATA(FMCHEDTP). 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 49.
IspfLAN		The language code for ISPF, for example ENU for US English.
IspfHLQ		The HLQ for the ISPF library.

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 z/OS.

This is a planning phase. You should not modify any of the files mentioned here, copies of these files will be generated during "Before starting 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
Workflow DatabaseName	*InstHLQ*.SFMCDB2 (FMCHDD**DB**)	*InstHLQ*.SFMCDB2 (FMCHDD**TS**)	*InstHLQ*.SFMCDB2 (FMCHDD**TB**)
PESDirectory DatabaseName	*InstHLQ*.SFMCDB2 (FMCHDD**PD**)	*InstHLQ*.SFMCDB2 (FMCHDD**PS**)	*InstHLQ*.SFMCDB2 (FMCHDD**PT**)
PESMapping DatabaseName	*InstHLQ*.SFMCDB2 (FMCHDD**MD**)	*InstHLQ*.SFMCDB2 (FMCHDD**MS**)	*InstHLQ*.SFMCDB2 (FMCHDD**MT**)

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	40							•				
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.

Customization

Chapter 3. Before starting customization

Before starting customization, you should check the following:

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

   Note: These are installation-defined library concatenations, common names for these libraries are `SYS1.PROCLIB`, `SYS1.PARMLIB`, `USER.PROCLIB`, and `USER.PARMLIB`.
6. The load libraries *InstHLQ*`.SFMCLINK` and *InstHLQ*`.SFMCLPA` 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 "After installing MQSeries Workflow for z/OS" followed by
9. "Before starting customization" on page 23.

After installing MQSeries Workflow for z/OS

After performing the installation as described in *MQSeries Workflow for z/OS: Program Directory*, you are ready to update the MVS Message Services (MMS) message catalog, and add the library *InstHLQ*`.SFMCLPA` to the Link Pack Area (LPA) library concatenation.

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

Create the MMS message catalogs

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

Customization

Table 10. Create MMS message catalogs

Step number	Required or optional	Description	Action	Verification
1	Required	Create the VSAM clusters, and load the input files *InstHLQ*.SFMCMSG (FMCHMxxx) into them.	1. Copy the job *InstHLQ*.SFMCCNTL(FMCHJMM1) to a private partitioned data set. 2. Edit your copy of FMCHJMM1, and make the changes described in the comment header of the job. 3. Submit your copy of FMCHJMM1	rc = 0
2	Required	Customize the MMS PARMLIB member	1. Copy *InstHLQ*.SFMCPARM(FMCHYMMS) to a private partitioned data set. 2. Edit your copy of FMCHYMMS, and replace <MQWFIHLQ> with your MQSeries Workflow for z/OS installation high level qualifier. **Note:** If you are creating a new system, use your value from Table 1 on page 9, if you are migrating from a previous release, use your value from Table 106 on page 230).	
3	Required	Copy MMS PARMLIB member.	Being careful not to overwrite an existing PARMLIB member: Copy your copy of FMCHYMMS to your system PARMLIB.	
4	Required if you are creating a **new** system	Rename the MMS PARMLIB member.	Being careful not to overwrite an existing PARMLIB member, rename the MMS PARMLIB member that you copied in step 3 to MMSLST*xx*.	
5	Required if you are migrating an **existing** system	Replace the MMS PARMLIB member.	Replace the existing MMS PARMLIB member MMSLST*xx* with the one that you copied in step 3.	
6	Required	Provide RACF authorization.	Give the system address space MMS read access to the VSAM data sets created in step 1.	
7	Required	Activate the MMS PARMLIB member.	Either: • Specify MMS(*xx*) on the INIT statement in the active console parmlib member CONSOL*nn*, or • Issue the operator command SET MMS=*xx*	

LPA library concatenation

To enable the TRACE system address space to access the MQSeries Workflow for z/OS component trace start/stop exit routine FMCHXTRC correctly, you must perform the following action:

Table 11. Copy LPALIB member

Step number	Optional or required	Description	Action
1	Required	Add MQSeries Workflow for z/OS LPA library to your LPA library concatenation list.	Concatenate the MQSeries Workflow for z/OS LPA library *InstHLQ*.SFMCLPA 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 z/OS functional within your system. This procedure consists of the following stages:

- "Before starting customization" is required.
- "System Group Customization" on page 24 is required.
- "System customization" on page 29 is required.
- "Verify Workflow client sample application" on page 40 is optional.
- "Program execution customization" on page 42 is optional.

Before starting customization

Each time that you want to create a new MQSeries Workflow for z/OS system, you must perform a customization. Before starting customization, you must perform the following precustomization tasks. 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 *InstHLQ*.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 z/OS customization high level qualifier, see *CustHLQ* 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: 1. *CustHLQ*.SFMCCNTL 2. *CustHLQ*.SFMCDATA 3. *CustHLQ*.SFMCDB2 4. *CustHLQ*.SFMCMQS 5. *CustHLQ*.SFMCPARM 6. *CustHLQ*.SFMCPROC 7. *CustHLQ*.GENPROC 8. *CustHLQ*.GENPARM **Note:** The last two libraries are for the generated PROCLIB and PARMLIB members.

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.	1. Copy the JCL *InstHLQ*.SFMCCNTL(FMCHJCCT) 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.
2	Required	Edit the customization parameter file.	1. Copy the customization parameter template member *CustHLQ*.SFMCDATA(FMCHECIF) to a private partitioned data set. 2. Edit your copy of FMCHECIF, 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. **Note:** This file is described in "Customization parameter file for a primary system" on page 191. 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.	
3	Required	Generate all the JCLs necessary to customize this product.	1. Copy the JCL *CustHLQ*.SFMCCNTL(FMCHJCUS) to a private partitioned data set. 2. Edit your copy of FMCHJCUS as described in the comment header. 3. Submit your copy of FMCHJCUS. 4. Check the job output (step IKJEFT01 / DD statement SYSTSPRT) for error messages. The program performs some syntax checking on the length and value of the variables you specified in your copy of the file FMCHECIF. 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 *CustHLQ*.GENPROC and *CustHLQ*.GENPARM.	

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 Group Customization

To prepare the database resources for the new system group, 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. "Populate the Workflow database" on page 26
4. "Program execution server directory DB2 customization" on page 27

5. "Populate the PES directory database" on page 28
6. "Program execution server mapping DB2 customization" on page 28

General DB2 customization

Before performing this customization you should ensure that you have DB2 SYSADM authority. This can be granted by a person having system administration authority 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 only if you are using DB2 for OS/390 Version 5	Build DSNTEP2	If you are using DB2 for OS/390 Version 5, you must build the sample application DSNTEP2 using the PL/I compiler.	
2	Required	Bind the plan for the DB2 sample application DSNTEP2.	Submit JCL *CustHLQ*.SFMCCNTL(FMCHJBTE)	rc=0
3	Optional	If you want to change the buffer pool names and sizes:	Edit *CustHLQ*.SFMCDB2(FMCHDDBP), and change the buffer pool definitions.	
	Required	Define buffer pools.	Submit JCL *CustHLQ*.SFMCCNTL(FMCHJDBP)	rc=0
4	Optional	If you want the storage groups to use more than one volume name, or SMS managed volumes:	Edit *CustHLQ*.SFMCDB2(FMCHDDST), and change the VOLUMES parameter as necessary.	
	Required	Create storage groups.	Submit JCL *CustHLQ*.SFMCCNTL(FMCHJDST)	rc=0
5	Required	Be sure that RRS is active.	If RRS is not active, you can activate it by issuing the command: START RRS	

Workflow DB2 customization

If you do not want to change any of the options described in the table below, you can use the fastpath job which combines all the required steps.

To use the fastpath customization job, submit *CustHLQ*.SFMCCNTL(FMCHJ0CW), and verify the return code as described in the job prolog.

If you do not use the fastpath customization to create the Workflow database, you must perform the following steps:

Customization

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 *CustHLQ*.SFMCDB2(FMCHDDDB), and change the buffer pool name and storage group.	
	Required	Create Workflow database.	Submit JCL *CustHLQ*.SFMCCNTL(FMCHJDDB)	rc=0
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 *CustHLQ*.SFMCDB2(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 *CustHLQ*.SFMCCNTL(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 *Step number 1: Create Workflow database*.
3	Optional	If you want to change the buffer pools, or the value for the primary space allocation for the indexes:	Edit *CustHLQ*.SFMCDB2(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 *CustHLQ*.SFMCCNTL(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 *Step number 2: Create Workflow table spaces*.
4	Required	Bind the Workflow packages and add the Workflow Collection to the Workflow plan.	Submit JCL *CustHLQ*.SFMCCNTL(FMCHJBDB)	rc=4 can be accepted.

Populate the Workflow database

To populate and verify the Workflow database, you must perform the steps in Table 16.

Table 16. Populate the Workflow database

Step number	Required or optional	Description	Action	Verification
1	Required	Populate the Workflow database with initial settings.	Submit JCL *CustHLQ*.SFMCCNTL(FMCHJRBS)	rc=0.

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, or programs that are invoked by an installation-provided PES invocation exit.

If you do not want to perform any of the optional changes listed in Table 17, you can use the fastpath job which combines all the steps. To use the fastpath job, submit *CustHLQ*.SFMCCNTL(FMCHJ0CD), and verify the return code as described in the job prolog.

If you do not use the fastpath job, you must perform the steps in Table 17.

Table 17. Program execution server 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 *CustHLQ*.SFMCDB2(FMCHDDPD), and change the buffer pool names. You can also change the storage group.	
	Required	Create the PES directory database.	Submit JCL *CustHLQ*.SFMCCNTL(FMCHJDPD)	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 *CustHLQ*.SFMCDB2(FMCHDDPS), 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 *CustHLQ*.SFMCCNTL(FMCHJDPS)	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 *CustHLQ*.SFMCDB2(FMCHDDPT), 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 *CustHLQ*.SFMCCNTL(FMCHJDPT)	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*.
4	Required	Bind the PES directory packages and add the PES directory collection to the Workflow plan.	Submit JCL *CustHLQ*.SFMCCNTL(FMCHJBPD)	rc=0

Populate the PES directory database

You must populate the program execution server (PES) directory with the initial definitions by performing the step in Table 18.

Table 18. Populate the PES directory

Step number	Required or optional	Description	Action	Verification
1	Required	Import the PES directory template.	Submit JCL *CustHLQ*.SFMCCNTL(FMCHJPIB) **Note:** For subsequent executions of this step, use FMCHJPIC	rc=0

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.

If you do not want to make any of the optional changes in Table 19, you can use the fastpath job, which combines all the required steps. To use the fastpath job, submit *CustHLQ*.SFMCCNTL(FMCHJ0CM), and verify the return code as described in the job prolog.

If you do not use the fastpath job, you must perform the steps in Table 19.

Table 19. 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 *CustHLQ*.SFMCDB2(FMCHDDMD), and change the buffer pool names.	
	Required	Create the PES mapping database.	Submit JCL *CustHLQ*.SFMCCNTL(FMCHJDMD)	rc=0
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 *CustHLQ*.SFMCDB2(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 *CustHLQ*.SFMCCNTL(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 *Step number 1: Create the PES mapping database*.

Table 19. Program execution server mapping DB2 customization (continued)

Step number	Required or optional	Description	Action	Verification
3	Optional	If you want to change the buffer pools, or the primary space allocation for the index:	Edit *CustHLQ*.SFMCDB2(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 *CustHLQ*.SFMCCNTL(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 *Step number 2: Create the PES mapping space*.
4	Required	Bind the PES mapping packages and add the PES mapping collection to the Workflow plan.	Submit JCL *CustHLQ*.SFMCCNTL(FMCHJBMA)	rc=0

System customization

To customize the primary system in the system group, you must perform the following tasks in the given sequence:

1. "MQSeries customization"
2. "Trace customization" on page 30
3. "CICS API support customization" on page 31
4. "IMS API support customization" on page 32
5. "Workflow server customization" on page 33
6. "LAN client customization" on page 33
7. "Customize Java-API support" on page 36
8. "System customization verification" on page 40

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

MQSeries customization

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

Table 20. MQSeries customization

Step number	Required or optional	Description	Action	Verification
1	Required	Define the MQSeries resources (except for program execution).	Submit JCL *CustHLQ*.SFMCCNTL(FMCHJDMQ)	rc=0

Table 20. MQSeries customization (continued)

Step number	Required or optional	Description	Action	Verification
2	Required	Define the MQSeries resources required by MQSeries Workflow for z/OS program execution.	Submit JCL *CustHLQ*.SFMCCNTL(FMCHJPMQ)	rc=0
3	Required	Isolate the trace queue for better performance	It is recommended to separate the MQSeries Page Set to improve the performance of the tracing. The trace queue name is ....TRC.LQ	

Trace customization

One Workflow Server Trace exists for each Workflow system. You must provide and specify in the JCL procedure, the trace output data sets for the external writer.

For more information about tracing, see "The MQSeries Workflow for z/OS system trace facility" on page 133.

Table 21. Trace customization

Step number	Required or optional	Description	Action
1	Required	Copy PROCLIB members	Making sure that no existing proclib members are accidentally overwritten: 1. Copy the trace writer start procedure *CustHLQ*.GENPROC(*TraceStart*) to your PROCLIB 2. Copy the trace writer stop procedure *CustHLQ*.GENPROC(*TraceStop*) to your PROCLIB where *TraceStart* and *TraceStop* are the values that you planned in Table 3 on page 11, and assigned to the variables TRCWPRC, and TRCSPRCin the file *CustHLQ*.SFMCDATA(*FMCHECIF*)
2	Required	Create the extended trace output data sets.	1. Edit data set *CustHLQ*.SFMCCNTL(FMCHJCTR): a. Set the value of TRCVOL to the volume for the trace data sets. b. Set the size of the trace data sets. The default value is approximately 350MB. 2. Submit JCL *CustHLQ*.SFMCCNTL(FMCHJCTR)
3	Required	Provide RACF profiles.	Grant the following access to the user ID that is assigned to the trace writer. 1. Update access to the trace data sets created in step 2. 2. Read access to the trace queue ....TRC.LQ **Note:** If no explicit user ID assignment is made, the trace writer runs under the user ID STCUSER and group ID STCGROUP.
4	Required	Enable IPCS to call Workflow trace programs	Make sure that the MQSeries Workflow for z/OS load library *CustHLQ*.SFMCLOAD is either in the LINKLIST concatenation, or in the STEPLIB of TSO sessions.

CICS API support customization

If you want to use the MQSeries Workflow for z/OS 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.

Table 22. CICS API support customization

Step number	Required or optional	Description	Action	Verification
1	Required if you are creating a new system (skip this step if you are migrating an existing system)	Enable LE and C/C++ features in CICS.	If they are not already enabled: 1. Enable LE in CICS. **Note:** The CSD definitions necessary to accomplish this task are located in *LEInstHLQ*.SCEESAMP(CEECCSD) 2. Enable the C/C++ feature in CICS. **Note:** A sample that may help you with this task is located in *CCPPInstHLQ*.SCLBSAM(CLB3YCSD) where *LEInstHLQ* and *CCPPInstHLQ* are your values from Table 4 on page 13.	
2	Required	Specify the location of the Workflow executables, and start-up parameters.	1. Edit your CICS start-up job. 2. Find the DFHRPL entry. 3. Add the MQSeries Workflow for z/OS library called *InstHLQ*.SFMCLOAD 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 EDSALIM=200M and REGION=220M in your CICS start-up job.	
3	Required	Create user profile, machine profile, and environment data in VSAM format.	1. Edit *CustHLQ*.SFMCCNTL(FMCHJCPR) 2. Change the CICSVOL value to the name of the volume where you want the profiles to be located. 3. Submit JCL *CustHLQ*.SFMCCNTL(FMCHJCPR)	rc=0
4	Optional	If you do not want to use the value for *CICSGroup* that you specified in Table 5 on page 14:	Change the group in the CSD file: 1. Edit the CSD file*CustHLQ*.SFMCDATA(FMCHEPRO) 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 z/OS.	1. Edit *CustHLQ*.SFMCCNTL(FMCHJCUP) 2. Change the CICSNAME value to the name of the CICS system that you are customizing. 3. Submit JCL *CustHLQ*.SFMCCNTL(FMCHJCUP)	rc=0
6	Required	Restart CICS.	Restart CICS.	

Table 22. CICS API support customization (continued)

Step number	Required or optional	Description	Action	Verification
7	Required if you are creating a new system (skip this step if you are migrating an existing system)	Make MQSeries CICS stubs available in CICS.	Make the MQSeries CICS Stubs `IMQB23IC` and `IMQS23IC` from *MQInstHLQ*`.SCSQLOAD` available to CICS.	

8	Required	Make the C/C++ group CLB and the MQSeries Workflow for z/OS group available in CICS.	1. Make the C/C++ group CLB available in CICS, with the command:
			`CEDA ADD G(CLB) LIST(`*xxx*`)`
			2. Make the MQSeries Workflow for z/OS group (*CICSGroup*, unless you changed it in step 4) available in CICS, with the command:
			`CEDA ADD G(`*yyy*`) LIST(`*xxx*`)`
			where *xxx* is a LIST used at CICS start-up, and *yyy* is the MQSeries Workflow for z/OS group.	

9	Required	Stop and restart CICS.	Stop and restart CICS.	
10	Required	Verify profile access.	1. Log on 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 z/OS DLLs available to IMS so that programs using the MQSeries Workflow for z/OS 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" on page 33.

Table 23. 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 *InstHLQ*`.SFMCLOAD` library to your IMS PGMLIB library.

Workflow server customization

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

Table 24. 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 *CustHLQ*.GENPROC(*UniqueSystemKey*) to your PROCLIB, where *UniqueSystemKey* is your value in Table 3 on page 11.	
2	Required	Assign user ID and group to the Workflow server started task. **Note:** This is the *ServerUserID*, see Table 3 on page 11.	Submit JCL *CustHLQ*.SFMCCNTL(FMCHJDSC)	rc=0
3	Required	Provide RACF profile.	Give the *ServerUserID* assigned in Step 2 read access to the data set *CustHLQ*.SFMCDATA.	
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.	1. Edit your PROCLIB(*UniqueSystemKey*). 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, simple trace, and language environment dump (CEEDUMP) output. 5. If you want, you can modify the sysout class for stdout, stderr, simple trace, and language environment dump (CEEDUMP) output.	
5	Required only if you want servers and tools to give MMS messages in uppercase	Modify the language setting in the configuration profile.	Edit the configuration profile *CustHLQ*.SFMCDATA(FMCHEMPR), 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, however, some of them also appear on the OS/390 system console.	
6	Required	Grant the server user ID execute access to the database plan.	Issue the command: GRANT EXECUTE ON PLAN *DB2Plan* TO *ServerUserID* using your values for *DB2Plan* in Table 2 on page 10, and *ServerUserID* 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 z/OS server. This task consists of two parts:

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

Customization

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 *Program Code* CD-ROM.

Customize the MQSeries client connection

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

Table 25. 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.
2	Required	Generate a channel table file.	1. Edit *CustHLQ*.SFMCCNTL(FMCHJCCC) and replace the <volume> parameter with the DASD name where you want to have your generated channel table 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. a. For the current Workflow system: Submit the JCL *CustHLQ*.SFMCCNTL(FMCHJCCC), and expect rc=0. b. For multiple Workflow systems: 1) Edit the JCL *CustHLQ*.SFMCCNTL(FMCHJCCC) 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 *CustHLQ*.SFMCMQS(FMCHNCCC), by the concatenation of FMCHNCCC files of multiple systems (if you have them all on shared DASD). Another way to do this is toconcatenate a file of your own to the collected client channel definitions. 2) Submit the JCL *CustHLQ*.SFMCCNTL(FMCHJCCC), 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 26. 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 *MQSeries Workflow - Program Code* CD-ROM as described in *IBM MQSeries Workflow: Installation Guide*.
2	Required	Download channel tab file.	Download the file *CustHLQ*.mqwfchl.tab 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 step.

To customize Java-API support you must do the following:

Table 27. Customize Java-API support

Step number	Required or optional	Description	Action
1	Required once per OS/390 system	Create an HFS directory and copy the unpack script to HFS	1. Edit the job *CustHLQ*.SFMCCNTL(FMCHJUS0), and replace the parameters as specified in the comment header, where: a. MQWFIHLQ is the MQSeries Workflow installation high level qualifier. b. MQWFDIR is the HFS directory for MQSeries Workflow. For example, /usr/lpp/fmc. c. DIRU is an existing HFS directory where the STDOUT and STDERR output are to be stored. 2. Submit the job *CustHLQ*.SFMCCNTL(FMCHJUS0).
2	Required	Unpack the Java API package into HFS	1. Edit the job *CustHLQ*.SFMCCNTL(FMCHJUS1), and replace the parameters as specified in the comment header, where: a. MQWFIHLQ is the MQSeries Workflow installation high level qualifier. b. MQWFDIR is the HFS directory for MQSeries Workflow. For example, /usr/lpp/fmc. c. DIRU is an existing HFS directory where the STDOUT and STDERR output are to be stored. d. USSPACK = **JAVAAPI** is the name of the package to be installed. 2. Submit the job *CustHLQ*.SFMCCNTL(FMCHJUS1).

3	Required	Create two external links in your HFS Java directory.	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:
			`ln -e FMCH1LOC libfmcojloc.so`
			`ln -e FMCH1PRF libfmcojprf.so`

4	Required	Set the environment variables.	1. In your HFS .profile file: 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 z/OS environment variables file *CustHLQ*.SFMCDATA(FMCHEENV). 2. Run your profile with the command: `. .profile`
5	Required	Decide whether to call JavaVM via LNKLST or STEPLIB definitions.	If you want to call JavaVM via a LNKLST then perform step 6a, otherwise perform step 6b.

Table 27. Customize Java-API support (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.	Add data sets to the linklist LNKLST:
			1. for the JNI-Layer data sets:
			a. *InstHLQ*.SFMCLOAD(FMCOJLOC)
			b. *InstHLQ*.SFMCLOAD(FMCOJPRF)
			2. and for the data sets where MQSeries is located:
			a. *MQInstHLQ*.SCSQLOAD
			b. *MQInstHLQ*.SCSQAUTH
			where *InstHLQ* and *MQInstHLQ* are your values from "Installation scope identifiers" on page 9 and "Flags and high level qualifiers" on page 13 respectively.
			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:
			`LNKLST DEFINE NAME(FMCJNI1) COPYFROM(CURRENT)`
			`LNKLST ADD NAME(FMCJNI1) DSNAME(`*InstHLQ*`.SFMCLOAD)`
			`LNKLST ADD NAME(FMCJNI1) DSNAME(`*MQInstHLQ*`.SCSQLOAD)`
			`LNKLST ADD NAME(FMCJNI1) DSNAME(`*MQInstHLQ*`.SCSQAUTH)`
			`LNKLST ACTIVE NAME(FMCJNI1)`
			then activate this member with the system console command:
			`/set prog=<xy>`

6b		Add JNI-Layer and MQSeries data sets to STEPLIB.	Set the environment variable STEPLIB to the JNI-Layer and MQSeries data sets using the command: `EXPORT STEPLIB=`*InstHLQ*`.SFMCLOAD:`*MQInstHLQ*`.SCSQLOAD:` *MQInstHLQ*`.SCSQAUTH` where *InstHLQ* and *MQInstHLQ* are your values from "Installation scope identifiers" on page 9 and "Flags and high level qualifiers" on page 13 respectively.
7	Required	Copy the user and configuration profiles to enable Workflow log on.	1. Copy and rename the profile from *CustHLQ*.SFMCDATA(FMCHE**C**PR) to the HFS directory from where you call the Java program. This profile must be renamed to FMCHE**M**PR: 2. Copy the profile *CustHLQ*.SFMCDATA(FMCHE**U**PR) to the HFS directory from where you call the Java program. This profile must not be renamed.

Now you have customized the Java-API support.

Customize the XML message API and distributed process sample using XML

This customization step is optional, if you do not do it now, but realize later that you want to use the XML API or distributed process sample using XML, you can perform this customization later:

Customization

Table 28. Customize the XML message API and distributed process sample using XML

Step number	Required or optional	Description	Action
1	Required	Enable XML message API	In order to use the XML message API of MQSeries Workflow for z/OS, you must install the following: 1. XML Toolkit for OS/390, V1.1 2. XML Parser for OS/390, C++ Edition You can download them from http://www.s390.ibm.com/xml/ or order it from IBM.
2	Required once per OS/390 system	Create an HFS directory and copy the unpack script to HFS.	If you have not already customized the Java-API support: 1. Edit the job *CustHLQ*.SFMCCNTL(FMCHJUS0), and replace the parameters as specified in the comment header, where: a. MQWFIHLQ is the MQSeries Workflow installation high level qualifier. b. MQWFDIR is the HFS directory for MQSeries Workflow. For example, /usr/lpp/fmc. c. DIRU is an existing HFS directory where the STDOUT and STDERR output are to be stored. 2. Submit the job *CustHLQ*.SFMCCNTL(FMCHJUS0).
3	Required	Unpack the package into HFS.	1. Edit the job *CustHLQ*.SFMCCNTL(FMCHJUS1), and replace the parameters as specified in the comment header. **Note:** If you have already customized this job for Java-API support, only the following change will be required: a. USSPACK = **DPXML** identifies the package to be installed. 2. Submit the job *CustHLQ*.SFMCCNTL(FMCHJUS1).
4	Required	Use the sample.	For more information about the distributed process sample using XML, see the file /usr/lpp/fmc/smp/DPXML/readme.txt, and follow the instructions detailed there.

Now you have customized the distributed process sample.

Customize the Web Client

This customization step is optional, if you do not do it now, but realize later that you want to use the Web Client, you can perform this customization later. To customize the Web Client sample you must do the following:

Table 29. Customize the Web Client

Step number	Required or optional	Description	Action
1	Required once per OS/390 system	Create an HFS directory and copy the unpack script to HFS.	If you have not already customized the Java-API support or distributed process sample using XML: 1. Edit the job *CustHLQ*.SFMCCNTL(FMCHJUS0), and replace the parameters as specified in the comment header, where: a. MQWFIHLQ is the MQSeries Workflow installation high level qualifier. b. MQWFDIR is the HFS directory for MQSeries Workflow. For example, /usr/lpp/fmc. c. DIRU is an existing HFS directory where the STDOUT and STDERR output are to be stored. 2. Submit the job *CustHLQ*.SFMCCNTL(FMCHJUS0).

Table 29. Customize the Web Client (continued)

Step number	Required or optional	Description	Action
2	Required	Unpack the Web Client package into HFS.	1. Edit the job *CustHLQ*.SFMCCNTL(FMCHJUS1), and replace the parameters as specified in the comment header. **Note:** If you have already customized this job for Java-API support or the distributed process sample using XML, only the following change will be required: a. USSPACK = **WEBCL** identifies the package to be installed. 2. Submit the job *CustHLQ*.SFMCCNTL(FMCHJUS1).
3	Required	Use the Web Client.	Use a Web browser to open the file *MQWFDIR*/doc/WebClient/index.html, and follow the instructions given there.

Now you have customized the Web Client.

System customization verification

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

Table 30. 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:
			`START UniqueSystemKey.AdminServerID`
			`MODIFY AdminServerID,START EXE`
			
			where *UniqueSystemKey* is your value specified in Table 3 on page 11, and *AdminServerID* is a made-up name used to identify the administration server for the system identified by *UniqueSystemKey*.
				You should get the system console messages
				`FMC19027I Administration server successfully started.`
				
				and
				`FMC10200I Execution server for system SystemName 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	Log on.	Log on using the user ID ADMIN, and the password "password".	If no error message is displayed, the verification is complete.
4	Required	Log off.	Log off the runtime client.	

5	Required only if you are **not** migrating an existing system	Stop the OS/390 execution and administration servers.	On the OS/390 system console, issue the command:
			`MODIFY AdminServerID,STOP EXE`
			`MODIFY AdminServerID,STOP ADM`
			
			where *AdminServerID* 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 42.

This uses a sample application to verify that the workstation client can work with MQSeries Workflow for z/OS 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 31. 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 `START UniqueSystemKey.AdminServerID` where *UniqueSystemKey* is your value specified in Table 3 on page 11, and *AdminServerID* is a made-up name used to identify the administration server.	You should get the system console messages `FMC19027I Administration server successfully started.`
2	Required	Import the Workflow sample process model.	Submit JCL *CustHLQ*`.SFMCCNTL(FMCHJFDL)`	`rc=0`
3	Required	Start the OS/390 execution server.	On the OS/390 system console, issue the command: `MODIFY AdminServerID,START EXE` where *AdminServerID* is the name you used in step 1, when starting the server.	You should get the system console message: `FMC10200I Execution server for system SystemName 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	Log on.	Log on 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 *IBM MQSeries Workflow: Getting Started with Runtime*.	
7	Required	Start a new work item.	1. Start the process instance. 2. Refresh the work item window.	An editor should be displayed. A new work item named `Edit_Activity` should appear in the work item window.
8	Required	Terminate the work item.	1. Close the editor. 2. Refresh the instance and work item windows.	The work item named `Edit_Activity` is finished, and the instance has terminated.
9	Required	Log off.	Log off the runtime client.	
10	Required	Stop the execution server.	On the OS/390 system console, issue the command: `MODIFY AdminServerID,STOP EXE`	
11	Required	Stop the OS/390 administration server.	On the OS/390 system console, issue the command: `MODIFY AdminServerID,STOP ADM`	

Program execution customization

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

Table 32. 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 44
IMS	CPIC	"Customize IMS CPIC invocation" on page 46
	MQSeries IMS Bridge	"Customize MQSeries IMS bridge invocation" on page 48
User-defined	User-defined	Depends on your implementation.

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 49
2. "Configure program execution samples" on page 52
3. "Verify program execution samples" on page 52

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 33. Customize CICS EXCI invocation

Step number	Required or optional	Description	Action	Verification
1	Required	Define group resources CONNECTION and SESSIONS.	Decide or identify the name of the group (*GroupName*) that will contain the resource definition *ConnectionName* for the connection and sessions. In that group: 1. Create a generic EXCI CONNECTION resource definition with the following values: a. ACCESSMETHOD=IRC b. PROTOCOL=EXCI c. CONNTYPE=GENERIC d. 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. e. 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. 2. Define a SESSIONS resource definition with the following values a. CONNECTION=*ConnectionName* b. PROTOCOL=EXCI c. RECEIVEPFX=RC d. 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.	1. Identify the user ID of the PES (see *ServerUserID* in Table 3 on page 11), the ID of the of the CICS server region (see *applid* 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. 2. Define RACF FACILITY class profile DFHAPPL.*applid* with universal access NONE 3. Give the *ServerUserID* and all users READ access to the RACF FACILITY class profile DFHAPPL.*applid*. 4. Give *ServerUserID* 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 user ID *userid* of the MQSeries Workflow user making the request. It is therefore necessary to give these *userid*s 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 userid 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 *ServerUserID* 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 34. Customize MQSeries CICS bridge invocation

Step number	Required or optional	Description	Action
1	Required	Prepare CICS to run the CICS bridge.	1. Make sure that the MQSeries CICS adapter is set up and customized on CICS. For more information see *MQSeries for OS/390 Version 2.1 System Management Guide — MQSeries CICS adapter*. 2. Define CICS bridge transactions and programs by running the resource definition utility `DFHCSDUP` on the CICS system using the sample *MQInstHLQ*`.SCSQPROC(CSQ4CKBC)`. 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.	1. Define a local MQSeries queue *CICSBridgeInputQueue* (see your value in Table 6 on page 14) with attributes: a. `SHARE` b. `MSGDLVSQ(FIFO)` c. `DEFPSIST(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 user IDs for which program execution requests to the CICS bridge should be issued.	1. For each MQSeries Workflow user ID `mqwf_uid` define a profile named *`mqwf_uid`*`.DFHSTART` in the RACF `SURROGATE` class without any general access rights using the RACF command:
			`RDEFINE SURROGAT` *`mqwf_uid`*`.DFHSTART UACC(NONE) OWNER(`*`mqwf_uid`*`)`
			2. Give `READ` access to `surrogate_id` to all of the above define profiles by issuing for each `mqwf_uid` using the RACF command:
			`PERMIT` *`mqwf_uid`*`.DFHSTART CLASS(SURROGAT) ID(`*`surrogate_id`*`) ACCESS(READ)`
			where
			surrogate_id
			Name of the user ID to be defined as surrogate user ID of all 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.
			mqwf_uid
			User IDs of all MQSeries Workflow users to be allowed to run CICS bridge invocations.
			Note: Set the CICS startup parameter `XUSER=YES` to enable surrogate user checking.

Table 34. 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.	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 user IDs for which CICS bridge request should be issued (user IDs corresponding to MQSeries Workflow user IDs). 2. Give `WRITE` access to the CICS bridge request queue to the *ServerUserID* (see your value in Table 3 on page 11). 3. Give `READ` access to the CICS bridge reply to queue(s) to the *ServerUserID*. 4. Give `WRITE` access to each reply queue to those 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 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 user IDs of MQSeries Workflow users.	Define RACF authority access to CICS programs to user IDs of MQSeries Workflow users.	1. Identify the user IDs *mqwf_userid* 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 *mqwf_userid* 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.	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.

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 user IDs but no passwords should be checked.	1. Start the CICS bridge with the CICS command:
			`CKBR Q=`*`InputQueue`* `AUTH=LOCAL`
			or
			`CKBR Q=`*`InputQueue`* `AUTH=IDENTIFY`
			where *InputQueue* is your value for *CICSBridgeInputQueue* in Table 6 on page 14.
			2. See *MQSeries for OS/390 System Management Guide - Starting the CICS bridge* 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.
			Note: Since MQSeries Workflow for z/OS does not support passwords the authority levels, `VERIFY_UOW` and `VERYFY_ALL` are not supported by CICS bridge invocations.

Customization

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 35. 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.	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: 1. If the APPCPM*xx* 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 APPCPM*xx* 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):	Make sure that there is a PARMLIB member ASCHPMxx available defining the scheduling characteristics of the ASCH, as described below: 1. If there is already an ASCHPMxx member in your 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 *OS/390 MVS Planning: APPC/MVS Management*, "Defining Scheduling Characteristics with ASCHPMxx." There are no CPIC invocation specific definitions needed in ASCHPMxx.
3	Required	Define an APPC LU associated with the target IMS system (service).	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. 1. Put LUADD statement to APPCPMxx PARMLIB member with following parameters: a. ACBNAME(<*ims_lu*>) — where *ims_lu* is the name of the APPC LU to be associated with the target IMS. b. BASE c. SCHED(<*ims_id*>) — where *ims_id* is the (1-4 character) ID of the target IMS as defined in the IMSCTRL installation macro. 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.

Table 35. Customize IMS CPIC invocation (continued)

Step number	Required or optional	Description	Action
4	Required	Define a system base APPC LU for VTAM(R) that is enabled for protected conversation support.	1. In a member of SYS1.VTAMLST define an APPL statement for the above defined system base LU with: a. ACBNAME=<*base_lu*> — where *base_lu* is the name of the system base LU. b. APPC=YES c. ATNLOSS=ALL d. SYNCLVL=SYNCPT e. VERIFY=NONE or VERIFY=OPTIONAL **Note:** For more information about VTAM definitions, see *VTAM Resource Definition Guide*. Now the system base LU is defined in VTAM and supports distributed syncpoint conversations.
5	Required	Define APPC LU of target IMS to VTAM enabled for protected conversation support.	1. In a member of SYS1.VTAMLST define an APPL statement for the target IMS LU defined in step 3 with: a. ACBNAME=<*ims_lu*> — where *ims_lu* 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 **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. f. If security checks are required — SECACPT=ALREADYV or SECACPT=AVPV **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. g. VERYFY=NONE or VERIFY=OPTIONAL The target IMS LU has now been defined in VTAM and supports distributed syncpoint conversations.
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 user IDs representing MQSeries Workflow users who are to be allowed to run transactions on the target IMS:	1. Prohibit general access to LU of target IMS by defining a RACF profile using the command RDEFINE APPL <*ims_lu*> UACC(NONE) — where *ims_lu* is the name of the APPC LU to be associated with the target IMS. 2. Give READ access to MQSeries Workflow for z/OS 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 **Note:** For more information about using RACF, see *OS/390 Security Server (RACF) Command Language Reference*. 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.

Table 35. 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.	1. If APPC/MVS is not running, start the APPC/MVS address space by issuing the command `START APPC,SUB=MSTR,APPC=xx` — where *xx* is the suffix in the name of the PARMLIB member `APPCPMxx` 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 `START ASCH,SUB=MSTR,ASCH=xx` where *xx* is the suffix in the name of the PARMLIB member `ASCHPMxx` containing the configuration of the ASCH. 3. If APPC/MVS is already running and the changes in `APPCPMxx` must be activated, issue the command `SET APPC=xx`. For more information, see *OS/390 MVS Planning: APPC/MVS Management*, "Starting the APPC and ASCH Address Spaces.". 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.
8	Required	Make APPC/IMS LU ready to receive inbound requests from CPIC invocations.	1. Start APPC/IMS on the target IMS by issuing the command `/START APPC` 2. If security checking is required, change the APPC/IMS security level to allow user ID checking by issuing the command `/SECURE APPC CHECK` — 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 `/SECURE APPC NONE` The target IMS is now ready to call transactions requested by inbound requests from CPIC invocations via its APPC LU.

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 36. Customize MQSeries IMS bridge invocation

Step number	Required or optional	Description	Action
1	Required	Define parameters for MQSeries.	1. Define *XCFGroupName* and *XCFMemberMQ* (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. For more information, see the *MQSeries for OS/390 System Management Guide*.

Table 36. Customize MQSeries IMS bridge invocation (continued)

Step number	Required or optional	Description	Action
2	Required	Define parameters for IMS.	In the IMS parameter list 1. Define the *XCFGroupName* using the `GRNAME` parameter. **Note:** The MQSeries instance and the target IMS system must belong to the same XCF group. 2. Define the *XCFMemberIMS* 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. For more information, see the *MQSeries for OS/390 System Management Guide*.
3	Required	Tell MQSeries the XCF group and member name of the IMS system.	1. Define an MQSeries storage class with the XCF group that MQSeries and the target IMS belong to (see *XCFGroupName* in Table 6 on page 14), and with the XCF member name of the IMS (see *XCFMemberIMS* in Table 6 on page 14).
4	Required	Define the MQSeries queue for the request messages to the IMS bridge.	1. Define a local MQSeries queue *IMSBridgeInputQueue* (see your value in Table 6 on page 14) with the storage class defined in step 3, and the attributes: 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.	So that user IDs will be checked, but no passwords are required: 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 `IMSXCF.`*XCFGroupName*`.`*XCFMemberMQ* 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.	So that user IDs will be checked, but no passwords are required: 1. Define an OTMA security level of `CHECK` by issuing the IMS command: `/SECURE OTMA CHECK` 2. Define RACF profile `IMSXCF.`*XCFGroupName*`.`*XCFMemberIMS* 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.	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.

Customization

Table 37. Customize program execution server directory

Step number	Required or optional	Description	Action	Verification
1	Required	Copy the PES directory template	Make a copy of the PES directory template *CustHLQ*.SFMCDATA(FMCHEDTP), and give it your PES directory source file name that you decided in Table 6 on page 14.	
2	Required	Add EXCI, CPIC, and MQSeries bridge connection parameters to the PES directory source file.	Edit your PES directory source file, and substitute your values for the following connection parameters for CICS and/or IMS. 1. If you intend to use CICS EXCI invocation, customize the EXCI invocation section in the following way: a. Change <*applid*> 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: a. Change <*queuename*> to the name of the MQSeries CICS bridge input queue, see your value for *CICSBridgeInputQueue* 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 QUEUEMANAGER=<*queuemanager*>; 3. If you intend to use IMS CPIC invocation, customize the CPIC invocation section in the following way: a. Change <*netid*> to the APPC network identifier of the IMS system you want to use b. Change <*luname*> 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: a. Change <*queuename*> to the name of the MQSeries IMS bridge input queue, see your value for *IMSBridgeInputQueue* 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 QUEUEMANAGER=<*queuemanager*>;	
3	Required if you want to use the CICS EXCI invocation type	Customize the connection parameters in the EXCI invocation section.	Edit your PES directory source file: 1. Find the EXCI invocation section. 2. Change <*applid*> to the application identifier of the CICS system you want to use.	
4	Required if you want to use the MQSeries CICS bridge invocation type	Customize the connection parameters in the MQSeries CICS bridge invocation section.	Edit your PES directory source file: 1. Find the MQSeries CICS bridge invocation section. 2. Change <*queuename*> to the name of the MQSeries CICS bridge input queue, see your value for *CICSBridgeInputQueue* in Table 6 on page 14. 3. If the queue manager of the CICS system belongs to the same MQSeries cluster as your Workflow system, then remove the text QUEUEMANAGER=<*queuemanager*>;	

Table 37. Customize program execution server directory (continued)

Step number	Required or optional	Description	Action	Verification
5	Required if you want to use the IMS CPIC invocation type	Customize the connection parameters in the IMS CPIC invocation section.	Edit your PES directory source file: 1. Find the CPIC invocation section. 2. Change <*netid*> to the APPC network identifier of the IMS system you want to use 3. Change <*luname*> to the LU Name of the IMS system you want to use. 4. If the mode name of your IMS LU is not `#INTER`, then change `#INTER` to your value.	
6	Required if you want to use the MQSeries IMS bridge invocation type	Customize the connection parameters in the MQSeries IMS bridge invocation section.	Edit your PES directory source file: 1. Find the MQSeries IMS bridge invocation section. 2. Change <*queuename*> to the name of the MQSeries IMS bridge input queue, see your value for *IMSBridgeInputQueue* in Table 6 on page 14. 3. If the queue manager of the IMS system belongs to the same MQSeries cluster as your Workflow system, then remove the text `QUEUEMANAGER=<queuemanager>;`	

7	Required if you want to use a user-defined invocation type	Create a new invocation section.	Edit your PES directory source file:
			1. Create a new invocation section for your invocation type:
			2. For your exit, define the following:
			`type`
			`exitName`
			`exitParameters`
			3. For each service, define the following:
			`type`
			`name`
			`connectionParameters`	

8	Required	Edit the PES directory import job.	If you are using your own version of the PES directory source file rather than the default one, edit *CustHLQ*`.SFMCCNTL(FMCHJPIC)`, and change *CustHLQ*`.SFMCDATA(FMCHEDTP)` to the PES directory source file that you want to import (see your value in Table 6 on page 14).	
9	Required	Import the PES directory.	Submit JCL *CustHLQ*`.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.

If you are creating a new system, you are ready to perform "Configure program execution samples" on page 52.

Configure program execution samples

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

Table 38. 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 *CustHLQ*.SFMCCNTL(FMCHJMPR)
2	Required	Copy sample programs.	Verify that the following files are in your IMS PGMLIB: 1. *InstHLQ*.SFMCLOAD(FMCH3ICS) 2. *InstHLQ*.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 *CustHLQ*.SFMCCNTL(FMCHJPDL)
5	Required	Enable PES to execute sample programs.	If security is active for either CICS or IMS, then you must enable the *ServerUserID* 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”.

Verify program execution samples

This is the final verification that you have configured your MQSeries Workflow for z/OS 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 39. Verify program execution samples

Step number	Required or optional	Description	Action	Verification
1	Required	Ensure that the necessary subsystems are running.	If necessary, start RRS, DB2, MQSeries *QueueManager*, CICS, or IMS.	
2	Required	Check external trace writer	Ensure that the external trace writer is in a link pack area with the command: `DISPLAY PROG,LPA`	
3	Required	Start the OS/390 administration server.	On the OS/390 system console, issue the command `START UniqueSystemKey.AdminServerID` where *UniqueSystemKey* is your value specified in Table 3 on page 11, and *AdminServerID* is a made-up name used to identify the administration server.	You should get the system console message: `FMC19027I Administration server successfully started.`
4	Required	Start the MQSeries Workflow for z/OS system.	On the OS/390 system console, issue the command: `MODIFY AdminServerID,START`	
5	Required	Start the runtime client.	Double-click on the runtime client icon.	You are prompted for the client's user ID and password.
6	Required	Log on.	Log on using the user ID `ADMIN`, and the password `"password"`. **Note:** For more information on using the runtime client, refer to *IBM MQSeries Workflow: Getting Started with Runtime*.	You will see the basic tree view with the icons labeled: • 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.
7	Required	Create new process template list.	Create process template lists for the sample processes.	You will see the sample items: 1. CICSMapping 2. CICSContainer 3. IMSMapping 4. IMSContainer
8	Required	Create process instances.	1. Create process instances for the processes you want to test. 2. Create process instance list items for the sample processes.	You will see the sample items: 1. CICSMapping 2. CICSContainer 3. IMSMapping 4. IMSContainer
9	Required	Create work items.	1. Select all process instance items and start them. 2. Click OK for every window that appears. 3. Create new worklist.	You will see work items for: 1. CICSMapping_Activity 2. CICSContainer_Activity 3. IMSMapping_Activity 4. IMSContainer_Activity

Table 39. Verify program execution samples (continued)

Step number	Required or optional	Description	Action	Verification
10	Required	Start new work items.	1. Start new work items a. CICSMapping_Activity b. CICSContainer_Activity c. IMSMapping_Activity d. IMSContainer_Activity 2. Refresh the work item list.	After successful completion, each work item returns "**Finished**".

11	Optional	Check results of **CICSMapping _Activity**.	The CEEOUT section of your CICS job should contain:
			`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`	

12	Optional	Check results of **CICSContainer _Activity**.	The CEEOUT section of your CICS job should contain:
			`FMCH2CCT: MQWF Program Execution Customization`
			`OutContainer Name = SimpleDS`
			`InContainer Name = SimpleDS`
			`Set OutputContainer long value ml rc = 0`
			`Main: Set OutputContainer rc = 0`	

13	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`	

14	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`	

15	Required	Log off.	Log off the runtime client.	

Table 39. Verify program execution samples (continued)

Step number	Required or optional	Description	Action	Verification
16	Required	Stop the MQSeries Workflow for z/OS system.	On the OS/390 system console, issue the command: `MODIFY AdminServerID,STOP`	
17	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 z/OS 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 57.
- Add extra clients as described in "Chapter 6. Adding extra Workflow clients to an existing system" on page 69.

Customization

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 59
3. "Create input files for customizing an additional system in a system group" on page 60
4. "General DB2 customization (DB2 data sharing)" on page 60
5. "Update topology setting in the Workflow database" on page 61
6. "MQSeries customization" on page 62
7. "Trace customization" on page 62
8. "CICS API support customization" on page 63
9. "IMS API support customization" on page 65
10. "Workflow server customization" on page 65
11. "LAN client customization" on page 66

Decide the new system's identifiers

The following identifiers have scope over a Workflow system.

Table 40. Identifiers required for each new system

Parameter	Your value	Name in customization parameter file	Description
CustHLQ		MQWFCHLQ	The high level qualifier for the new MQSeries Workflow for z/OS system you want to create.
UniqueSystemKey		MQWFUKEY	Unique key for an MQSeries Workflow for z/OS 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 your PROCLIB. This key is used in the START command to start an administration server on the *System* associated with this key.
SystemName		MQWFSYSN	MQSeries Workflow for z/OS 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 *UniqueSystemKey.AdminServerID*.

Table 40. Identifiers required for each new system (continued)

Parameter	Your value	Name in customization parameter file	Description
SystemIdentifier		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.
MQWFConfiguration Key		MQWFCFGK	This key must be unique for all MQSeries Workflow for z/OS 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.
ServerUserID		STTSKUID	The server started task user ID used by all MQSeries Workflow for z/OS 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 *DB2Plan*
ServerGroupID		STTSKGRP	The server started task RACF group ID for all MQSeries Workflow for z/OS servers.
CTComponent		CTRCNAME	CTRACE component name.
TraceStart		TRCWPRC	Trace writer start procedure name.
TraceStop		TRCSPRC	Trace writer stop procedure name.
ARMRestartPolicy		ARMPOLNM	The name of the ARM restart policy.
ARMRestartElement NameSuffix		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*Application EnvironmentName*		WLMAE**EXE**	WLM Application Environment for the MQSeries Workflow for z/OS **execution server**. It must be unique in the parallel sysplex environment.
PES*Application EnvironmentName*		WLMAE**PES**	WLM Application Environment for the MQSeries Workflow for z/OS **program execution server**. It must be unique in the parallel sysplex environment.
ClusterNamelist		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 40. Identifiers required for each new system (continued)

Parameter	Your value	Name in customization parameter file	Description
QueueManager		MQQMNAME	Name of the MQSeries queue manager that is to be used by MQSeries Workflow for z/OS. 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 z/OS 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.
OS/390System TCP/IPAddress		STCPADDR	This is the TCP/IP address of the OS/390 system.
QueueManagerTCP/IP Port		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.
DB2SubsystemName		DB2SSYSN	Name of the DB2 subsystem that is to be used by MQSeries Workflow for z/OS.

Data set allocation

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

Table 41. Data set allocation

Step number	Required or optional	Description	Action	Verification
1	Required	Copy allocation job.	Copy the JCL *InstHLQ*.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 z/OS customization high level qualifier, see *CustHLQ* in Table 40 on page 57).	
3	Required	Allocate customization data sets.	Submit your copy of FMCHJACD.	rc=0 indicates that the following libraries have been created: 1. *CustHLQ*.SFMCCNTL 2. *CustHLQ*.SFMCDATA 3. *CustHLQ*.SFMCDB2 4. *CustHLQ*.SFMCMQS 5. *CustHLQ*.SFMCPARM 6. *CustHLQ*.SFMCPROC 7. *CustHLQ*.SFMCREXX **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 42. Create input files for customization

Step number	Required or optional	Description	Action	Verification
1	Required	Copy customization templates.	1. Copy the JCL *InstHLQ*.SFMCCNTL(FMCHJCCT) 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.
2	Required	Copy customization parameter file for additional systems.	Copy the file from *CustHLQ*.SFMCDATA(FMCHEC**SY**) for your primary system to new *CustHLQ*.SFMCDATA(FMCHEC**IF**). 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 40 on page 57).	
3	Required	Edit the customization parameter file.	Edit the customization parameter template member *CustHLQ*.SFMCDATA(FMCHECIF), and enter your values from Table 40 on page 57, 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 195. From now on, this member will contain your customization parameters for this new system, and from here on in this chapter, *CustHLQ* refers to the value in Table 40 on page 57. 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.	1. Copy the JCL *CustHLQ*.SFMCCNTL(FMCHJCUS) to a private partitioned data set. 2. Edit your copy of FMCHJCUS as described in the comment header. 3. Submit your copy of FMCHJCUS. 4. Check the job output (step IKJEFT01 / DD statement SYSTSPRT) for error messages. The program performs some syntax checking on the length and value of the variables you specified in the file *CustHLQ*.SFMCDATA(FMCHECIF). 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 *CustHLQ*.GENPROC and *CustHLQ*.GENPARM.	

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 61, 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 43. 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 *CustHLQ*`.SFMCDB2(FMCHDDBP)` 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 *CustHLQ*`.SFMCNTL(FMCHJDBP)`	`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 *CustHLQ*`.SFMCDB2(FMCHDDBP)`. 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 refer to *DB2 for OS/390 Data Sharing: Planning and Administration* (SC26-8961-00). **Group Buffer Pool Sizes**: For each buffer pool you are using (see *CustHLQ*`.SFMCDB2(FMCHDDBP)`) 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 *CustHLQ*`.SFMCDB2(FMCHDDGB)` 2. Change the default values to values suitable for your environment. 3. Submit JCL *CustHLQ*`.SFMCNTL(FMCHJDGB)`	`rc=0`

Update topology setting in the Workflow database

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

Table 44. 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 *CustHLQ*`.SFMCDATA(FMCHESDL)`, modifying the default definitions for the additional system where necessary.	

Table 44. 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 *CustHLQ*.SFMCCNTL(FMCHJRIS)	rc=0

MQSeries customization

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

Table 45. MQSeries customization

Step number	Required or optional	Description	Action	Verification
1	Required	Define the MQSeries resources (except for program execution.)	Submit JCL *CustHLQ*.SFMCCNTL(FMCHJDMQ)	rc=0
2	Required	Define the MQSeries resources required by MQSeries Workflow for z/OS program execution.	Submit JCL *CustHLQ*.SFMCCNTL(FMCHJPMQ)	rc=0
3	Required	Isolate the trace queue for better performance	It is recommended to separate the MQSeries Page Set to improve the performance of the tracing. The trace queue name is ....TRC.LQ	

Trace customization

One Workflow Server Trace exists for each Workflow system. You must provide and specify in the JCL procedure, the trace output data sets for the external writer.

For more information about tracing, see "The MQSeries Workflow for z/OS system trace facility" on page 133.

Table 46. Trace customization

Step number	Required or optional	Description	Action
1	Required	Copy PROCLIB members	Making sure that no existing proclib members are accidentally overwritten: 1. Copy the trace writer start procedure *CustHLQ*.GENPROC(*TraceStart*) to your PROCLIB 2. Copy the trace writer stop procedure *CustHLQ*.GENPROC(*TraceStop*) to your PROCLIB where *TraceStart* and *TraceStop* are the values that you planned in Table 3 on page 11, and assigned to the variables TRCWPRC, and TRCSPRCin the file *CustHLQ*.SFMCDATA(*FMCHECIF*) **Note:** These procedures must be defined as 'started tasks' in RACF.

Table 46. Trace customization (continued)

Step number	Required or optional	Description	Action
2	Required	Create the extended trace output data sets.	1. Edit data set *CustHLQ*.SFMCCNTL(FMCHJCTR): a. Set the value of TRCVOL to the volume for the trace data sets. b. Set the size of the trace data sets. The default value is approximately 350MB. 2. Submit JCL *CustHLQ*.SFMCCNTL(FMCHJCTR)
3	Required	Provide RACF profiles.	Grant the following access to the user ID that is assigned to the trace writer. 1. Update access to the trace data sets created in step 2. 2. Read access to the trace queue ....TRC.LQ **Note:** If no explicit user ID assignment is made, the trace writer runs under the user ID STCUSER and group ID STCGROUP.
4	Required	Enable IPCS to call Workflow trace programs	Make sure that the MQSeries Workflow for z/OS load library *CustHLQ*.SFMCLOAD is either in the LINKLIST concatenation, or in the STEPLIB of your TSO session.

CICS API support customization

If you want to use the MQSeries Workflow for z/OS 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 z/OS are members of the same cluster, you should also perform a CICS shut down.

Table 47. CICS API support customization

Step number	Required or optional	Description	Action	Verification
1	Required	Enable LE and C/C++ features in CICS.	If they are not already enabled: 1. Enable LE in CICS. **Note:** The CSD definitions necessary to accomplish this task are located in *LEInstHLQ*.SCEESAMP(CEECCSD) 2. Enable the C/C++ feature in CICS. **Note:** A sample that may help you with this task is located in *CCPPInstHLQ*.SCLBSAM(CLB3YCSD) where *LEInstHLQ* and *CCPPInstHLQ* are your values from Table 4 on page 13.	
2	Required	Specify the location of the Workflow executables, and start-up parameters.	1. Edit your CICS start-up job. 2. Find the DFHRPL entry. 3. Add the MQSeries Workflow for z/OS library called *InstHLQ*.SFMCLOAD 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 EDSALIM=200M and REGION=220M in your CICS start-up job.	

Table 47. CICS API support customization (continued)

Step number	Required or optional	Description	Action	Verification
3	Required	Create user profile, configuration profile, and environment data in VSAM format.	1. Edit *CustHLQ*.SFMCCNTL(FMCHJCPR) 2. Change the CICSVOL value to the name of the volume where you want the profiles to be located. 3. Submit JCL *CustHLQ*.SFMCCNTL(FMCHJCPR)	rc=0
4	Optional	If you do not want to use the value for *CICSGroup* that you specified in Table 5 on page 14:	Change the group in the CSD file: 1. Edit the CSD file*CustHLQ*.SFMCDATA(FMCHEPRO) 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 z/OS.	1. Edit *CustHLQ*.SFMCCNTL(FMCHJCUP) 2. Change the CICSNAME value to the name of the CICS system that you are customizing. 3. Submit JCL *CustHLQ*.SFMCCNTL(FMCHJCUP)	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 *MQInstHLQ*.SCSQLOAD available to CICS.	
8	Required	Make the C/C++ group CLB and the MQSeries Workflow for z/OS group available in CICS.	1. Make the C/C++ group CLB available in CICS, with the command: CEDA ADD G(CLB) LIST(*xxx*) 2. Make the MQSeries Workflow for z/OS group (*CICSGroup*, unless you changed it in step 4) available in CICS, with the command: CEDA ADD G(*yyy*) LIST(*xxx*) where *xxx* is a LIST used at CICS start-up, and *yyy* is the MQSeries Workflow for z/OS group.	
9	Required	Stop and restart CICS.	Stop and restart CICS.	
10	Required	Verify profile access.	1. Log on 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 z/OS DLLs available to IMS so that programs using the MQSeries Workflow for z/OS 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 48. 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 *InstHLQ*.SFMCLOAD library to your IMS PGMLIB library.

Workflow server customization

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

Table 49. 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 *CustHLQ*.GENPROC(*UniqueSystemKey*) to your PROCLIB, where *UniqueSystemKey* is your value in Table 40 on page 57.	
2	Required	Assign user ID and group to the Workflow server started task. **Note:** This is the *ServerUserID*, see Table 40 on page 57	Submit JCL *CustHLQ*.SFMCCNTL(FMCHJDSC)	rc=0
3	Required	Provide RACF profile.	Give the *ServerUserID* assigned in Step 2 read access to the data set *CustHLQ*.SFMCDATA.	
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.	1. Edit your PROCLIB(*UniqueSystemKey*). 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, simple trace, and language environment dump (CEEDUMP) output. 5. If you want, you can modify the sysout class for stdout, stderr, simple trace, and language environment dump (CEEDUMP) output.	
5	Required only if you want servers and tools to give MMS messages in uppercase	Modify the language setting in the configuration profile.	Edit the server configuration profile *CustHLQ*.SFMCDATA(FMCHEMPR), 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, however, some of them also appear on the OS/390 system console.	

Table 49. 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 DB2Plan TO ServerUserID` using your values for *DB2Plan* in Table 2 on page 10, and *ServerUserID* in Table 40 on page 57.	

LAN client customization

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

1. "Customize the MQSeries client connection"
2. "Customize the MQSeries Workflow client" on page 67

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 *Program Code* CD-ROM.

Customize the MQSeries client connection

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

Table 50. 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.	1. Edit *CustHLQ*`.SFMCCNTL(FMCHJCCC)` 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. a. For the current Workflow system: Submit the JCL *CustHLQ*`.SFMCCNTL(FMCHJCCC)`, and expect rc=0. b. For multiple Workflow systems: 1) Edit the JCL *CustHLQ*`.SFMCCNTL(FMCHJCCC)` 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 *CustHLQ*`.SFMCMQS(FMCHNCCC)`, by the concatenation of `FMCHNCCC` files of multiple systems (if you have them all on shared DASD). Another way to do this is toconcatenate a file of your own to the collected client channel definitions. 2) Submit the JCL *CustHLQ*`.SFMCCNTL(FMCHJCCC)`, 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 51. 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 *MQSeries Workflow - Program Code* CD-ROM as described in *IBM MQSeries Workflow: Installation Guide*.
2	Required	Download channel tab file.	Download the file *ClientCustHLQ*`.mqwfchl.tab` 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 40.

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 57.

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 72
- "IMS API support customization for new client" on page 74
- "LAN Client Customization" on page 74
- "Client request concentrator customization" on page 75

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 z/OS client on a queue manager, you must perform a customization. Before starting customization, you must complete the following precustomization tasks. This creates the libraries and copies files from the customization high level qualifier *CustHLQ* of an existing MQSeries Workflow for z/OS 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 70
3. "Create input files for customizing a new Client" on page 71
4. "MQSeries customization for a new client" on page 71
5. "Generate MQSeries channel tab file for LAN client" on page 72

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 52. Identifiers required for each new client

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

Table 52. Identifiers required for each new client (continued)

Parameter	Your value	Name in customization parameter file	Description
ClientConfigKey		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.
ClientQueueManager		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 z/OS 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.
OS/390 System TCP/IP address		CTCPADDR	This is the TCP/IP address of the OS/390 system where the MQSeries Queue Manager resides.
TCP/IP port of MQSeries Queue Manager		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.
ClientCICSGroup		CICSGRPC	CICS group name used for program execution server invocations.
CICSInstHLQ		CICSLPFX	CICS installation high level qualifier.

Data set allocation for client

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

Table 53. Data set allocation for client

Step number	Required or optional	Description	Action	Verification
1	Required	Copy allocation job.	Copy the JCL *CustHLQ*.SFMCCNTL(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 z/OS client customization high level qualifier, see *ClientCustHLQ* in Table 52 on page 69).	
3	Required	Allocate customization data sets.	Submit your copy of FMCHJACL.	rc=0 indicates that the following libraries have been created: 1. *ClientCustHLQ*.SFMCCNTL 2. *ClientCustHLQ*.SFMCDATA 3. *ClientCustHLQ*.SFMCMQS 4. *ClientCustHLQ*.SFMCPROC 5. *ClientCustHLQ*.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 54. Create input files for client customization

Step number	Required or optional	Description	Action	Verification
1	Required	Copy customization templates.	1. Copy the JCL *CustHLQ*.SFMCCNTL(FMCHJCLT) 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.
2	Required	Edit the customization parameter file.	Edit the customization parameter template member *CustHLQ*.SFMCDATA(FMCHECCL), and enter your values from Table 52 on page 69, 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.	1. Copy the JCL *ClientCustHLQ*.SFMCCNTL (FMCHJCCS) 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 *ClientCustHLQ*.SFMCDATA (FMCHECCL). The program substitutes your values for variables (see Table 52 on page 69) into the client customization template files.

MQSeries customization for a new client

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

Table 55. MQSeries customization for new client

Step number	Required or optional	Description	Action	Verification
1	Required	Define the MQSeries resources.	Submit JCL *ClientCustHLQ*.SFMCCNTL(FMCHJCMC)	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 56. 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.	1. Edit the job *ClientCustHLQ*.SFMCCNTL(FMCHJCCL) 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. a. For the Workflow system to which you want to add the current client: Submit the JCL *ClientCustHLQ*.SFMCCNTL(FMCHJCCL), and expect rc=0. b. For multiple Workflow systems: 1) Edit the JCL *ClientCustHLQ*.SFMCCNTL(FMCHJCCL) 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 *ClientCustHLQ*.SFMCMQS(FMCHNCCL), 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 *ClientCustHLQ*.SFMCCNTL(FMCHJCCL), and expect rc=0.	
2	Required	Download the generated tab file to your workstation.	Transfer the binary file *ClientCustHLQ*.MQWFCHL.TAB 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 z/OS 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 z/OS 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 71.
4. Perform a CICS shut down.

Table 57. 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 71.	

Table 57. 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.	If they are not already enabled: 1. Enable LE in CICS. **Note:** The CSD definitions necessary to accomplish this task are located in *LEInstHLQ*.SCEESAMP(CEECCSD) 2. Enable the C/C++ feature in CICS. **Note:** A sample that may help you with this task is located in *CCPPInstHLQ*.SCLBSAM(CLB3YCSD) where *LEInstHLQ* and *CCPPInstHLQ* are your values from Table 4 on page 13.	
2	Required	Specify the location of the Workflow executables, and start-up parameters.	1. Edit your CICS start-up job. 2. Find the DFHRPL entry. 3. Add the MQSeries Workflow for z/OS library called *InstHLQ*.SFMCLOAD 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 EDSALIM=200M and REGION=220M in your CICS start-up job.	
3	Required	Create user profile, machine profile, and environment data in VSAM format.	1. Edit *ClientCustHLQ*.SFMCCNTL(FMCHJCPT) 2. Change the CICSVOL value to the name of the volume where you want the profiles to be located. 3. Submit JCL *ClientCustHLQ*.SFMCCNTL(FMCHJCPT)	rc=0
4	Optional	If you do not want to use the value for *ClientCICSGroup* that you specified in Table 52 on page 69:	Change the group in the CSD file: 1. Edit the CSD file*CustHLQ*.SFMCDATA(FMCHEPRT) 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 z/OS.	1. Edit *ClientCustHLQ*.SFMCCNTL(FMCHJCUT) 2. Change the CICSNAME value to the name of the CICS system that you are customizing. 3. Submit JCL *ClientCustHLQ*.SFMCCNTL(FMCHJCUT)	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 *MQInstHLQ*.SCSQLOAD available to CICS.	
8	Required	Make the C/C++ group CLB and the MQSeries Workflow for z/OS group available in CICS.	1. Make the C/C++ group CLB available in CICS, with the command: CEDA ADD G(CLB) LIST(*xxx*) 2. Make the MQSeries Workflow for z/OS group (*CICSGroup*, unless you changed it in step 4) available in CICS, with the command: CEDA ADD G(*yyy*) LIST(*xxx*) where *xxx* is a LIST used at CICS start-up, and *yyy* is the MQSeries Workflow for z/OS group.	

Table 57. 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.	1. Log on 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 z/OS DLLs available to IMS so that programs using the MQSeries Workflow for z/OS container API can be executed in IMS.

Table 58. 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 *InstHLQ*`.SFMCLOAD` library to your IMS PGMLIB library.

LAN Client Customization

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

Table 59. 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 71.
2	Required	Check or decide the identifier names required for step 3.	1. Collect the following information: • *SystemGroup* from Table 2 on page 10. • *MQWFSystemPrefix* from Table 2 on page 10. • *System* from Table 3 on page 11. • *ClientQueueManager* from Table 52 on page 69. 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 60. 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 69.
2	Required	MQSeries customization.	Perform "MQSeries customization for a new client" on page 71.

Customization

Part 2. Administration

Chapter 7. Introduction to system administration

This chapter introduces you to system administration in an MQSeries Workflow for z/OS system and describes the administration server tasks.

Objects you will need to administer or use

The administration of MQSeries Workflow for z/OS 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.

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" on page 87. The administration server accepts administration commands for starting and stopping systems and servers. You can also use it to display how many of server instances are running, and also to administer hold queues. For more information, see "Chapter 8. Administration server tasks" on page 87.

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.

Notification exit
A notification exit is the executable that is called by the program execution server when it is about to invoke a program, when it has successfully invoked a program, or when an error has occurred during program invocation.

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 z/OS: Programming*. How to administrate program mapping is described in "Administering program mapping" on page 112.

Server hold queues
When a message cannot be delivered, it is placed on a hold queue. How you can display, replay, or delete these messages is described in "Hold queue commands" on page 94.

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 z/OS database using the import tool. See

"Chapter 9. Buildtime administration tasks" on page 97. For more information about Buildtime see *IBM MQSeries Workflow: Getting Started with Buildtime*.

ARM You can register the MQSeries Workflow for z/OS administration server with the OS/390 Automatic Restart Manager to ensure that the MQSeries Workflow for z/OS 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 42 and "Customize MQSeries CICS bridge invocation" on page 44.

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 z/OS 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 z/OS 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 46 and "Customize MQSeries IMS bridge invocation" on page 48.

MQSeries
MQSeries Workflow for z/OS 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 z/OS 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 z/OS system trace facility" on page 133.

Audit trail
When a process instance is executed, MQSeries Workflow writes information about each significant event into an audit trail. For more details and example queries, see "Appendix M. Audit Trail" on page 221.

Address spaces
During normal operation, you should only administer servers and systems using the administration server commands. 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 123). 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 performed by issuing commands to the administration server. Each administration server 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 MQSeries Workflow server topology 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 or more**systems** which contain 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.

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 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 workflow servers are connected to a DB2 database subsystem which may be a member of a DB2 data sharing group. System Groups that are distributed over multiple OS/390 images in a sysplex require a DB2 data sharing group

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.

Figure 5 on page 83 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.

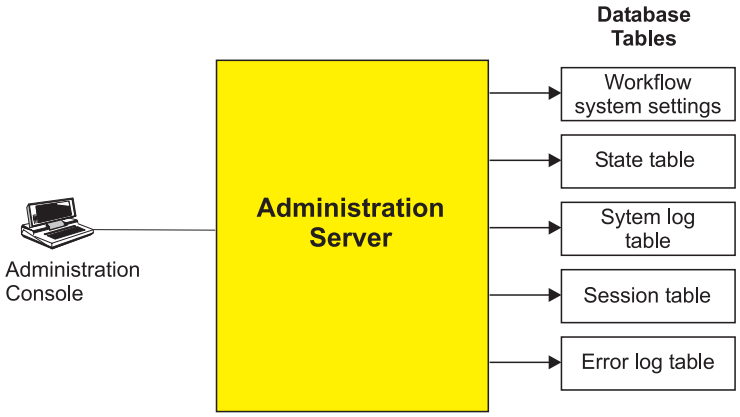


Figure 5. Implementation of the administration component in an MQSeries Workflow system

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 log on 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 other servers in 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 log on.

The administration server provides a command line interface. This allows the OS/390 administrator to start and stop MQSeries Workflow systems and servers, and to query how many server instances are running. These tasks are described in "Chapter 8. Administration server tasks" on page 87.

Overview of administration tasks

To administrate MQSeries Workflow for z/OS 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 Buildtime and console commands.

Table 61. System and server administration tasks

Task	Buildtime	System console commands	Admin server commands
"Starting the administration server" on page 87		●	
"Stopping the administration server" on page 88		●	●
"Starting the system" on page 88			●
"Stopping the system" on page 89			●
"Restarting the system" on page 89			●
"Starting servers" on page 91			●
"Stopping servers" on page 92			●
"Restarting servers" on page 92			●
"Displaying the number of instances of a server" on page 93			●
"Hold queue commands" on page 94			●
"Switching servers between WLM and non-WLM mode by importing an FDL file" on page 164		●	●
"Switching servers between WLM and non-WLM mode using Buildtime" on page 98	●	●	●
"Chapter 11. Administering Servlets on the WebSphere Application Server" on page 119		●	

System console

Before you can administer a system, you must start the administration server for that system. You do this by issuing the start server command on the system console. Then you can issue commands to the administration server.

Administration server
You start and stop systems and servers by isuing administration server commands on the system console.

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 62. Program and user administration tasks: tool dependencies

Task	Buildtime	PES directory	RACF	Program mapping
"Defining process models" on page 97	●			
"Uploading process models to the host" on page 104				
"Importing and exporting process models" on page 104				
"Enabling an OS/390 program to be run as a program activity" on page 110	●	●	●	●
"Disabling a program" on page 111	●			
"Enabling an OS/390 program to run as a safe application" on page 111	●			
"Authorizing a user to access an OS/390 program" on page 111		●	●	
"Revoking a user's access to OS/390 programs" on page 112		●	●	
"Importing a program mapping definition" on page 112				●
"Enabling a program's mapping" on page 114	●	●		
"Disabling a program's mapping" on page 115	●			

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.

Administration

Chapter 8. Administration server tasks

Most MQSeries Workflow system administration is done by issuing commands to an administration server. Before you can do this, an administration server must be started as described in "Starting the administration server".

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

- "Administration server commands"
- "System commands" on page 88
- "Server commands" on page 90
- "Hold queue commands" on page 94

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 90.

Some administration server 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 and the program execution server can be managed by WLM). For more information about using WLM, see "Part 3. Using OS/390 Workload Manager with Workflow" on page 143. You should administer the WLM managed servers using the normal administration commands as described in "Server commands" on page 90.

Administration server commands

Starting the administration server

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

Before you can issue any administration server commands, the administration server must be started. You start the administration server 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. If your `PROCLIB` is shared in a sysplex, then the value must be unique in the sysplex.

AdminServerID
: A name you will use to identify the administration server when you issue administration server 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:

`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`.

System commands

You can start, stop, and display any MQSeries Workflow for z/OS system from the system 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 89
- "Restarting the system" on page 89
- "Displaying all server instances in the system" on page 90

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 97 for more information.

To start the MQSeries Workflow for z/OS system, issue the 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 z/OS.

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 it the servers were started as described in "Starting servers" on page 91.

Stopping the system

To stop the MQSeries Workflow for z/OS system, issue the 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 88.

When the system is shut down, all session records are deleted from the MQSeries Workflow database which effectively logs off all clients (including MQSeries Workflow API programs), and all the program execution agents (PEAs) are shut down.

Note: If you do not want the sessions to be deleted when the system is shut down, you can set the profile variable *RTSystemShutdownMode* to `KeepSessions`, and enable a regular cleanup of sessions by setting a `session expiration time` and a `session expiration check interval` in the system settings (using the Buildtime tool).

Restarting the system

If you want such changes to the start parameters in the configuration profile or the environment variable file (for example, turning tracing on) to consistently 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 88
3. "Starting the administration server" on page 87

4. "Starting the system" on page 88
5. Then start extra server instances if required, see "Starting servers" on page 91.

which requires the command sequence:

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

Note: Remember that these commands are executed asynchronously, and you must wait until each command has completed before issuing the next command.

Some parameters in the configuration file can be changed without needing to restart the system, for more details see "Server configuration profile" on page 201.

Displaying all server instances in the system

To display the number of running instances of each server type in the MQSeries Workflow for z/OS system, issue the 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 93.

Server commands

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

- "Starting servers" on page 91
- "Stopping servers" on page 92
- "Restarting servers" on page 92
- "Displaying the number of instances of a server" on page 93

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 63. Server types

ServerType	Server	Instantiation type
ADM	Administration server.	Single instance
SCH	Scheduling server	Single instance
CLE	Cleanup server.	Single instance

Table 63. Server types (continued)

ServerType	Server	Instantiation type
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" on page 87.

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 165.

Starting execution servers

To activate the MQSeries Workflow for z/OS 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 z/OS server instances of a particular type, issue the 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 129.

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

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

Stopping the system by stopping the servers

The recommended way to stop the system is described in "Stopping the system" on page 89.

If you stop the system by stopping all servers (using the command `MODIFY AdminServerID,STOP ServerType`), and then shut down the administration server, the clients are not logged off, and any running program execution agents are not stopped.

If you stop the system in this way, the session records will not be removed from the database, and you should make sure that the administration server regularly cleans up the session records by setting values for `session expiration time` and `session expiration check interval` in the system settings (using Buildtime).

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 114.

- If PES directory caching is enabled as described in "Caching the PES directory at runtime" on page 110, 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 configuration 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 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 90.

Hold queue commands

Hold queue commands are used to administer undelivered messages. If an input message for a server cannot be processed, it is rolled back. If this happens enough times for the execution and program execution servers, that the retry limit is exceeded, the message is put into the server's hold queue. Message retries are a normal part of the MQSeries Workflow transaction processing, however, if several consecutive messages are put into the hold queue of a server, the server is considered to be unreliable, and it is shut down.

Note: The default retry limit is set to five, which means that MQSeries Workflow for z/OS will try five times to process a message before it puts it into the hold queue for later processing. If you want to change it, you can setting the profile variable *RTMessageRetryLimit* to a different value, for example, RTMessageRetryLimit=10.

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 server's input queue.

Note: Currently, only the execution and program execution servers offers hold queue handling, so the *ServerType* parameter in the following commands can only have the values EXE, or PES.

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 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 z/OS.

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 z/OS. 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 181.

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 100
- "Defining the connection between a program activity and the PES" on page 103

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 104.

Defining server properties

This section describes which server properties may be changed, which may not be changed, and which are ignored.

Server properties that can be changed

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

Table 64. 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 z/OS" on page 155.	Inherited (None)

Table 64. Server properties that can be changed (continued)

Property	Description	Initial default value
Context Information for External Control	Specifies the application environment name for the server type.	
User support (on the PES properties/Support modes page)	Indicates whether the PES is able to execute programs using the *ServerUserID* or the ID of the user requesting the invocation. **Agent** selects the *ServerUserID*, **Program** selects *UserID* option.	Agent
Start mode	When set to `IMMEDIATE`, the server is started automatically when the system is started. When set to `DEFERRED`, the server is not started when the system is started, and must be started manually, or by WLM.	`IMMEDIATE` when not using WLM. `DEFERRED` when using WLM mode.

Server properties that should not be changed

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

Table 65. 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 66. Server properties that are ignored on OS/390

Property	Description	Default
Start time	This setting is ignored on OS/390.	
Stop time	This setting is ignored on OS/390.	Duration = Forever
Check_Interval	Specifies how often the administration server checks whether the servers are still running. This setting is ignored on OS/390.	300 seconds

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 z/OS" on page 155, 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 164. 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:

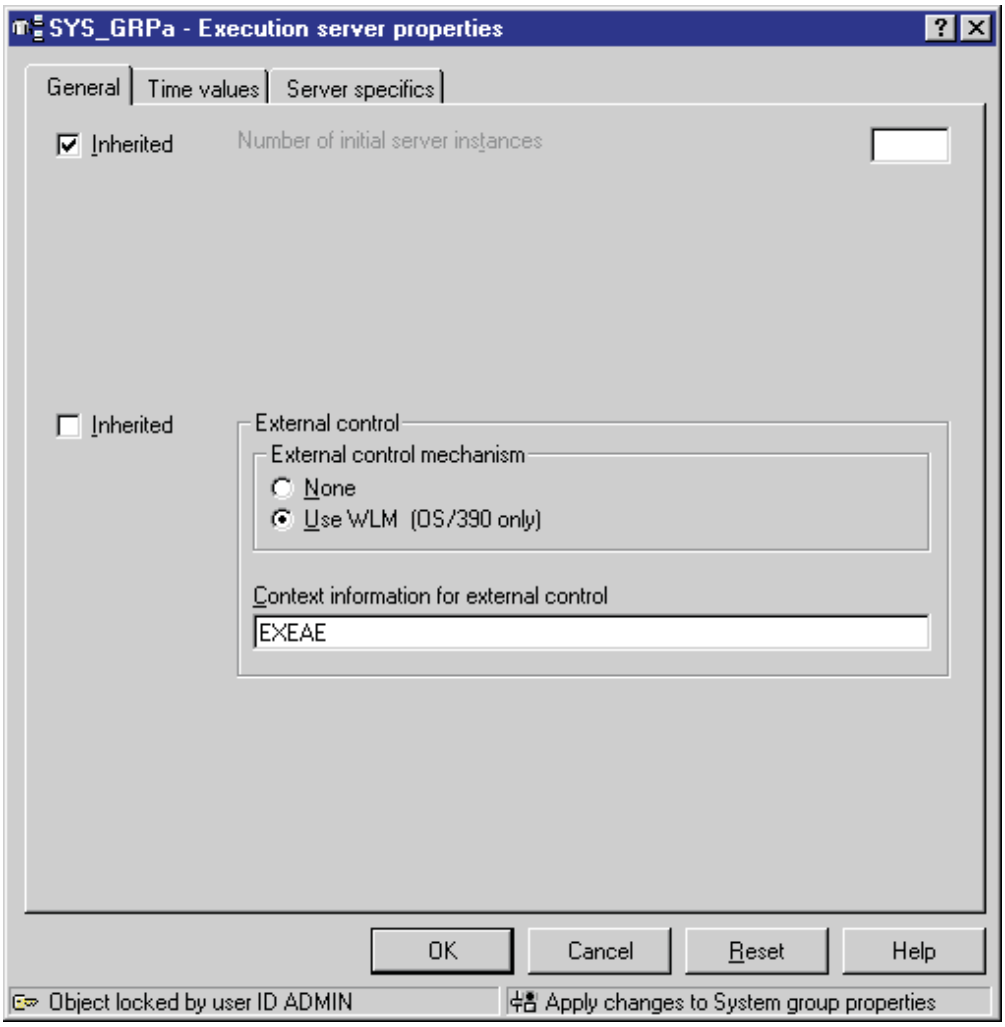


Figure 6. Execution server properties: General page

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 160.
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 screenshots 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.

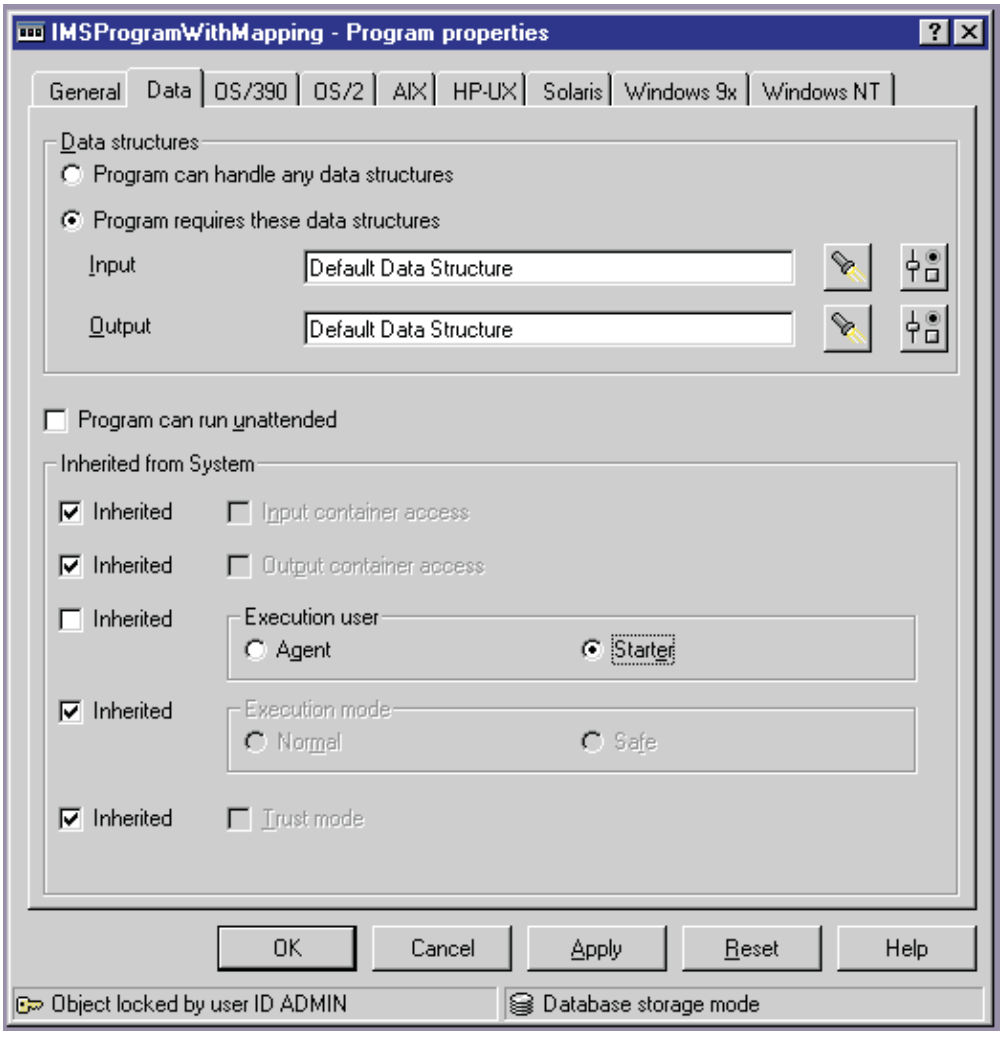


Figure 7. Program properties: Data page

Figure 7 on page 100 shows the program properties data page for 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 108.

For more information about these settings, see "Program execution security" on page 116.

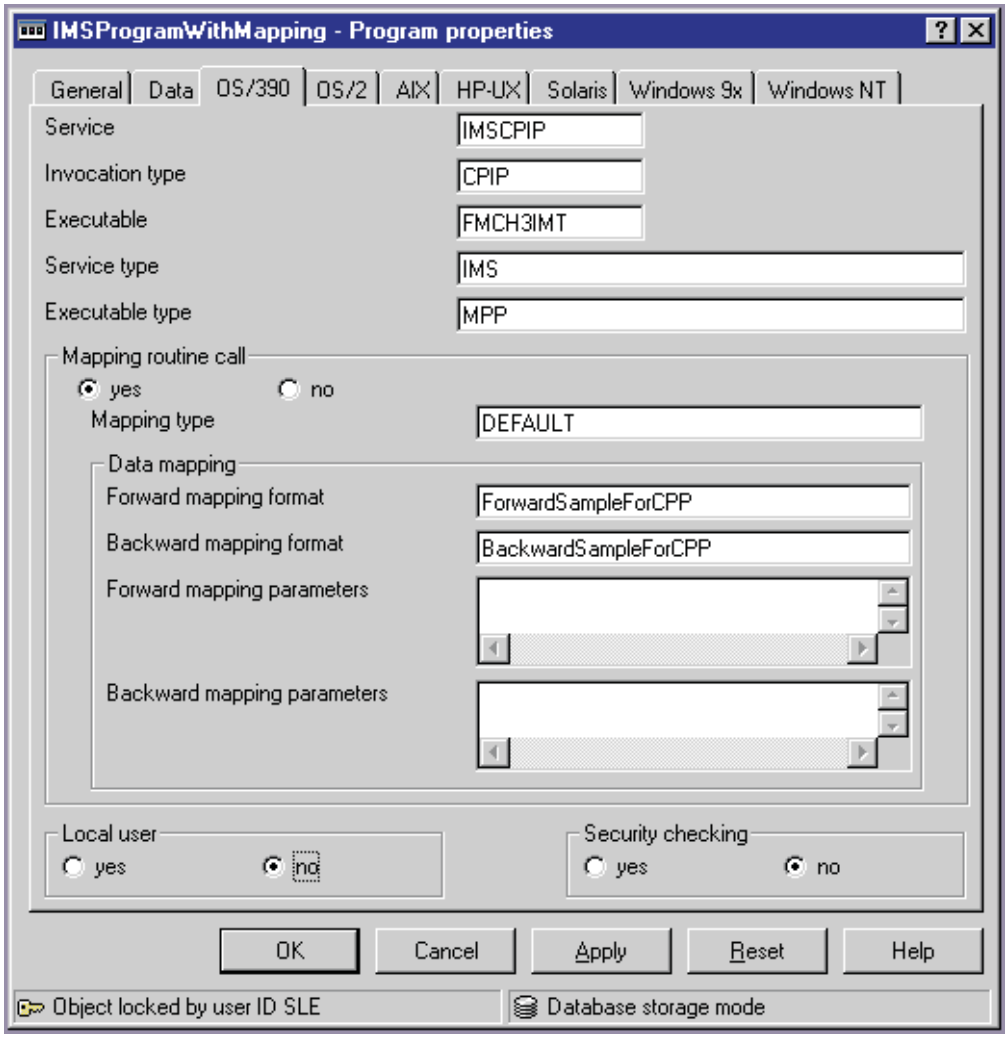


Figure 8. Program properties: OS/390 page

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

Table 67. 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.
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 *MQSeries Workflow for z/OS: Programming*.
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 *MQSeries Workflow for z/OS: Programming*.
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 *MQSeries Workflow for z/OS: Programming*.
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 *ServerUserID*. 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 defined as described in "Program security" on page 118.

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.

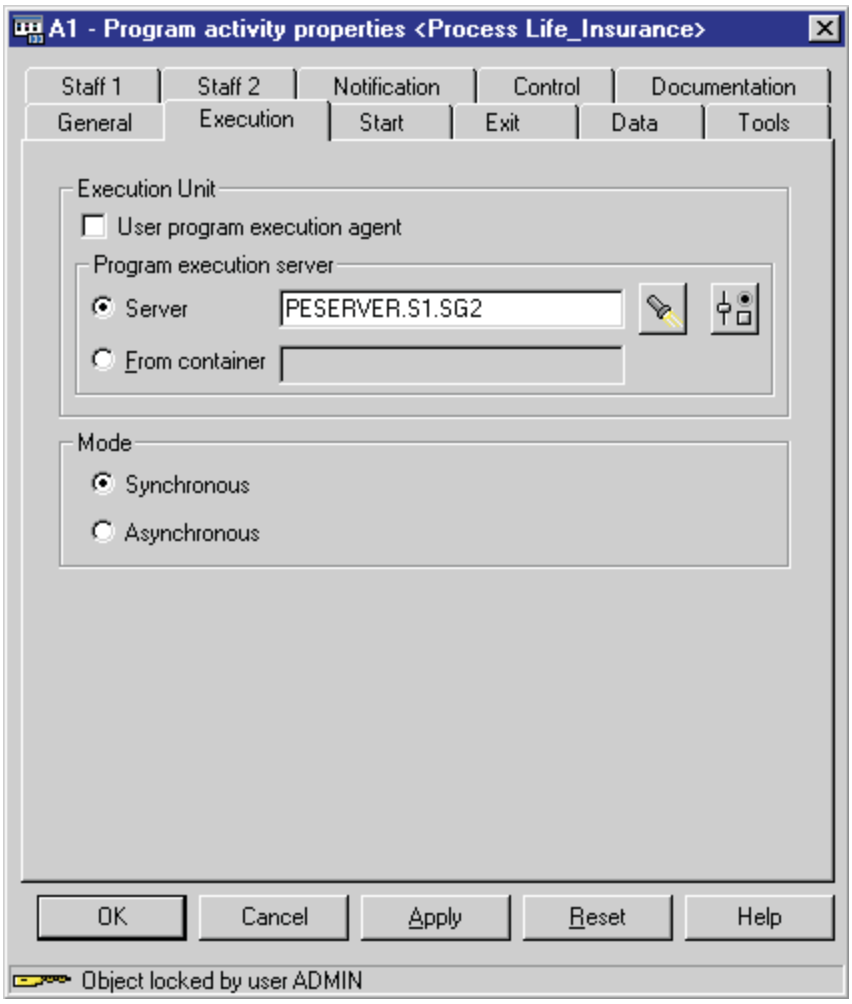


Figure 9. Program activity properties: Execution page

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 183.

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 181.

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 z/OS 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 185 and "Examples" on page 188.

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 187)
   b. Specify the input and output files using the predefined `DD`-names `FMCIIMP`, `FMCIEXP`, `FMCICMD`, and `FMCILOG`, as illustrated in "Examples" on page 188.
   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 106.

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 External CICS Interface (EXCI), APPC to IMS or the MQSeries CICS DPL and MQSeries 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 z/OS: 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 107 and "Appendix A. Program Execution Server directory" on page 171.

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 108. The PES can perform security checks to ensure that only authorized users can run a program, as described in "Program security" on page 118.

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.

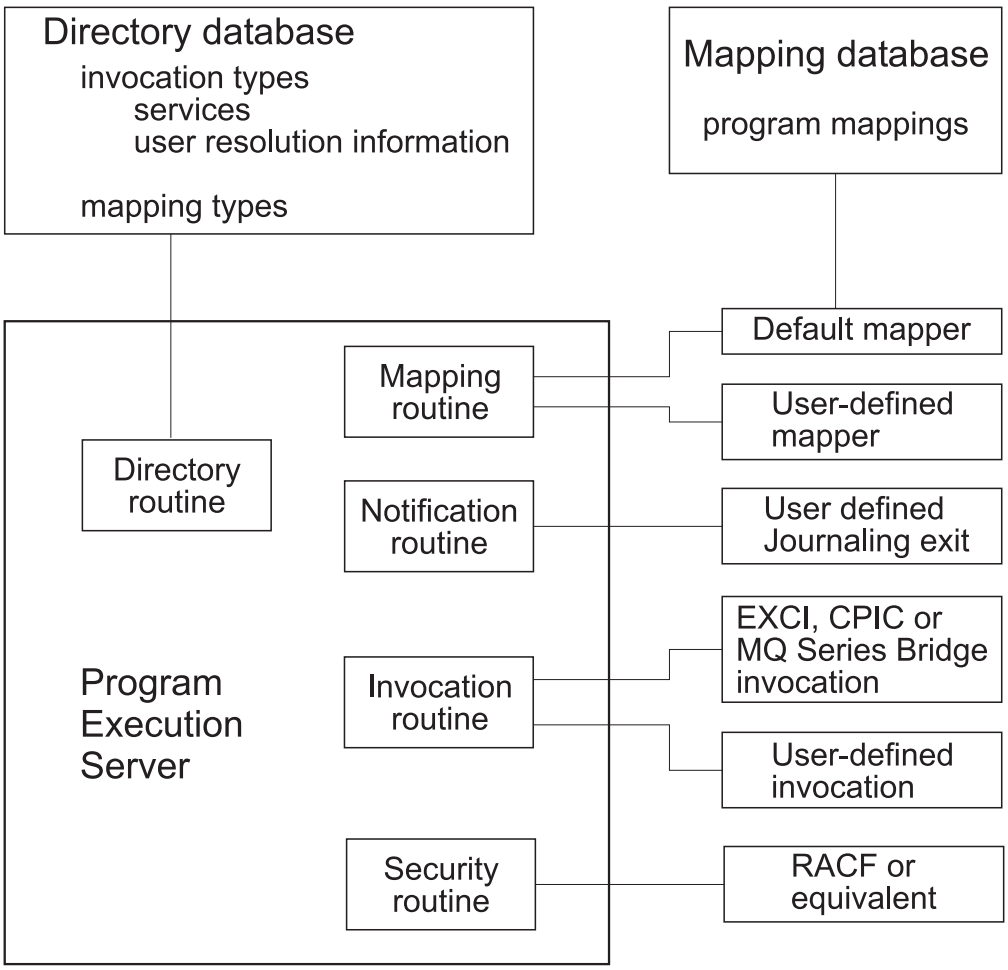


Figure 10. OS/390 Program execution server: component structure

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. Notify a 'before invocation' event.
7. For a legacy program call the mapping routine to map input container to program parameter data.
8. Call the invocation routine to handle the request.
9. For a legacy program call the mapping routine to map output parameters to the output container.
10. If none of the above step has failed, notify a 'successful invocation' event. Otherwise notify an 'invocation failure' event.
11. If none of the above step has failed, the PES returns a completion message.
12. 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 z/OS 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 z/OS: Programming*. The mapping type to be used for each program is defined in the process model as shown in Figure 8 on page 101.

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 z/OS: 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 174.

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 z/OS 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 110.

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 171.

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 109.

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 175.
- "PES directory import examples" on page 175.

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 109. For more information about the PES directory structure see "Appendix A. Program Execution Server directory" on page 171.

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 173 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.
5. To add user resolution information, substitute <*user1*> with the MQSeries Workflow user identification used to start the program and <*xxxxxxxx*> 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:

1. Copy an existing invocation section including the service and user sections.
2. 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).
3. Then update

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

4. 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 ...>
```

5. 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:

1. Copy the last existing mapping section KEYTOMAPPING<m>.
2. 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.
3. 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:

1. 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 175.
   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 configuration 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 configuration profile, see "Appendix H. Configuration profiles" on page 201.

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 111
- "Disabling a program" on page 111
- "Authorizing a user to access an OS/390 program" on page 111
- "Revoking a user's access to OS/390 programs" on page 112

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 100
2. "Creating a program mapping"
3. "Defining a security profile" on page 111

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 z/OS: 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 114.

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 100

2. "Uploading process models to the host" on page 104
3. "Importing and exporting process models" on page 104

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 118.

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 101), 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 118.
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 userID, as described in "Adding a new service definition and the related user resolution information" on page 108.
4. Reload the PES directory as described in "Importing the PES directory" on page 109.

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 z/OS 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 z/OS 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 z/OS: 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 114.

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 92.

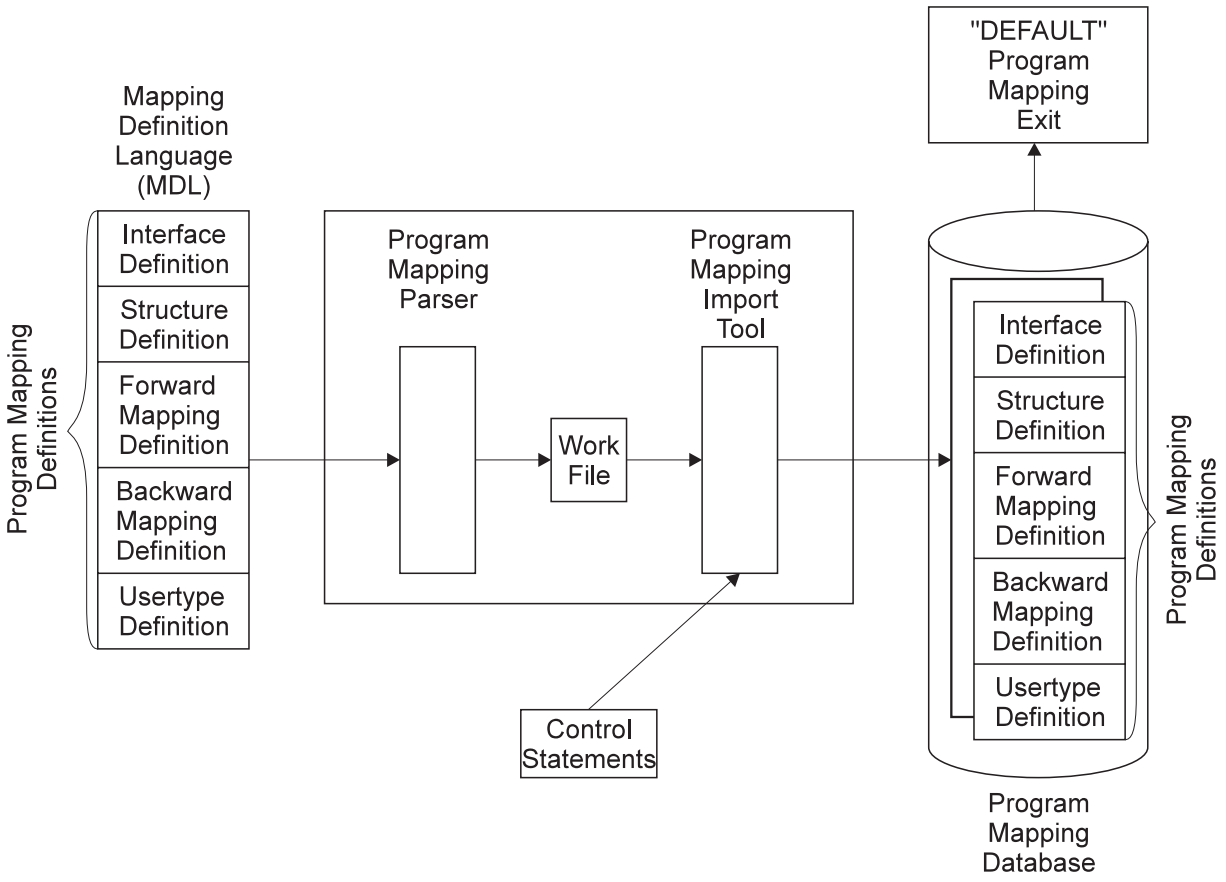


Figure 11. Program mapping definition process and components

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 z/OS: Programming*.
2. Customize a copy of *CustHLQ*.SFMCCNTL(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
//*
//FMCRMPRS EXEC PROC=FMCHPBAT,PROGRAM=FMCH1XMP,
//             PARM='ENVAR("_CEE_ENVFILE=DD:FMCHEENV")/'
//**** Mapping Definitions
//FMCIN    DD DSN=CustHLQ.SFMCDATA(FMCHEMDL),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
//FMCIN    DD DSN=*.FMCRMPRS.FMCHPBAT.FMCOUT,DISP=(SHR,DELETE)
//**** Program Mapping Import Tool Control Statements
//FMCCTL   DD DSN=CustHLQ.SFMCDATA(FMCHEMCT),DISP=SHR
//**** Listing
//FMCLST   DD SYSOUT=*
//*
```

3. Customize a copy of *CustHLQ*.SFMCDATA(SFMCEMCT) which contains the control statements for the utility so that the parsed mapping definitions are created. Update DD FMCCTL 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*.SFMCDATA(FMCHEMDL) and the sample control statements *CustHLQ*.SFMCDATA(FMCHEMCT).

Return codes

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

Table 68. 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 100:
      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 104.
3. Import the process model on the host, as described in "Using the FDL import/export tool" on page 104.

4. If mapping definitions were modified, then restart the PES as described in "Restarting the program execution server" on page 92.

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 101.
  2. Upload the new process model to the host, as described in "Uploading process models to the host" on page 104.
  3. Import the process model on the host, as described in "Using the FDL import/export tool" on page 104.

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 177.
2. Customize a copy of *CustHLQ*.SFMCCNTL(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 z/OS: Programming*.
2. Provide the mapping exit DLL to MQSeries Workflow for z/OS. 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 109.
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 101.

6. Import the new process model into the Workflow database as described in "Importing and exporting process models" on page 104.

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 101.
2. Import the changes, as described in "Importing and exporting process models" on page 104.
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 109.

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 z/OS 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 109.
2. The invocation exit DLL for MQSeries Workflow for z/OS 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 108.

Note: If there is more then 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 101.
2. Import the changes, as described in "Importing and exporting process models" on page 104.
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 109.

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 100.

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 101. 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 101.

The following combinations of settings are meaningful:

Table 69. 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**	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. If **Security checking=Yes** then a security check will be performed on the execution user ID. 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.
Program	**Agent**	If **Security checking=Yes** then a security check will be performed on the *ServerUserID*. If the security check is passed successfully (or if **Security checking=No**) then the program will be invoked using the *ServerUserID*. The setting of **Local user** has no effect.

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 117 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 108.

Program security

If a program is defined with **Security checking=Yes**, then it must have a security profile defined for the executable. The Workflow resource profile name is constructed from four qualifiers:*MQWFSystemPrefix.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 101.

The security profiles belong to the resource class `FACILITY`. They must be defined using a command like:

```
RDEFINE FACILITY workflow_resource_profile UACC(NONE)
```

where the resource profile name *workflow_resource_profile* is constructed from four qualifiers *MQWFSystemPrefix.service.invocationtype.executable*.

The installation can then use the `PERMIT` command to authorize users or groups to execute programs described by the resource profile. Generic resource profiles can be used in order to authorize the execution of a group of programs instead of individual programs.

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 Servelet 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 36.

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 120.
3. "Running a sample servlet, to log on MQSeries Workflow" on page 120.

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` be 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 configuration profiles to your WebServer's started task:

   ```
   //FMCHEUPR DD   DISP=SHR,DSN=CustHLQ.SystemGroup.System.SFMCDATA(FMCHEUPR)
   //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 36, 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.

Administration

Chapter 12. Performance tuning

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

- "Changing the number of running server instances"
- "Changing the number of server instances per address space"
- "Using the OS/390 Link Pack Area for MQSeries Workflow load libraries" on page 124
- "Caching the PES directory" on page 124
- "Tuning DB2" on page 124

Note: Some other performance issues are covered in "Response times are unacceptably long" on page 131.

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 server commands, as described in "Starting servers" on page 91. 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 configuration profile, see "Appendix H. Configuration profiles" on page 201.

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 `SRVNO`, shown in Figure 24 on page 161.

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 by the server start job to eight. If you set the number of server instances per address space (see `ExeSvrsPerAS` and `PESvrsPerAS` in "Server configuration profile" on page 201), to a number greater than eight, a dump will be written if you attempt to start more than eight instances. If you want to allow more than eight execution or program execution server instances per address

space, you must add more DD statements to the Workflow server start job as described in step 4 of “Workflow server customization” on page 33.

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 110.

Using the OS/390 Link Pack Area for MQSeries Workflow load libraries

When your installation is running with a large number of workflow servers, you should consider adding the MQSeries Workflow load libraries to the Link Pack Area (LPA) library concatenation., in order to reduce the overall storage consumption of the servers. In this case, you must add all libraries listed in the server JCL procedure's `STEPLIB` and `FMCSVLIB DD` names, and remove the libraries from the server JCL procedure. In this case, however, it is not possible to run two or more different service levels of MQSeries Workflow concurrently because the LPA library concatenation is a shared resource.

Tuning DB2

During configuration rebindings using initial data, it is possible that DB2 has chosen not to use all indices provided. After the database has reached a typical population level, it is recommended that you optimize the access paths:

1. Execute DB2 RUNSTATS by running the JCL *CustHLQ*`.SFMCCNTL(FMCHJRST)`
2. Rebind the Workflow packages by running the JCL *CustHLQ*`.SFMCCNTL(FMCHJBDB)`

Note: rc=4 can be ignored for both steps.

Chapter 13. Problem determination

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

1. If you have problems with servers or undelivered messages (on hold queues), see "Server problems" on page 126.
2. If you have problems with resources or performance, see:
   a. "Resource and performance problems" on page 131
   b. "Chapter 12. Performance tuning" on page 123
3. If you got an SVC dump, see "Using IPCS to analyze extended trace or dump output" on page 137.
4. If you have problems with WLM, see "Chapter 16. WLM problem determination" on page 167.
5. If none of the previous solutions apply, you may decide to use one of the following:
   a. "Simple trace" on page 133
   b. "Extended trace" on page 133
   c. "Simple tracing in IBM WebSphere Application Server" on page 142
   d. "Tracing in CICS" on page 142

Where to find information

The administration server is the focal point for error management within an MQSeries Workflow system. All errors and failures reported to the administration server are recorded in an error log, which can be found in the error dataset `FMCERR01` of the administration server job output. Another important concept of error reporting are exceptions. Exceptions are internal error events created by one of the components in an MQSeries Workflow system. Exceptions are mainly raised by the servers, the database layer, the message layer or by the kernel. These exceptions are also reported to the administration server and are recorded in the error log. Error logging begins when the administration server is started and ends when the administration server is shut down. It cannot be switched off.

In addition, each server instance documents errors which can not be reported to the administration server in its own error dataset. Therefore the error log contains error reports from other servers, error messages from the administration server and error information from the administration server concerning its role as server instance.

For each server instance the following error information is written to the error dataset:

- Exceptions during server initialization (they lead to an immediate shut down of the server.)
- Compact error reports, these are error reports which could not be delivered to the administration server, for an example, see "Compact error reports" on page 218.
- Other error notifications.

In addition, the error log of the administration server contains the following error information:

- Error reports from other servers.
- Error messages from the administration server.

Error log

The error log provides further information that can be used for problem determination. **Error reports** are written to the error log whenever a component within an MQSeries Workflow system encounters a problem that severely affects its operation or causes it to fail. An error report consist of a system log record and an error log record. The layout is explained in "Appendix L. Error reporting" on page 217.

If an error can not be reported to the administration server, a compact error report is written to the error data set FMCERR*xx* of the server instance concerned.

Data sets of the job output

Several data sets of the job output contain information that may help you solve some problems.

Table 70. Job output data sets

Data set	Contents/Action
FMCOUT00	Server address space startup information.
FMCOUT*xx*	The in-flight information for server instance number *xx*. **Note:** For single instance servers *xx*=01
FMCERR*xx*	Server error information, compact error reports, and error notifications such as protection exceptions. For the administration server, this may also contain error reports and error messages.
FMCTRC*xx*	Trace information.
CEEDUMP	Dump information created by Language Environment. Look at upper part showing the call stack.
FMCDMP*xx*	Dump information for each server instance *xx*. It has the same contents as CEEDUMP, but the dump was initiated by the MQSeries Workflow server itself.

Server problems

Message catalog not available

The following message is displayed on the console:

```
'+Message string not found'
```

This can be caused if the message catalog is old, not available, deleted, or corrupted. It is possible that the message with the specified ID is empty or that the message doesn't exist at all. This situation can occur, for example, if you are using an old message file that does not include new messages.

This problem may also occur when the OS/390 MVS Message Services (MMS) message file is not properly installed.

Problem starting servers

Problems starting the administration server and program execution server are described in "The administration server cannot be started" on page 128, and "The program execution server cannot be started" on page 128.

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 instances terminates immediately after starting

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

An arbitrary number of server instances terminates immediately after starting

This happens if you try to start more than one instance of a multiple instance server type, and there is insufficient virtual storage for all server instances in one server address space.

An unexpected number of server instances start

Did you wait long enough?: Starting each server instance takes some time. Issue the display command from time-to-time to check on the progress as the server instances are started:

```
MODIFY AdminServerID,DISPLAY
```

Is it managed by WLM?: If the server 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=ApplicationEnvironmentName
```

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

A dump is written before all server instances are started

No server instances started: This can be caused if the message catalog is not available. Verify that the language code `xxx` specified in the configuration profile is correct (see "Appendix H. Configuration profiles" on page 201), and that `FMCHMxxx` and `MMSLST`*`xx`* were created correctly during "Create the MMS message catalogs" on page 19.

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 by the server start job to eight. If you set the number of server instances per address space to a number greater than eight (see `ExeSvrsPerAS` and `PESvrsPerAS` in "Server configuration profile" on page 201), a dump will be written if you attempt to start more than eight instances. If you want to allow more than eight execution or program execution server instances per address space, you must add more DD statements to the Workflow server start job as described in step 4 of "Workflow server customization" on page 33.

Is the maximum 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 (see ExeSvrsPerAS and PESvrsPerAS in "Server configuration profile" on page 201). This is described in "Changing the number of server instances per address space" on page 123.

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 the database subsystem started?

The administration server requires that the DB2 subsystem is running.

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 queue ....BOOT.REQUEST
- Administration client queue ....ADC
- Administration input queue ....ADM

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 queue ....BOOT.REQUEST
- Boot reply queue ....BOOT.REPLY
- PES input queue ....PES.PESERVER

- Working queues ....PES.PESERVER.COR and ....PES.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 instances terminate

All server instances in an address space terminate

Unexpected server termination can be caused by an MQSeries or DB2 problem. Turn tracing on, as described in "The MQSeries Workflow for z/OS system trace facility" on page 133, 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, the activity goes in state error

If the error code is FMC_ERROR_RETRY_LIMIT_REACHED, then 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.

Are the server instances managed by WLM?

If the server instances are managed by WLM, then this is not a problem. It is normal for WLM to start and stop server instances according to the actual workload in the system.

Program activity stays in the 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' Recogn() interface. An internal error notification has been sent to the administration server and written to its error log. Try to correct the Recogn() interface and then restart the request.

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 167.

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 configuration profile *CustHLQ*`.SFMCDATA(FMCHEMPR)`, see "Appendix H. Configuration profiles" on page 201 for more information. After changing the configuration profile you must perform"Restarting the administration server" on page 93.

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 configuration profile are not activated

Have you restarted your servers?

Changes to the configuration profile will only become active servers have been restarted. All existing server instances will continue to use the old configuration profile settings. All new server instances will use the new configuration profile settings.

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

Changes made to the PES directory are not activated

Is PES directory caching enabled?

If PES directory caching is enabled in the configuration 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 92.

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 92.

Hold queue problems (undelivered messages)

While administering undelivered messages (as described in "Hold queue commands" on page 94), the following problems may occur:

DELETE or REPLAY affected fewer messages than expected

A message cannot be removed from the hold queue if it has not been displayed first.

DELETE or REPLAY affected the wrong messages

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.

The hold queue contains fewer messages than expected

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

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 4 on page 135.

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 91.

Are too many server instances running?

In this case restart fewer servers instances, as described in "Restarting servers" on page 92.

Is the DB2 response time too long?

During configuration rebindings using initial data, it is possible that DB2 has chosen not to use all indices provided. After the database has reached a typical population level, it is recommended that you optimize the access paths:

1. Execute DB2 RUNSTATS by running the JCL *CustHLQ*`.SFMCCNTL(FMCHJRST)`
2. Rebind the Workflow packages by running the JCL *CustHLQ*`.SFMCCNTL(FMCHJBDB)`

Note: rc=4 can be ignored for both steps.

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 latest MQSeries Workflow runtime client.

Running out of spool space

Is tracing turned on?

This problem can be caused by the trace facility. You should consider the following:

1. Check which servers have been started with trace turned on. Trace entries are written even when the servers are idle.
2. Reduce the trace level and restart the server type that was being traced.
3. If you need to write trace entries and do not want to write them synchronously to spool data sets, use the extended trace described in "The MQSeries Workflow for z/OS system trace facility" on page 133.

The MQSeries Workflow for z/OS system trace facility

Trace is used to diagnose reproducible problems by recording which statements and instructions are executed by the MQSeries Workflow for z/OS system in the sequence in which they occur. The trace facility records system events in in-storage buffers and data sets.

The system trace facility of MQSeries Workflow provides different trace modes for recording system events. Depending on the trace mode of the MQ Workflow component, the trace records are written to different places. This section describes which trace modes are available, how the different trace modes work, and how you can enable them.

The trace facility provides two types of tracing:

- "Simple trace"
- "Extended trace"

Simple trace

Simple trace writes all trace entries directly to the data set, allocated as SYSOUT in the server JCL procedure PROCLIB(*UniqueSystemKey*). Because simple trace creates synchronous input/output overheads, it is normally recommended to use the extended trace.

To use the simple trace, you must perform the following steps:

1. Edit the server configuration file *CustHLQ*.SFMCDATA(FMCHEMPR):
   a. To select simple tracing, set the value:
      FMC_SIMPLE_TRACE_ONLY=YES
   b. To get the maximum information, set the trace level to **3**:
      Configuration.*MQWFSystemPrefix*.FMC_TRACE_CRITERIA:**3**,FFFF,FFFFFFFF

      Note: Trace level 3 significantly reduces the performance of the system, and requires a large quantity of disk space. Trace level 0, gives just the most important information, and levels 2 and 1 provide intermediate details.

   Now any servers or tools that you start will produce the trace information you specified.
2. Restart the server (or tool) that is causing the problem.
3. Reproduce the problem.
4. Turn the tracing level to zero, by editing the server configuration file *CustHLQ*.SFMCDATA(FMCHEMPR), and setting
   Configuration.*MQWFSystemPrefix*.FMC_TRACE_CRITERIA:0,0000,00000000
5. Stop or restart the server that was being traced.
6. 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 FMCTRC01, FMCTRC02, ...

Extended trace

Extended trace is recommended because it does not create any synchronous input/output overhead. Extended trace is only available for servers. If you want to trace any tools, you must use the simple trace. How the extended trace works is illustrated in Figure 12 on page 137.

The MQSeries Workflow system provides multiple memory buffers. Each MQ Workflow server that has the trace facility enabled writes its trace records to these memory buffers. You can optionally externalize the memory buffers so that the buffer information is written to the input queue of the **external trace writer**. The external trace writer then writes the content of the queue to trace data sets.

The external trace writer always writes complete buffer information to a trace file in **wrap-around** mode. **Wrap-around mode** means that the external trace writer writes the buffer information to the specified trace file (also called output file) until it is full, then it switches to the next trace file. When the last trace file is full, the external trace writer starts writing to the first trace file again.

If you do not want to lose the content of a buffer during tracing, you can request to write the content of a memory buffer to an MQSeries queue before the system trace switches to the next buffer. This function is called **externalization**. To enable the externalization, you must perform the following steps:

- Start the external trace writer
- Setting the variable `FMC_EXTERNALIZE_TRACE_BUFFERS=YES`

The MQ Workflow Server writes trace entries to its internal trace buffers. If the externalization of memory buffers is enabled, the MQ Workflow Server writes the trace buffer information to the input queue of the external trace writer. The MQ Workflow server does this every time its internal trace buffer is full.

Then, the external trace writer takes the trace buffer information from the input queue and writes it to the trace data sets. For details about the trace variables, refer to "MQSeries Workflow trace variables" on page 139.

The name of the external writer's input queue is defined during the system configuration. The file names, the number, and the size of the external writer's output files can be specified by setting the appropriate environment variables.

In case of a server failure, the buffer information is written to the dump data sets, even if externalization is not switched on.

Performing an extended trace

Performing an extended trace is illustrated in Figure 12 on page 137, it requires the following actions:

1. Edit the trace variables in the server configuration file *CustHLQ*`.SFMCDATA(FMCHEMPR)`. For more information about the trace variables, see "MQSeries Workflow trace variables" on page 139.
   a. If you want to change the size of the trace buffers in each server, change the following variable, for example:

      ```
      FMC_TRACE_BUFFER_SIZE=256
      ```

      where the buffer size is in KB. Changes to this variables only take effect when a server is restarted.
   b. If you want to change the trace file size used by the trace writer, change the following variable, for example:

      ```
      FMC_TRACE_FILE_SIZE=5
      ```

      where the file size is in MB. The value specified here must not be larger than the size of the files specified in the external writer procedure (see the DD statements `DD:TRCOUT01`, `...02`, `...03`). Changes to this variable only take effect when the trace writer is restarted.

c. To select extended tracing, set the value:

```
FMC_SIMPLE_TRACE_ONLY=NO
```

d. To get the maximum information, set the trace level to **3**:

```
Configuration.MQWFSystemPrefix.FMC_TRACE_CRITERIA:3,FFFF,FFFFFFFF
```

Note: Trace level 3 significantly reduces the performance of the system, and requires a large quantity of disk space. Trace level 0, gives just the most important information, and levels 2 and 1 provide intermediate details.

e. If you want the trace information to be written to the trace data sets as each buffer becomes full, set the variable:

```
FMC_EXTERNALIZE_TRACE_BUFFERS=YES
```

Note: Even when `FMC_EXTERNALIZE_TRACE_BUFFERS` is set to `NO`, the trace buffers will be written to the dump data sets if an ABEND occurs, or if a `DUMP` command is issued.

f. If you want to be able to dynamically change the trace options during tracing, set a non-zero value that controls the trace criteria refresh rate, for example:

```
FMC_REFRESH_COUNT_FOR_TRACE_CRITERIA=50
```

If the variable `FMC_REFRESH_COUNT_FOR_TRACE_CRITERIA` is set to zero, the server only reads the trace options when it is started. If it is set to a value, *n*, that is greater than zero, the server rereads the trace options after every *n* transactions.

g. Save your changes to the server configuration file.

h. If you modified the trace buffer size in step 1a on page 134 you must restart the servers that are to be traced.

Note: The component name, *CTComponent*, specified in the server start procedure `PROCLIB(UniqueSystemKey)` will be used later for the analysis with IPCS. Where *UniqueSystemKey* is the name that you planned in Table 3 on page 11.

2. Start the trace writer by running the trace start procedure, using the command: `START TraceStart`, where *TraceStart* is the the value you planned in Table 3 on page 11.
3. Now the extended trace is taking place and you should try to reproduce the problem. You can dynamically change the trace options that you set in step 1, by editing the server configuration file. Changes only take effect after a given number of transactions, determined by the variable `FMC_REFRESH_COUNT_FOR_TRACE_CRITERIA`.
4. Turn the tracing level to zero, by editing the configuration profile *CustHLQ*`.SFMCDATA(FMCHEMPR)`, and setting

   ```
   Configuration.MQWFSystemPrefix.FMC_TRACE_CRITERIA:0,0000,00000000
   ```

5. To stop writing trace data, you can stop the external writer by running the trace stop procedure, using the command:

   ```
   START TraceStop
   ```

   where *TraceStop* is the the value you planned in Table 3 on page 11.
6. You can use IPCS to analyze the trace, as described in "Using IPCS to analyze extended trace or dump output" on page 137.

7. To convert the format of the extended trace data sets into one file with the same format as the simple trace, you must run the JCL *CustHLQ*.SFMCCNTL(FMCHJTRC).

   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(  )
   ```

   FMCHJTRC can return the following return codes:

Table 71. 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. 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 the MQSeries Workflow trace writer.

   If necessary, send the trace file to the appropriate IBM support personnel.

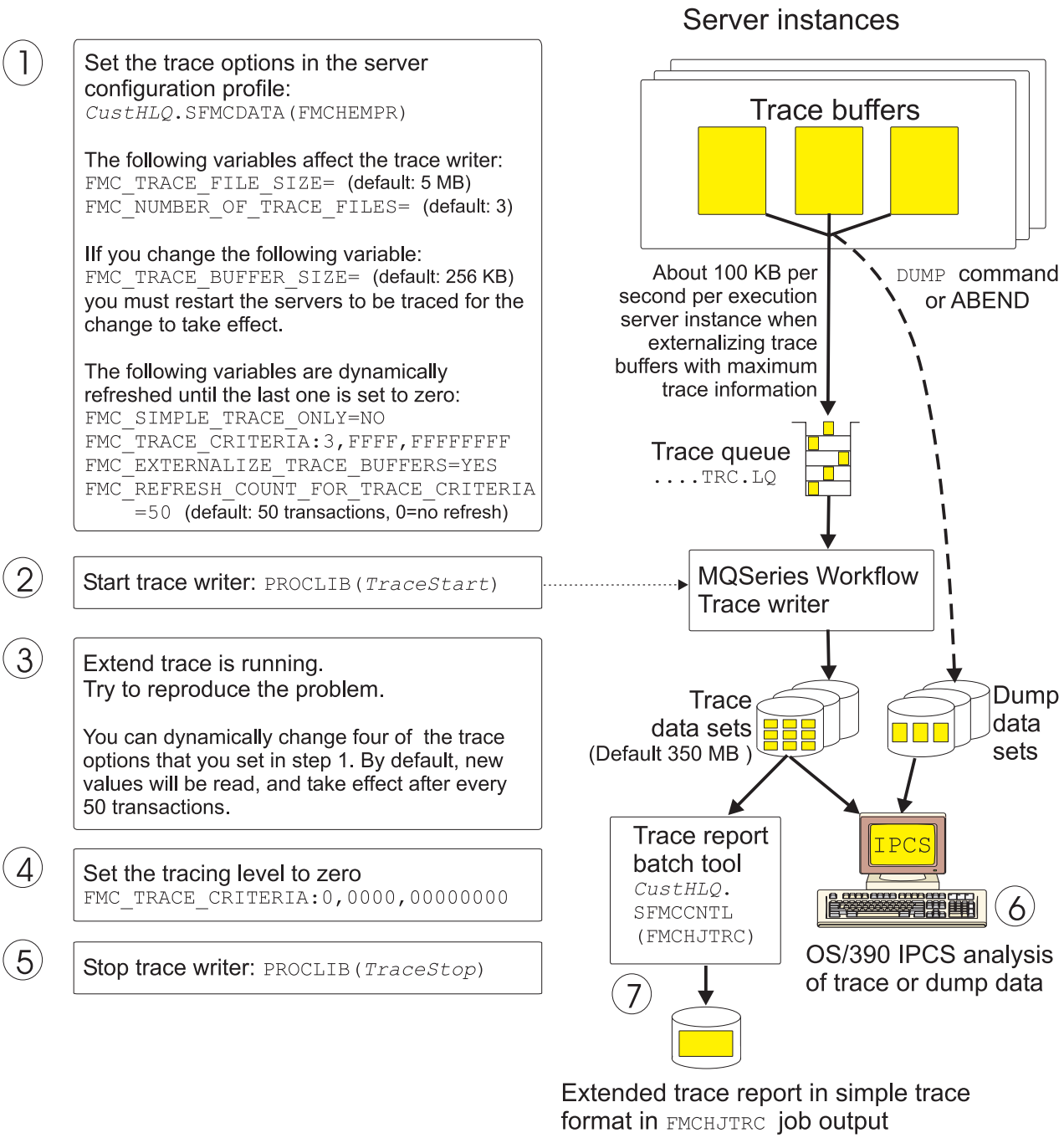


Figure 12. How extended trace works

Using IPCS to analyze extended trace or dump output

The extended trace data sets and the dump data sets can both be analyzed using IPCS.

In the TSO ISPF session:

1. Start the IPCS dialog.
2. Select option `0`.

3. Specify one of:
   a. The extended trace data set name, or a DD-name allocated to one or more of the trace data sets.
   b. The dump data set name.
4. Use the IPCS CTRACE subcommand. For example:
   ```
   CTRACE COMP(CTComponent) FULL
   ```
   where *CTComponent* is the CTRACE component name value you planned in Table 3 on page 11. It is specified in your server start procedure in PROCLIB(*UniqueSystemKey*).

Creating a problem summary from an SVC dump

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. You should submit the problem summary and job output, this will be sufficient for the analysis of most problems.

Problems with extended tracing

The most common problems with extended tracing are caused by the large quantity of trace data that is generated. For example, tracing the execution server with the maximum trace criteria level, can typically generate about 100 KB per second per instance. All other servers produce much less trace data.

1. If you get errors or warnings saying that you ran out of disk space, that trace data was lost, or that a trace buffer write failed, you can retry tracing, but changing one or more of the following:
   a. Specify a larger file size, larger buffer size, or lower trace criteria level.
   b. Allocate more disk space:
      1) Edit *CustHLQ*.SFMCCNTL(FMCHJCTR) and change the data set size. The original value is approximately 350 MB.
      2) Submit the JCL *CustHLQ*.SFMCCNTL(FMCHJCTR).

      Note: It will delete any existing data sets before creating the new ones.
2. A system ABEND B37 or D37 indicates that the trace data set is full. You can avoid this by setting the value of the trace variable FMC_TRACE_FILE_SIZE to a value that is smaller than the size of the trace data set.
3. You can increase the number of trace data sets from the default of three up to a maximum of 16. Their names must be consistently specified in all of the following files:
   a. Data set allocation job: FMCHJCTR.
   b. Customized trace writer procedure template: *TraceStart*.
   c. Trace data conversion utility: FMCHJTRC.

   You must set the variable FMC_NUMBER_OF_TRACE_FILES to reflect how many data sets are to be used for extended tracing. The trace writer will only use the number of files specified by FMC_NUMBER_OF_TRACE_FILES. After the writer has

written FMC_TRACE_FILE_SIZE KB of trace data to each of the FMC_NUMBER_OF_TRACE_FILES trace data sets, it starts to overwrite the trace information in the first trace data set.

MQSeries Workflow trace variables

You can can MQSeries Workflow control tracing by setting variables in the environment variables file. The following section provides detailed information about each variable that is listed in Table 72.

You can enable tracing by setting the appropriate variables on the machine that is to be traced. It is recommended to put the variables in the server configuation profile.You can also set these variables in the environment variable file, however it is only read when a server is started. When a process is started, it checks whether there are trace variables set in the system environment. If the process does not find any variables in the system environment, it checks the MQ Workflow configuration.

The following table lists the variables showing the default value for each variable and the mode in which you can use the respective variable.

Table 72. Variables for simple and extended tracing

Variable	Default value	Comments
FMC_TRACE_CRITERIA	Not set	Used in **simple** or in **extended** trace mode.
FMC_EXTERNALIZE_ TRACE_BUFFERS	NO	Used in **extended** trace mode. This variable must be set to ″YES″ to enable the external trace writer.
FMC_TRACE_ BUFFER_SIZE	256 KB	Used in **extended** trace mode to specify the size of the trace buffers.
FMC_TRACE_FILE_SIZE	5 MB	Used in **extended** trace mode.
FMC_REFRESH_COUNT _FOR_TRACE_CRITERIA	50 transactions	Used in **extended** trace mode, that is, for the servers only. This variable is used to control the refresh rate of the following variables: • *FMC_TRACE_CRITERIA* • *FMC_EXTERNALIZE_ TRACE_BUFFERS* • *FMC_REFRESH_COUNT_ FOR _TRACE_CRITERIA*
FMC_NUMBER_OF_TRACE_ FILES	3	Used in **extended** trace mode.

FMC_EXTERNALIZE_TRACE_BUFFERS

If the extended trace mode is active (the trace records are written to memory buffers) and you want to write the content of a full buffer to an MQSeries queue for the trace writer, you must set the variable:

```
FMC_EXTERNALIZE_TRACE_BUFFERS=YES
```

By default, this variable is set to ″NO″.

FMC_TRACE_BUFFER_SIZE

This variable specifies the size of the trace buffer in kilobytes (KB) when using the extended trace mode. If you want to change the default buffer size to 378 KB, for example, set the variable:

```
FMC_TRACE_BUFFER_SIZE=378.
```

Default size: 256 KB

Minimum size: 64 KB

Maximum size: 4 MB

FMC_TRACE_FILE_SIZE
This variable specifies the size of the trace file in megabytes (MB). It is used in by the external trace writer in the extended trace mode. If the specified file size is exceeded, the external trace writer switches to the next trace file, according to the values you have set for the trace file. The default value for the maximum trace file size is 5 MB.

FMC_REFRESH_COUNT_FOR_TRACE_CRITERIA
This variable specifies the number of transactions that the server performs before any changes of the trace criteria settings become active. The default value for this variable is 50. If you use the default value, the MQ Workflow servers read the profile settings and update any changes after every 50 transactions. If you want to disable the refresh function, set the variable to `0`.

FMC_NUMBER_OF_TRACE_FILES
This variable specifies the number of trace files that the external trace writer will attempt to write data to. The standard customization process allocates three data sets, and generates DD statements for them in the external trace writer procedure, *`TraceStart`*, and in the trace data conversion utility, `FMCHJTRC`. If you want to increase the number of trace files used, you must manually add the appropriate DD statements and allocate the necessary extra data sets.

FMC_TRACE_CRITERIA
The *FMC_TRACE_CRITERIA* variable is necessary to enable traces. It contains three parameters that define what is to be traced, and takes the form:

`FMC_TRACE_CRITERIA:`*`level`*`,`*`category`*`,`*`component`*

Level The parameter *level* defines the importance of the generated trace points. There are four levels of importance you can set:

- Level `0` traces only the most important events.
- Level `1` traces more important events.
- Level `2` is an intermediate trace level.
- Level `3` traces all important events.

Category
The parameter *category* is a bit field where each bit specifies whether a specific category should be traced or not. The bits are numbered from 0 to 15, where bit 0 is the rightmost least significant bit and bit 15 is the leftmost most significant bit. The following table describes the bits 0 to 6. These are the only bits that are currently used.

Bit	Category	Description
0	Control flow	Function entries and exits, and major conditional branches are traced.
1	Error conditions	Error conditions are traced when detected, for example, when a file is not found or the queue manager is not available.
2	System call	Traces before and after a system call is issued.
3	Messages	Traces when messages are created, sent, or received.
4	Context	Traces various context settings used to identify the flow of requests and responses through the system.

5	Performance	Performance-specific events are traced, for example, before and after DB2 is started.
6	General	All trace points that are not covered by one of the other categories are traced.

You can set any combination of these bits. For example, if you want to set the categories control flow, context, and general, you must switch on the bits 0, 4, and 6. Specifying FFFF as category ensures that all bits are switched on, inluding those that are currently not used.

Component

The parameter *component* is also a bit field. Each bit specifies whether a specific MQ Workflow internal component should issue trace entries. The bits are numbered from 0 to 31, where bit 0 is the rightmost least significant bit, and bit 31 is the leftmost most significant bit. Currently only the bits 0 to 26 are used. The following table lists the component names of each used bit.

Bit	Internal component name
0	Kernel
1	Communications
2	Messages
3	Database access
4	Tiny object manager
5	Containers
6	Staff resolution
7	Server framework
8	Administration Server
9	Execution Server
10	Modeling Server
11	Scheduling Server
12	Cleanup Server
13	Distribution Server
14	API functions
15	Administration client
16	Modeling client
17	Runtime client
18	Other clients
19	Program Execution Agent
20	Installation
21	ActiveX controls
22	Test
23	Gateway server
24	Migration
25	Cleanup audit trail
26	Configuration

As for the category, you can set any combination of these bits. Specifying FFFFFFFF as component ensures that all components issue trace entries.

Simple tracing in IBM WebSphere Application Server

If you experience problems when using the Java API in IBM WebSphere Application Server, for example, with a servlet or a Java Server Page, you may use the simple trace facility for further diagnosis. For more information about using simple trace, see "Simple trace" on page 133.

Turning tracing on

To turn tracing on, you must perform the following actions:

1. Add the following DD statement for the trace output to your WebServer's started task:

   ```
   //FMCTRC00 DD  SYSOUT=H
   ```

2. Add the following enviroment settings to your WebServer's `httpd.envvars` file:

   ```
   FMC_SIMPLE_TRACE_ONLY=YES
   FMC_TRACE_CRITERIA=TraceLevel,FFFF,FFFFFFFF
   ```

   where *TraceLevel* has the value 1, 2, or 3. Trace level 3 gives the most information.

3. Restart your WebServer.

Turning tracing off

To turn tracing off, you must perform the following actions:

1. Edit your WebServer's environment file *httpd.envvars* and change the setting of FMC_TRACE_CRITERIA as follows:

   ```
   FMC_TRACE_CRITERIA=0,0000,00000000
   ```

   Note: You may also completely remove the statement to deactivate tracing.

2. Restart your WebServer.

Tracing in CICS

All MQSeries Workflow for z/OS components running in CICS use the CICS trace facilities to generate trace entries. Trace parameters are read from the configuration and user profiles, and from the environment VSAM files that where generated during customization. The settings in the environment file overrule the configuration profile settings. The settings affect all MQSeries Workflow for z/OS 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 z/OS trace are described in "The MQSeries Workflow for z/OS system trace facility" on page 133. For more information about the CICS trace facilities see *CICS/ESA: Problem Determination Guide* and *CICS Transaction Server for OS/390: CICS Problem Determination*.

Part 3. Using OS/390 Workload Manager with Workflow

Chapter 14. Introduction to WLM

Using OS/390 Workload Manager (WLM) for the server instance management of MQSeries Workflow for z/OS is optional. You can use WLM to manage the system according to the performance goals specified in the WLM service policy. It does this by automatically adjusting address space resources and, optionally, the number of running server instances for the execution server and/or the program execution server. If you do not use WLM for workflow server instance management, you must start and stop server instances manually, using the administration commands described in "Chapter 8. Administration server tasks" on page 87.

This chapter introduces workload management concepts with emphasis on it's 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 z/OS. 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 147 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 z/OS" on page 155.

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.

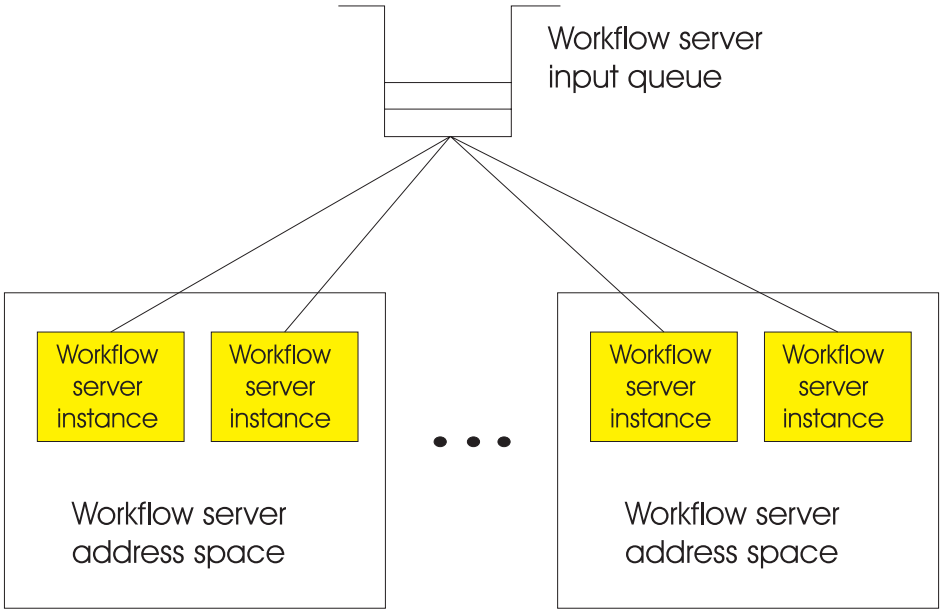


Figure 13. Server address spaces

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.

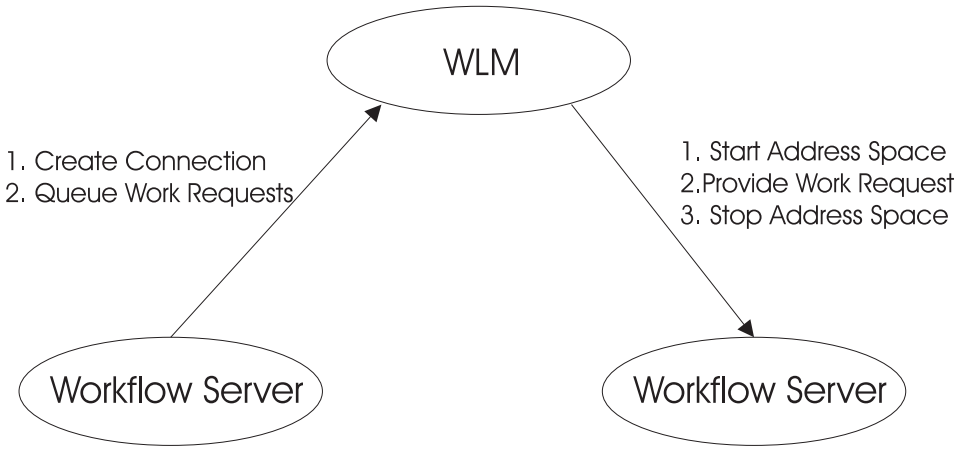


Figure 14. Interactions between MQSeries and WLM to manage execution server address spaces

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 147). 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 155.

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 z/OS. 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 209.

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 15 on page 149.

Response Time = Queue Time + Service Time.

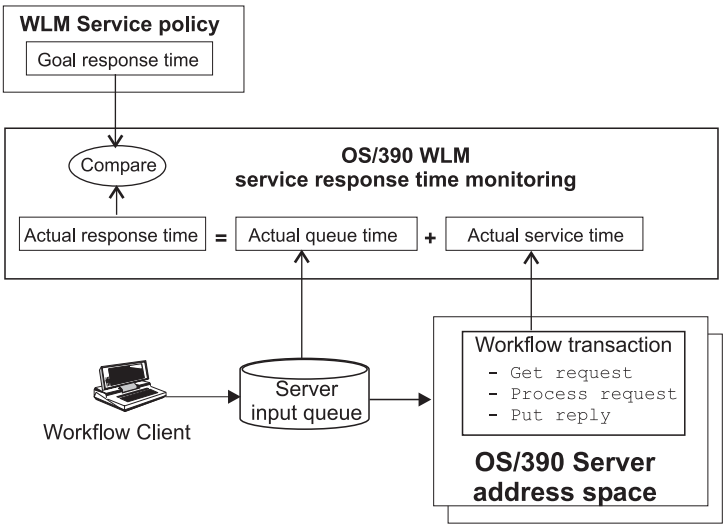


Figure 15. WLM service policy: response time goals

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. For the program execution server and execution server hotpools, the workflow administrator can still use the same administration server 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 server command, as described in "Starting the administration server" on page 87.

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 147 and "Creating a WLM service definition" on page 155). 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 server 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 server commands are described in "Chapter 8. Administration server tasks" on page 87.

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 164. 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 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 z/OS" on page 155 for more details.

You manage the application environments using the administration server 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
five or more unexpected server terminations have been encountered within ten minutes.

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 configuration profile or environment variable file can take effect. For program execution server (PES) instances, this also applies to a refresh of the cached PES directory.

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 using MQSeries Workflow administration server commands as described in "Chapter 8. Administration server tasks" on page 87.

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. The MQSeries Workflow administration server commands 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 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 16. 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.

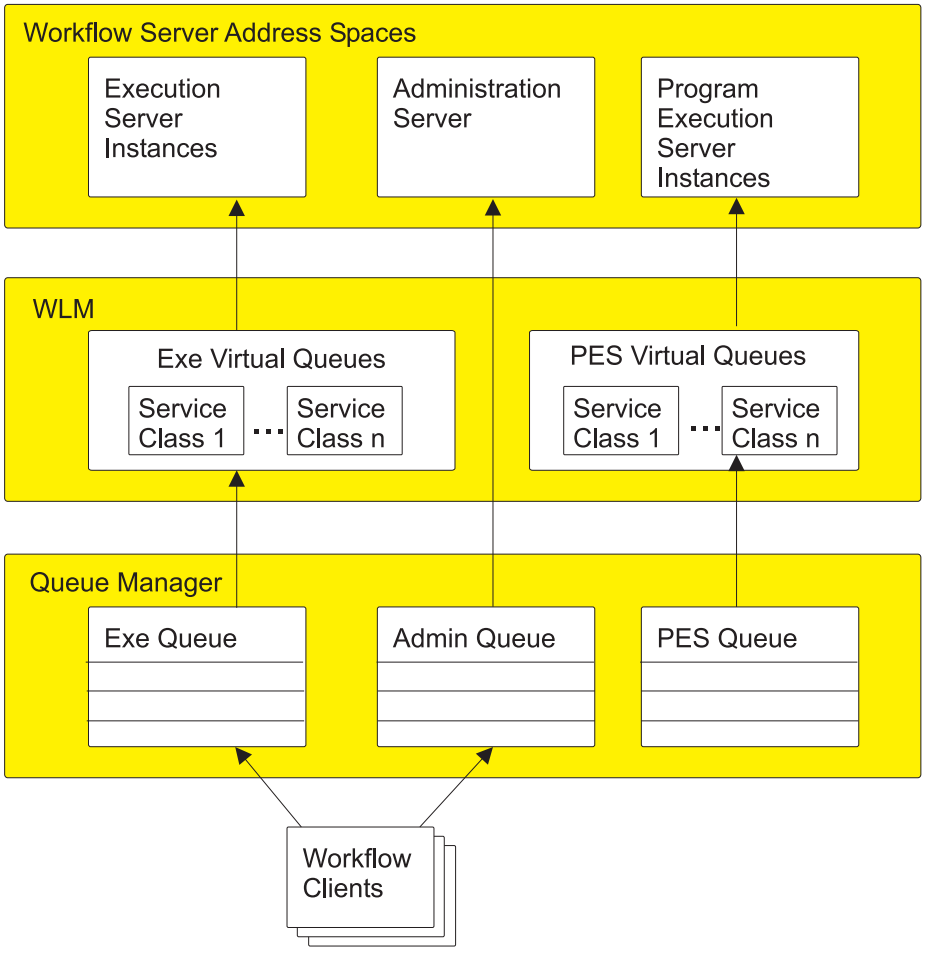


Figure 16. The WLM queuing model used by MQSeries Workflow

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 workflow server processes 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 a request message is put to a WLM-managed queue, the queue manager interprets the WIH and notifies WLM about the arrival of work which then assigns 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 z/OS.

For more information about message classification, see "Appendix J. WLM message classification" on page 209.

Chapter 15. Setting up WLM for MQSeries Workflow for z/OS

To enable WLM support in MQSeries Workflow for z/OS you need to create definitions for MQSeries for OS/390, MQSeries Workflow for z/OS and WLM itself. These definitions must be consistent with each other. In MQSeries Workflow for z/OS 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 z/OS. 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  . . . . . MQWFSEDE  (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 17. WLM panel: Definition menu

Parameter	Required or Optional	Description	Examples
Definition Name	Required	Eight character identifier for the service definition.	MQWFSEDE
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  . . . . . MQWFSEPO  (Required)
Description  . . . . . . . . . Service Policy for MQWF
```

Figure 18. 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 19.

```
Workload  Notes  Options  Help
------------------------------------------------------------------------------
                           Create a Workload
Command ===> ____________________________________________________________

Enter or change the following information:

Workload Name  . . . . . . . . MQWFWOLO  (Required)
Description  . . . . . . . . . Workload for MQWF
```

Figure 19. 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	MQWFWOLO
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  . . . . . . . . MQWFWOLO  (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 20. 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.	MQWFWOLO
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 21 on page 159.

```
*****************************************
* 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 21. 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, and so on.

```
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

                                                    ------Class-------
Action  Type      Description                       Service   Report
  __    ASCH      Use Modify to enter YOUR rules
  __    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        Workflow Request Classification
  __    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 22. 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 ===>
         -------Qualifier-------------            -------Class--------
Action   Type       Name     Start                Service     Report
                                          DEFAULTS: DEF_SC
____   1  TN         FMCIPGST                      PES_SC
____   1  TN         FMC*                          EXE_SC
```

Figure 23. 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
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: • 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 215. For a list of predefined MQSeries Workflow for z/OS message types, see "Static Workflow message classification" on page 210.	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 theses 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 . . . . . . . WLMAE=MQWFEXAE,WLMSN=&IWMSSNM,SRVEP=FMC
                               EMAIN,SRVNO=1_____________________________
                              ___________________________________

Limit on starting server address spaces for a subsystem instance:
1   1.  No limit
    2.  Single address space per system
    3.  Single address space per sysplex
```

Figure 24. 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 *UniqueSystemKey* 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 165.	

Using WLM

Parameter	Required or Optional	Description	Examples
Start Parameters	Required	Start parameters are the parameters required by the JCL procedure defined in *Procedure Name*. 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 configuration profile – as described in "Changing the number of server instances per address space" on page 123. The `WLMAE`[2] start parameter provides a way to set boundaries for the maximum and minimum number of server instances WLM may activate.	For the execution server, using the symbol `&IWMSSNM`[1] for the the subsystem instance name:`WLMAE=MQWFEXEAE, WLMSN=&IWMSSNM, SRVEP=FMCEMAIN,SRVNO=1` For the program execution server, using `WLMAE`[2] to have a maximum of 20 server instances and a minimum of 5 active instances: `WLMAE=MQWFPESAE/20/5, WLMSN=&IWMSSNM, SRVEP=FMCXMAIN,SRVNO=1`
Limit on starting server address space for a subsystem instance	Required	You must use the option: `1. No limit` The other options are not meaningful.	`No limit`

Notes:

1. 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 z/OS, 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 z/OS.
2. For the `WLMAE` parameter, you must specify an application environment name. This can optionally be followed by */max* or */max/min*, where *max* and *min* are the maximum and minimum number of server instances to be active at any point in time. The number of server instances may go outside the defined range while refreshing the WLM application environment, or if the *max* and *min* values are not exact multiples of the `SRVNO` parameter.

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 25 on page 163.

```
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 25. 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 26 and Figure 27.

```
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 26. WLM menu: Activate a service definition

```
                        Policy Selection List          Row 1 to 3 of 3
Command ===> ____________________________________________________________

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 27. WLM panel: Policy selection list

MQSeries Workflow for z/OS 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*. Note that the other WLM mode - the compatibility mode - is not supported in MQSeries Workflow for z/OS. 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 165.

Switching servers between WLM and non-WLM mode by importing an FDL file

An MQSeries Workflow for z/OS 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 28 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 '<WLMPESAE>'
END '<MQWFSYSN>'
```

Figure 28. 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 29 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 29. 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 24 on page 161), 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 EXE
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 24 on page 161.

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 shut down 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 73. 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.

Using WLM

Table 74. Unexpected runtime behavior of MQSeries Workflow with 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 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`.	1. Shut down the Workflow for z/OS system, as described in "Stopping the system" on page 89. 2. Shut down the administration server, as described in "Stopping the administration server" on page 88. 3. Run the JCL *CustHLQ*`.SFMCCNTL (FMCHJWIB)` 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 unexpected termination of MQWF server address spaces, WLM does not start new server address spaces.	If five unexpected terminations of server address spaces occur within ten minutes, WLM will quiesce the application environment for these server address spaces and shut down all running instances. **Note:** This rule does not apply when server address spaces are cancelled.	1. Make any necessary changes to fix the condition that causes WLM to quiesce the application environment. 2. Stop the server type that was affected by the terminations using the administration server command (see "Stopping servers" on page 92) in order to switch the corresponding application environment to the quiesced state. 3. Wait until the application environment is in the state **quiesced**. 4. Start the server type as described in "Starting servers" on page 91.

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 174.

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=KeyValue` 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 172
- "Mapping section" on page 172
- "Security section" on page 172

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 173 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 `inputqueuename`.
- 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 z/OS.

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                     =CICSEXCI
   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 101. 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 75. 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 76. 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	

For non-zero return codes, check the log file for more information.

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:FMCHEENV")/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:FMCHEENV")/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:FMCHEENV")/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 178
- "Deleting a program mapping definition" on page 178
- "Listing program mapping definitions" on page 178

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 z/OS 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 code page. MQSeries Workflow for z/OS relies on the converters available on the system and does not provide them as part of the product. 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 z/OS requires that certain naming restrictions are followed.

Naming Buildtime objects

The names given to MQSeries Workflow for z/OS 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 183.

Restrictions for passwords in CICS

Passwords specified in the Log on API call from CICS programs must only include characters contained in code page 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. See the *OS/390 C/C++ Programming Guide* for a list of supported code set converters.
6. By default, the conversion tool replaces all square brackets ('[' and ']') with parenthesis ('(' and ')') during the conversion. If you have code page conversion problems due to these replacement you can change the default behaviour of the conversion tool by changing the environment variable FMC_NO_BRACKET_REPLACE to any value you like. This variable is defined in the server environment file *CustHLQ*.SFMCDATA(FMCHEENV).

Return codes

The FDL code page conversion tool can return the following return codes:

Table 77. FDL code page conversion tool's return codes

Value	Description
0	Successful execution
4	Warning
12	Error
16	Severe error

For non-zero return codes, see the SYSOUT messages or log file for more information.

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:

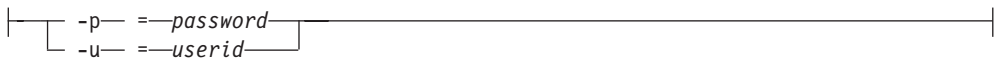
- Import process model information into the Workflow database.
- Translate and verify process model information that is stored in the Workflow database.
- Export process model information from the Workflow database.

FDL import/export tool's syntax

The following syntax diagram shows how to use the FMCH0IBA tool:

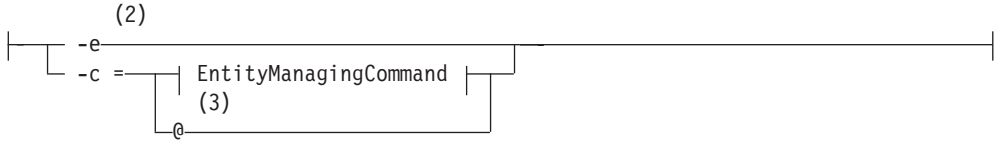
Import tool FMCH0IBA syntax

Logon:

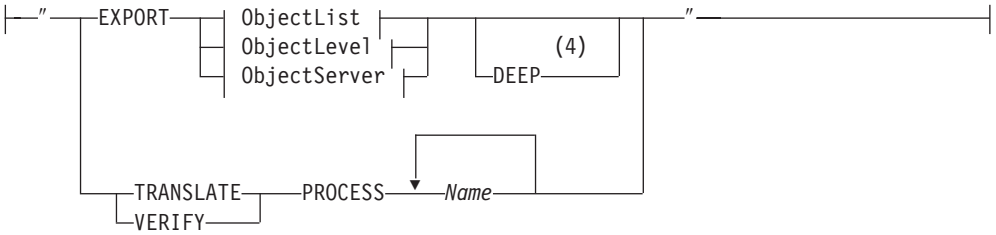


Import:

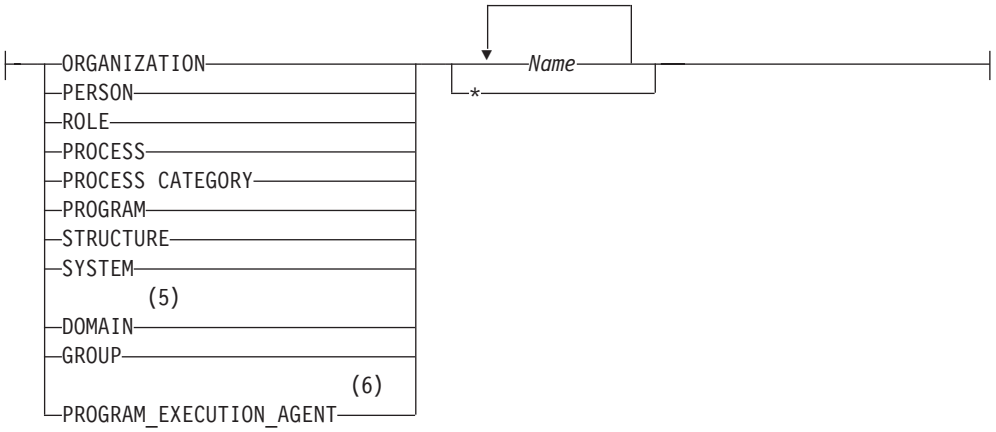
Export:



EntityManagingCommand:

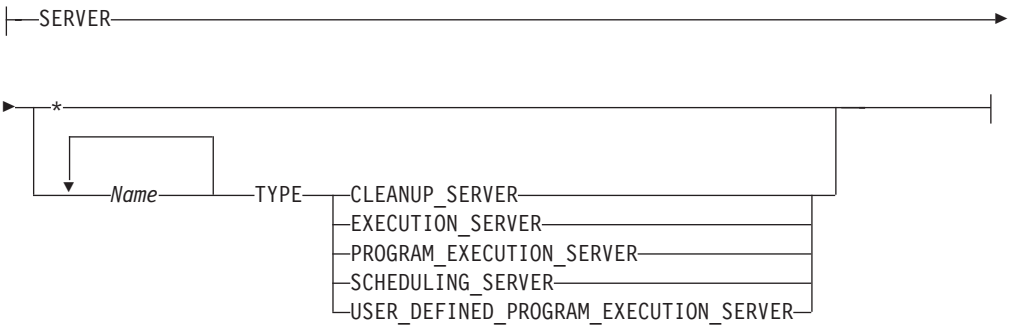


ObjectList:



ObjectLevel:

ObjectServer:



Notes:

1 When option **i** is selected, the file specified by the DD name FMCIIMP 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	*"Command string"*	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	*password*	This is the password for the specified user ID.
t		Translates and verifies a process model, however, only in import mode.
u	*userid*	This is the log on user ID for Workflow.
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 78. 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

The following examples show the use of different import, export and translate options *options* and DD statements. The examples all start the import tool, log on using the user ID *uid* and a password *pwd*:

```
//FMCHJRIF EXEC PROC=FMCHPBAT,PROGRAM=FMCH0IBA,
//         PARM='ENVAR("_CEE_ENVFILE=DD:FMCHEENV")/options'
//*
//FMCIIMP  DD   DISP=SHR,DSN=CustHLQ.SFMCDATA(FDL)
//FMCIEXP  DD   DISP=SHR,DSN=CustHLQ.SFMCDATA(OUT)
//FMCICMD  DD   DISP=SHR,DSN=CustHLQ.SFMCDATA(CMD)
//FMCILOG  DD   SYSOUT=*
//*
```

To import an FDL file

To import the FDL file that is specified by the DD name FMCIIMP, *options* should have the value:

```
-u uid -p pwd -i -o
```

To import an FDL file and translate the contained process models

To import the FDL file that is specified by the DD name FMCIIMP and translate the imported process models, *options* should have the value:

```
-u uid -p pwd -i -o -t
```

To import an FDL file and write messages in a separate log file

To import the FDL file that is specified by the DD name FMCIIMP and write all information, warning, and error messages to the log file that is specified by the DD name FMCILOG, *options* should have the value:

```
-u uid -p pwd -i -o
```

To export all workflow entities

To export the entities from the Workflow database to the output file specified by the DD name FMCIEXP, *options* should have the value:

```
-u uid -p pwd -e
```

To export all people

To export the definitions for all persons from the Workflow database to the output file specified by the DD name FMCIEXP, *options* should have the value:

```
-u uid -p pwd -e -c"EXPORT PERSON*"
```

To export individual people

To export the definitions for the people Eric and Tom from the Workflow database to the output file specified by the DD name FMCIEXP, *options* should have the value:

```
-u uid -p pwd -e -c"EXPORT PERSON ''ERIC'' ''TOM''"
```

To export an individual process (deep)

To export 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, *options* should have the value:

```
-u uid -p pwd -e -c"EXPORT PROCESS process1 DEEP"
```

To export Workflow entities using a command file

To 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 FMCICMD, *options* should have the value:

```
-u uid -p pwd -e -c @
```

To translate existing models

To translate an existing process model in the MQSeries Workflow for z/OS runtime database with the process name *process1*, *options* should have the value:

```
-u uid -p pwd -c "TRANSLATE PROCESS process1"
```

To translate existing process models using a command file

To translate the existing process models using the commands in the file that is specified by the DD name FMCICMD, *options* should have the value:

```
-u uid -p pwd -c @
```

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 195
- "Customization parameter file for a client on a queue manager" on page 198

Customization parameter file for a primary system

All the customization parameters for an MQSeries Workflow for z/OS primary system are entered into the customization parameter file. Each time that you create a new MQSeries Workflow for z/OS primary system in a new system group, you must copy and complete this file. This is done during precustomization.

The customization parameter file template is *CustHLQ*.SFMCDATA(FMCHECIF). During precustomization (in step 2 of the task "Create input files for customization" on page 24) you will have 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 jobs necessary to customize the Workflow system that is defined in this file.

The template file contains the following:

```
************************************************************************
*
* MQSeries Workflow for OS/390
*
* Customization Parameter File - for primary system
*
* The purpose of this file is to define the customization parameters
* for an MQWF primary system.
* It contains a list of key/value pairs which are used to tailor the
* customization jobs.
* These jobs as they are shipped with the product contain variables
* which must be replaced by the customer defined values.
*
* ---------------------------------------------------------------------
* Handling Instructions:
* - Do not delete a line in this file
* - Lines starting with an asterisk are comment lines
* - All keys and values must be enclosed in quotes
* - The syntax of the key/value pairs is:
*      'key' = 'value' comment
* - Some of the keys below contain volume labels for disks.
*   You can specify the volume label in the value field if you
*   want to allocate a dataset on a specific disk.
*   If you want SMS to allocate the file for you then you must
*   specify NOVOLUME (which is the default). This will
*   will lead to the deletion of the VOL=SER parameter in the jobs.
*
************************************************************************

*--- Installation Scope Identifiers --------------------------------
```

```
  * MQWF Customization High Level Qualifier
   'MQWFCHLQ' = 'MQWFCHLQ'

*--- System Group Scope Identifiers --------------------------------

  * MQWF Installation High Level Qualifier
   'MQWFIHLQ' = 'MQWFIHLQ'

  * MQWF System Group Name
   'MQWFSGNM' = 'MQWFSGNM'

  * MQWF System Prefix
   'MQWFSYSP' = 'MQWFSYSP'

  * MQ-Series Cluster Name
   'MQWFCLST' = 'MQWFCLST'

  * MQWF System Group Locale
   'MQWFSGLC' = 'MQWFSGLC'

  * DB2 Sample Load Library
   'DB2SMPRL' = 'DB2SMPRL'

  * DB2 Sample DBRM Library
   'DB2SMPDL' = 'DB2SMPDL'

  * DB2 System Group Prefix
    'DB2SGPRE' = 'DB2SGPRE'

  * DB2 Storage Group Name
    'DB2STGNW' = 'DB2STGNW'

  * DB2 Data Storage Group Dataset Prefix
    'DB2STGPW' = 'DB2STGPW'

  * DB2 Data Storage Group Volume Set
    'DB2STGVW' = 'DB2STGVW'
                                * volume name or
                                * '*' for SMS managed volumes

  * DB2 Audit Storage Group Name
    'DB2STGNA' = 'DB2STGNA'

  * DB2 Audit Storage Group Dataset Prefix
    'DB2STGPA' = 'DB2STGPA'

  * DB2 Audit Storage Group Volume Set
    'DB2STGVA' = 'DB2STGVA'
                                * volume name or
                                * '*' for SMS managed volumes

  * DB2 Workflow Database Name
    'DB2DBNAM' = 'DB2DBNAM'

  * DB2 PES Mapping Database Name
    'DB2MDBNM' = 'DB2MDBNM'

  * DB2 PES Directory Database Name
    'DB2PDBNM' = 'DB2PDBNM'

  * DB2 Workflow 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'

*--- System Scope Identifiers -------------------------------------

  * MQWF Unique System Key
   'MQWFUKEY' = 'MQWFUKEY'

  * MQWF Configuration Key
   'MQWFCFGK' = 'MQWFCFGK'

  * 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'

  * TRACE Writer Start Procedure Name
   'TRCWPRC' = 'TRCWPRC'

  * TRACE Writer Stop Procedure Name
   'TRCSPRC' = 'TRCSPRC'

  * ARM Restart Policy Name
   'ARMPOLNM' = 'ARMPOLNM'

  * ARM Restart Element Name Suffix
   'ARMRESFX' = 'ARMRESFX'

  * WLM Application Environment Name for MQWF Execution Server (EXE)
   'WLMAEEXE' = 'WLMAEEXE'

  * WLM Application Environment Name for MQWF Program Exe Server (PES)
   'WLMAEPES' = 'WLMAEPES'

  * MQ-Series Cluster Namelist Name
   'MQCLNAME' = 'MQCLNAME'

  * MQSeries Queue Manager Name
   'MQQMNAME' = 'MQQMNAME'

  * MQ Host Name - TCP/IP Address of the OS/390 system where
  *  the MQSeries Queue Manager resides
   'STCPADDR' = 'STCPADDR'

  * TCP/IP Port of the MQSeries Queue Manager (default is port 1414)
   'STCPPORT' = 'STCPPORT'


*--- Flags and High Level Qualifiers ------------------------------

  * CICS Flag

   * (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.)

  * if CICS is Installed
```

```
   'CICSFL  ' = '        '

  * if CICS is NOT Installed
  *'CICSFL  ' = '*       '

  * CICS Installation High Level Qualifier
   'CICSLPFX' = 'CICSLPFX'

  * 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'

  * XML Tool Kit Flag

    * (Used to control whether to include an XML installation library
    *  into the steplib concatenation of the jcl procedures. One of the
    *  following two lines must be commented out with a '*'.
    *  If the XML Tool Kit is not installed the parameter
    *  XMLTKPFX must not be customized.)

  * if XML Tool Kit is installed
  *'XMLTFL' = '        '

  * if CICS is NOT Installed
   'XMLTFL' = '*       '

  * XML Tool Kit Installation High Level Qualifier
   'XMLTKPFX' = 'XMLTKPFX'

*--- Subsystem Identifiers --------------------------------------

  * DB2 Subsystem Name
   'DB2SSYSN' = 'DB2SSYSN'

  * CICS Group Name
   'CICSGRPN' = 'CICSGRPN'

*--- Volume Parameters ------------------------------------------
  * The following variables are used in various jobs to specify
  * the disk volumes for datasets to be allocated.
  * If SMS is to be used , leave the following lines unchanged,
  * otherwise, change these lines to comments.
  * The variables must then be changed manually in the relevant job.

   'CICSVOL'  = '*'
   'VOLUME  ' = 'NOVOLUME'
```

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:

```
************************************************************************
*
* MQSeries Workflow for OS/390
*
* Customization Parameter File - for an additional system
*
* The purpose of this file is to define the customization parameters
* for an additional MQWF system.
* It contains a list of key/value pairs which are used to tailor the
* customization jobs.
* These jobs as they are shipped with the product contain variables
* which must be replaced by the customer defined values.
*
* ---------------------------------------------------------------------
* Handling Instructions:
* - Do not delete a line in this file
* - Lines starting with an asterisk are comment lines
* - All keys and values must be enclosed in quotes
* - The syntax of the key/value pairs is:
*      'key' = 'value' comment
* - Some of the keys below contain volume labels for disks.
*   You can specify the volume label in the value field if you
*   want to allocate a dataset on a specific disk.
*   If you want SMS to allocate the file for you then you must
*   specify NOVOLUME (which is the default). This will
*   will lead to the deletion of the VOL=SER parameter in the jobs.
*
************************************************************************

*--- Installation Scope Identifiers --------------------------------

  * MQWF Installation High Level Qualifier
  *'MQWFIHLQ' = '<MQWFIHLQ>'             primary system value
   'MQWFIHLQ' = 'MQWFIHLQ'

*--- System Group Scope Identifiers --------------------------------

  * MQWF Customization High Level Qualifier
   'MQWFCHLQ' = 'MQWFCHLQ'

  * MQWF System Group Name
   'MQWFSGNM' = '<MQWFSGNM>'

  * MQWF Primary System Name of System Group <MQWFSGNM>
   'MQWFPSYN' = '<MQWFSYSN>'

  * MQWF System Prefix
   'MQWFSYSP' = '<MQWFSYSP>'

  * MQ-Series Cluster Name
   'MQWFCLST' = '<MQWFCLST>'

  * MQWF System Group Locale
   'MQWFSGLC' = '<MQWFSGLC>'

  * DB2 Sample Load Library
  *'DB2SMPRL' = '<DB2SMPRL>'             primary system value
   'DB2SMPRL' = 'DB2SMPRL'

  * DB2 Sample DBRM Library
  *'DB2SMPDL' = '<DB2SMPDL>'             primary system value
```

```
   'DB2SMPDL' = 'DB2SMPDL'

  * DB2 System Group Prefix
    'DB2SGPRE' = '<DB2SGPRE>'

  * DB2 Storage Group Name
    'DB2STGNW' = '<DB2STGNW>'

  * DB2 Data Storage Group Dataset Prefix
    'DB2STGPW' = '<DB2STGPW>'

  * DB2 Data Storage Group Volume Set
    'DB2STGVW' = '<DB2STGVW>'
                                 * volume name or
                                 * '*' for SMS managed volumes

  * DB2 Audit Storage Group Name
    'DB2STGNA' = '<DB2STGNA>'

  * DB2 Audit Storage Group Dataset Prefix
    'DB2STGPA' = '<DB2STGPA>'

  * DB2 Audit Storage Group Volume Set
    'DB2STGVA' = '<DB2STGVA>'
                                 * volume name or
                                 * '*' for SMS managed volumes

  * DB2 Workflow Database Name
    'DB2DBNAM' = '<DB2DBNAM>'

  * DB2 PES Mapping Database Name
    'DB2MDBNM' = '<DB2MDBNM>'

  * DB2 PES Directory Database Name
    'DB2PDBNM' = '<DB2PDBNM>'

  * DB2 Workflow 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>'

*--- System Scope Identifiers -------------------------------------

  * MQWF Unique System Key
   'MQWFUKEY' = 'MQWFUKEY'

  * MQWF Configuration Key
   'MQWFCFGK' = 'MQWFCFGK'

  * MQWF System Name
   'MQWFSYSN' = 'MQWFSYSN'

  * MQWF System Identifier
   'MQWFSYID' = 'MQWFSYID'

  * MQWF Server Started Task RACF UserId
   'STTSKUID' = 'STTSKUID'

  * MQWF Server Started Task RACF GroupId
   'STTSKGRP' = 'STTSKGRP'
```

```
  * CTRACE Component Name
   'CTRCNAME' = 'CTRCNAME'

  * TRACE Writer Start Procedure Name
   'TRCWPRC' = 'TRCWPRC'

  * TRACE Writer Stop Procedure Name
   'TRCSPRC' = 'TRCSPRC'

  * ARM Restart Policy Name
   'ARMPOLNM' = 'ARMPOLNM'

  * ARM Restart Element Name Suffix
   'ARMRESFX' = 'ARMRESFX'

  * WLM Application Environment Name for MQWF Execution Server (EXE)
   'WLMAEEXE' = 'WLMAEEXE'

  * WLM Application Environment Name for MQWF Program Exe Server (PES)
   'WLMAEPES' = 'WLMAEPES'

  * MQ-Series Cluster Namelist Name
   'MQCLNAME' = 'MQCLNAME'

  * MQSeries Queue Manager Name
   'MQQMNAME' = 'MQQMNAME'

  * MQ Host Name - TCP/IP Address of the OS/390 system where
  *  the MQSeries Queue Manager resides
   'STCPADDR' = 'STCPADDR'

  * TCP/IP Port of the MQSeries Queue Manager (default is port 1414)
   'STCPPORT' = 'STCPPORT'


*--- Flags and High Level Qualifiers ------------------------------

  * CICS Flag

   * (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.)

  * if CICS is Installed
   'CICSFL  ' = '        '

  * if CICS is NOT Installed
  *'CICSFL  ' = '*       '

  * CICS Installation High Level Qualifier
  *'CICSLPFX' = '<CICSLPFX>'             primary system value
   'CICSLPFX' = 'CICSLPFX'

  * DB2 Installation High Level Qualifier
  *'DB2INHLQ' = '<DB2INHLQ>'             primary system value
   'DB2INHLQ' = 'DB2INHLQ'

  * MQSeries Installation High Level Qualifier
  *'MQPREFIX' = '<MQPREFIX>'             primary system value
   'MQPREFIX' = 'MQPREFIX'

  * Language Environment High Level Qualifier
  *'LELIBPFX' = '<LELIBPFX>'             primary system value
   'LELIBPFX' = 'LELIBPFX'
```

```
  * C/C++ Installation High Level Qualifier
  *'CLIBRPFX' = '<CLIBRPFX>'             primary system value
   'CLIBRPFX' = 'CLIBRPFX'

  * Cobol Installation High Level Qualifier
  *'CBLIBPFX' = '<CBLIBPFX>'             primary system value
   'CBLIBPFX' = 'CBLIBPFX'

  * IMS Installation High Level Qualifier
  *'IMSLIBPX' = '<IMSLIBPX>'             primary system value
   'IMSLIBPX' = 'IMSLIBPX'

  * ICONV Installation High Level Qualifier
  *'ICONVPFX' = '<ICONVPFX>'             primary system value
   'ICONVPFX' = 'ICONVPFX'

  * IPCS Installation High Level Qualifier
  *'IPCSPRFX' = '<IPCSPRFX>'             primary system value
   'IPCSPRFX' = 'IPCSPRFX'

*--- Subsystem Identifiers ---------------------------------------

  * DB2 Subsystem Name
   'DB2SSYSN' = 'DB2SSYSN'

  * CICS Group Name
  *'CICSGRPN' = '<CICSGRPN>'             primary system value
   'CICSGRPN' = 'CICSGRPN'

*--- Volume Parameters -------------------------------------------
  * The following variables are used in various jobs to specify
  * the disk volumes for datasets to be allocated.
  * If SMS is to be used , leave the following lines unchanged,
  * otherwise, change these lines to comments.
  * The variables must then be changed manually in the relevant job.

   'CICSVOL'  = '*'
   'VOLUME  ' = 'NOVOLUME'
```

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(FMCHECCL), it contains the following:

```
**********************************************************************
*
* MQSeries Workflow for z/OS
*
* Customization Parameter File - for client
*
* The purpose of this file is to define the customization parameters
* for a new MQWF client.
* It contains a list of key/value pairs which are used to tailor the
* customization jobs.
* These jobs as they are shipped with the product contain variables
* which must be replaced by the customer defined values.
*
* --------------------------------------------------------------------
* Handling Instructions:
* - Do not delete a line in this file
* - Lines starting with an asterisk are comment lines
* - All keys and values must be enclosed in quotes
* - The syntax of the key/value pairs is:
*      'key' = 'value' comment
*
```

```
*********************************************************************

*--- Installation Scope Identifiers --------------------------------
 * These parameters are changed during the System Customization
 *   (two phase cutomization: system , client)

  * MQWF Installation High Level Qualifier
   'MQWFIHLQ' = '<MQWFIHLQ>'

  * MQWF System Customization High Level Qualifier
   'MQWFCHLQ' = '<MQWFCHLQ>'

*--- Client Scope Identifiers --------------------------------------

  * MQWF Client Customization High Level Qualifier
   'CLNTCHLQ' = 'CLNTCHLQ'

  * MQWF Client Configuration Key
   'CLNTCFGK' = 'CLNTCFGK'

  * MQSeries Client Queue Manager Name
   'CLQMNAME' = 'CLQMNAME'

  * MQ Host Name - TCP/IP Address of the OS/390 system where
  *  the MQSeries Queue Manager resides
   'CTCPADDR' = 'CTCPADDR'

  * TCP/IP Port of the MQSeries Queue Manager (default is port 1414)
   'CTCPPORT' = 'CTCPPORT'

  * CICS Group Name
   'CICSGRPC' = 'CICSGRPC'

  * CICS Installation High Level Qualifier
   * CICSLPFX = '<CICSLPFX>'    used for system customization
   'CICSLPFX' = 'CICSLPFX'
```

Appendix H. Configuration profiles

There are two configuration profiles, these are:

- "Server configuration profile"
- "Client configuration profile" on page 203

Server configuration profile

Each MQSeries Workflow for z/OS system has a configuration profile in *CustHLQ*.SFMCDATA(FMCHEMPR). This profile contains system settings that affect the operation of MQSeries Workflow for z/OS servers and tools. Some of the values are substituted automatically during customization, these must not be changed. Changes made to the configuration 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 89.

Table 79. Server configuration profile settings

Variable	Value may be changed?	Description
System	No	This value should be your value for *System* 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 191.
SystemGroup	No	This value should be your value for *SystemGroup* 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 191.
DatabaseName	No	This value should be your value for *WorkflowDatabaseName* 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 191.
DbPlan	No	This value should be your value for *DB2Plan* 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 191.
DbSubSystem	No	This value should be your value for *DB2SubSystem* 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 191.
ExecutionServer OperationMode	No	For future use.
APITimeOut	Tune carefully	API time-out in milliseconds.
FMLConnectName	No	*QueueManager* and Workflow context.
FMLConnect DelayTime	Tune carefully	Interval in milliseconds to wait between consecutive retries to reconnect to the *QueueManager*.
FMC_TRACE_ CRITERIA	Yes	Determines the level of trace detail provided by newly started servers or tools. Valid values are between **0,**0000,00000000 (no trace) and **3,**FFFF,FFFFFFFF (full trace). For more information about the trace variables, see "MQSeries Workflow trace variables" on page 139.
FMC_SIMPLE_ TRACE_ONLY	Yes	Activates simple tracing in newly started servers, clients and tools. For servers, valid values are YES or NO. For clients and tools, only the value YES is meaningful. For more information about the trace variables, see "MQSeries Workflow trace variables" on page 139.

Table 79. Server configuration profile settings (continued)

Variable	Value may be changed?	Description
FMC_TRACE_ BUFFER_SIZE	Yes	This determines the size of the trace buffers in each server. The default value is 256 KB. Changes to this variable only take effect after the server is restarted. For more information about the trace variables, see "MQSeries Workflow trace variables" on page 139.
FMC_TRACE_ FILE_SIZE	Yes	This determines how much trace information will be written to each data set before starting writing to the next one. The default value is 5 MB. The value of this variable must not be larger than the size of the trace data sets. For more information about the trace variables, see "MQSeries Workflow trace variables" on page 139.
FMC_REFRESH_ COUNT_FOR_ TRACE_ CRITERIA	Yes	Determines how often the trace options are refreshed. The default value is every 50 transactions. Setting this to zero disables the refreshing. For more information about the trace variables, see "MQSeries Workflow trace variables" on page 139.
FMC_NUMBER _OF_TRACE _FILES	Yes	Determines how many data sets the external trace writer will attempt to write to before starting to overwrite the first data set. The default value is 3. For more information about the trace variables, see "MQSeries Workflow trace variables" on page 139.
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 *InstHLQ*.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. Initially, the server start job does not allow more than eight, for more information on changing this value, see "Changing the number of server instances per address space" on page 123.
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. Initially, the server start job does not allow more than eight, for more information on changing this value, see"Changing the number of server instances per address space" on page 123.
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 130 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 110.

After customization, your configuration 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 angled brackets ('<...>'):

```
*********************************************************************
*
* Description: MQ WorkFlow server configuration 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:0,0000,00000000
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:<MQWFUKEY>
Configuration.<MQWFCFGK>.WaitBetweenQInhibitAndAllowed:30
```

Client configuration profile

Each MQSeries Workflow for z/OS system has a client configuration profile in *CustHLQ*.SFMCDATA(FMCHECPR). Some of the values are substituted automatically during customization, these must not be changed.

Table 80. Client configuration profile settings

Variable	Value may be changed?	Description
System	No	This value should be your value for *System* 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 191.
SystemGroup	No	This value should be your value for *SystemGroup* 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 191.
DatabaseName	No	This value should be your value for *WorkflowDatabaseName* 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 191.
DbPlan	No	This value should be your value for *DB2Plan* 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 191.
DbSubSystem	No	This value should be your value for *DB2SubSystem* 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 191.
ExecutionServer OperationMode	No	For future use.
APITimeOut	Tune carefully	API time-out in milliseconds.
FMLConnectName	No	*QueueManager* and Workflow context.

Table 80. Client configuration profile settings (continued)

Variable	Value may be changed?	Description
FMLConnect DelayTime	Tune carefully	Interval in milliseconds to wait between consecutive retries to reconnect to the *QueueManager*.
FMC_TRACE_ CRITERIA	Yes	Determines the level of trace detail provided by newly started servers or tools. Valid values are between **0,**0000,00000000 (no trace) and **3,**FFFF,FFFFFFFF (full trace). For more information, see "MQSeries Workflow trace variables" on page 139.
FMC_SIMPLE_TRACE_ONLY	Yes	Activates simple tracing in newly started servers, clients and tools. For servers, valid values are YES or NO. For clients and tools, only the value YES is meaningful. For more information, see "MQSeries Workflow trace variables" on page 139.
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 *InstHLQ*.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 123.
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 123.
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 130 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 110.

The client configuration profile contains the following before your customization values are automatically substituted:

```
*********************************************************************
*
* Description: MQ WorkFlow client configuration 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 z/OS system has a server environment variable file in *CustHLQ*.SFMCDATA(FMCHEENV). This file contains system settings that affect the operation of MQSeries Workflow for z/OS 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 89.

Table 81. Server environment variable file settings

Variable	Value may be changed?	Description
_ICONV_UCS2_PREFIX	No	Your value for *ICONVInstHLQ* in Table 4 on page 13.
LC_ALL	No	Selects the code page to be used by the Workflow servers and tools. This is set to your value for *SystemGroupLocale*, see Table 2 on page 10 for more details.
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_CURRENT_CONFIG=<MQWFCFGK>
FMC_DEFAULT_CONFIGURATION=<MQWFCFGK>
FMC_ELAPSED_TIME=YES
FMC_IENV=1
```

Client environment file

Each MQSeries Workflow for z/OS system has a client environment variable file in *CustHLQ*.SFMCDATA(FMCHECEV). This file contains system settings that affect the operation of MQSeries Workflow for z/OS clients. Some of the values are substituted automatically during customization, these must not be changed.

Table 82. Client environment variable file settings

Variable	Value may be changed?	Description
_ICONV_UCS2_PREFIX	No	Your value for *ICONVInstHLQ* in Table 4 on page 13.

Table 82. Client environment variable file settings (continued)

Variable	Value may be changed?	Description
LC_ALL	No	Selects the code page to be used by the Workflow servers and tools. This is set to your value for *SystemGroupLocale*, 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 "MQSeries Workflow trace variables" on page 139.
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 83) and `MQWIH_ServiceName` (described in Table 84).

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 83. MQWIH_ServiceStep field definition

Name	(Position, Length)	Description	Example
Prefix	(1 , 3)	Each message type starts with a fixed prefix (**FMC**).	**FMC**EPICS
Class Type	(4 , 1)	The class type specifies the kind of request, e.g. Process Execution (**E**), or a Query (**Q**).	FMC**E**PICS
Object Type	(5 , 2)	The object type classifies the Workflow object type, e.g. Process Model (**PM**), Process Template (**PT**) or Process Instance (**PI**).	FMCE**PI**CS
Action Type	(7 , 2)	The action type specifies the specific action executed to satisfy the Workflow request, e.g. CreateAndStart (**CS**) or TErminate (**TE**).	FMCEPI**CS**

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 210.

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 84. MQWIH_ServiceName field definition

Name	(Position, Length)	Description	Example
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.	

Table 84. MQWIH_ServiceName field definition (continued)

Name	(Position, Length)	Description	Example
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.	″**CICSEXCI**″

Static Workflow message classification

The tables below contains all MQSeries Workflow Server Messages sent to the MQSeries Workflow for z/OS 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:

Table 85. MQSeries Workflow server message types

Message Type	Described in
Process Template	Table 86 on page 211
Process Template List	Table 87 on page 211
Process	Table 88 on page 211
Process InstList	Table 89 on page 211
Work Item	Table 90 on page 212
Activity	Table 91 on page 212
User Information	Table 92 on page 213
Process Monitor	Table 93 on page 213
WorkList	Table 94 on page 213
PEA/PES-Server	Table 95 on page 213
PEA/PES Reply	Table 96 on page 213
Scheduling	Table 97 on page 214
SubProcess	Table 98 on page 214
Internal Server	Table 99 on page 214

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 86. Process Template messages

Message Type	MQWIH_ServiceStep	Class Type	Object Type	Action Type
DelProcTempl	FMCXPTDL	**X** (other action)	**P**rocess **T**emplate	**DE**L**E**te
QryProcTempl	FMCGPTPR	**G**et (property access)	**P**rocess **T**emplate	**PR**operties
QryProcessTemplates	FMCQPTMU	**Q**uery	**P**rocess **T**emplate	**MU**ltiple
QryProcTemplInpCtnr	FMCQPTCT	**Q**uery	**P**rocess **T**emplate	**C**on**T**ainer

Process Template List messages

Table 87. Process Template List messages

Message Type	MQWIH_ServiceStep	Class Type	Object Type	Action Type
QryProcTemplLists	FMCQLTMU	**Q**uery	**L**ist **T**emplate	**MU**ltiple
QryProcTemplListData	FMCGLTPR	**G**et (property access)	**L**ist **T**emplate	**PR**operties
CreateProcTemplList	FMCXLTCR	**X** (other action)	**L**ist **T**emplate	**CR**eate
ChgProcTemplList	FMCXLTCH	**X** (other action)	**L**ist **T**emplate	**CH**ange
DelProcTemplList	FMCXLTDL	**X** (other action)	**L**ist **T**emplate	**DE**L**E**te

Process messages

Table 88. Process messages

Message Type	MQWIH_ServiceStep	Class Type	Object Type	Action Type
ChgProcInstDesc	FMCXPICD	**X** (other action)	**P**rocess **I**nstance	**C**hange **D**escription
ChgProcInstName	FMCXPICN	**X** (other action)	**P**rocess **I**nstance	**C**hange **N**ame
CreateProcInst	FMCEPICR	**E**xecute process	**P**rocess **I**nstance	**CR**eate
CreateStartProcInst	FMCEPICS	**E**xecute process	**P**rocess **I**nstance	**C**reate and **S**tart
DelProcInst	FMCXPIDL	**X** (other action)	**P**rocess **I**nstance	**DE**L**E**te
QryProcInst	FMCGPIPR	**G**et (property access)	**P**rocess **I**nstance	**PR**operties
QryProcInstCtnr	FMCGPICT	**G**et (property access)	**P**rocess **I**nstance	**C**on**T**ainer
QryProcessInstances	FMCQPIMU	**Q**uery	**P**rocess **I**nstance	**MU**ltiple
ResumeProcInst	FMCXPIRE	**X** (other action)	**P**rocess **I**nstance	**RE**sume
StartProcInst	FMCEPIST	**E**xecute process	**P**rocess **I**nstance	**ST**art
SuspendProcInst	FMCXPISU	**X** (other action)	**P**rocess **I**nstance	**SU**spend
TermProcInst	FMCXPITE	**X** (other action)	**P**rocess **I**nstance	**TE**rminate

Process InstList messages

Table 89. Process InstList messages

Message Type	MQWIH_ServiceStep	Class Type	Object Type	Action Type
QryProcInstLists	FMCQLPMU	**Q**uery	**L**ist **P**rocess	**MU**ltiple
QryProcInstListData	FMCGLPPR	**G**et (property access)	**L**ist **P**rocess	**PR**operties
CreateProcInstList	FMCXLPCR	**X** (other action)	**L**ist **P**rocess	**CR**eate
ChgProcInstList	FMCXLPCH	**X** (other action)	**L**ist **P**rocess	**CH**ange

Table 89. Process InstList messages (continued)

Message Type	MQWIH_ServiceStep	Class Type	Object Type	Action Type
DelProcInstList	FMCXLPDL	**X** (other action)	**L**ist **P**rocess	**D**e**L**ete

Work Item messages

Table 90. Work Item messages

Message Type	MQWIH_ServiceStep	Class Type	Object Type	Action Type
CancelCheckout WorkItem	FMCXWICC	**X** (other action)	**W**ork **I**tem	**C**ancel **C**heckout
CheckinWorkItem	FMCEWICI	**E**xecute process	**W**ork **I**tem	**C**heck**I**n
CheckoutWorkItem	FMCEWICO	**E**xecute process	**W**ork **I**tem	**C**heck**O**ut
ChgWorkItemDesc	FMCXWICD	**X** (other action)	**W**ork **I**tem	**C**hange **D**escription
ChgWorkItemName	FMCXWICN	**X** (other action)	**W**ork **I**tem	**C**hange **N**ame
DelWorkItem	FMCXWIDL	**X** (other action)	**W**ork **I**tem	**D**e**L**ete
FinishWorkItem	FMCXWIMF	**X** (other action)	**W**ork **I**tem	**M**anual **F**inish
ForceFinishWorkItem	FMCXWIFF	**X** (other action)	**W**ork **I**tem	**F**orce **F**inish
ForceRestartWorkItem	FMCXWIFR	**X** (other action)	**W**ork **I**tem	**F**orce **R**estart
PutWorkItemOffHold	FMCXWIPF	**X** (other action)	**W**ork **I**tem	**P**ut o**F**f hold
PutWorkItemOnHold	FMCXWIPN	**X** (other action)	**W**ork **I**tem	**P**ut o**N** hold
QryWorkItem	FMCGWIPR	**Q**uery	**W**ork **I**tem	**PR**operties
QryWorkItemCtnr	FMCGWICT	**Q**uery	**W**ork **I**tem	**C**on**T**ainer
QryWorkItems	FMCQWIMU	**Q**uery	**W**ork **I**tem	**MU**ltiple
RestartWorkItem	FMCXWIMR	**X** (other action)	**W**ork **I**tem	**M**anual **R**estart
StartSupportTool	FMCXWISS	**X** (other action)	**W**ork **I**tem	**S**tart **S**upport tool
StartWorkItem	FMCEWIST	**E**xecute process	**W**ork **I**tem	**ST**art
TerminateWorkItem	FMCXWITE	**X** (other action)	**W**ork **I**tem	**TE**rminate
TransferWorkItem	FMCXWITR	**X** (other action)	**W**ork **I**tem	**TR**ansfer

Activity messages

Table 91. Activity messages

Message Type	MQWIH_ServiceStep	Class Type	Object Type	Action Type
CancelCheckOutAct	FMCXAICC	**X** (other action)	**A**ctivity **I**nstance	**C**ancel **C**heckout
ForceFinishAct	FMCXAIFF	**X** (other action)	**A**ctivity **I**nstance	**F**orce **F**inish
ForceRestartAct	FMCXAIFR	**X** (other action)	**A**ctivity **I**nstance	**F**orce **R**estart
RescheduleAct	FMCXAIRS	**X** (other action)	**A**ctivity **I**nstance	**R**e**S**chedule
TransferAct	FMCXAITR	**X** (other action)	**A**ctivity **I**nstance	**TR**ansfer

User Information messages

Table 92. User Information messages

Message Type	MQWIH_ServiceStep	Class Type	Object Type	Action Type
ChgSubstitute	FMCXUSCU	**X** (other action)	**US**er	**C**hange s**U**bstitute
QryUserDetails	FMCGUSPR	**Q**uery	**US**er	**PR**operties
ChgAbsence	FMCXUSCA	**X** (other action)	**US**er	**C**hange **A**bsence

Process Monitor messages

Table 93. Process Monitor messages

Message Type	MQWIH_ServiceStep	Class Type	Object Type	Action Type
QryProcInstStatus	FMCQPISA	**Q**uery	**P**rocess **I**nstance	**St**A**t**us
QryActCtnr	FMCQAICT	**Q**uery	**A**ctivity **I**nstance	**C**on**T**ainer

WorkList messages

Table 94. WorkList messages

Message Type	MQWIH_ServiceStep	Class Type	Object Type	Action Type
QryWorkLists	FMCQLWMU	**Q**uery	**L**ist **W**ork	**MU**ltiple
QryWorkListData	FMCGLWPR	**G**et (property access)	**L**ist **W**ork	**PR**operties
CreateWorkList	FMCXLWCR	**X** (other action)	**L**ist **W**ork	**CR**eate
ChgWorkList	FMCXLWCH	**X** (other action)	**L**ist **W**ork	**CH**ange
DelWorkList	FMCXLWDL	**X** (other action)	**L**ist **W**ork	**D**e**L**ete
CheckoutNext WorkItem	FMCELWCO	**E**xecute process	**L**ist **W**ork	**C**heck**O**ut
CheckinNext WorkItem	FMCELWCI	**E**xecute process	**L**ist **W**ork	**C**heck**I**n
StartNextWorkItem	FMCELWST	**E**xecute process	**L**ist **W**ork	**ST**art

PEA/PES Server messages

Table 95. PEA-Server messages

Message Type	MQWIH_ServiceStep	Class Type	Object Type	Action Type
InvokeProgram	FMCIPGST	**I**nvoke program	**P**ro**G**ram	**ST**art
ActivityExpired	FMCXAIAE	**X** (other action)	**A**ctivity **I**nstance	**A**ctivity **E**xpiration

PEA/PES Reply messages

Table 96. PEA Reply messages

Message Type	MQWIH_ServiceStep	Class Type	Object Type	Action Type
ProgramError	FMCEPGER	Execute **P**rocess	**P**ro**G**ram	**ER**orr
ProgramFinished	FMCEPGFI	Execute **P**rocess	**P**ro**G**ram	**FI**nished

Scheduling messages

Table 97. Scheduling messages

Message Type	MQWIH_ServiceStep	Class Type	Object Type	Action Type
CleanupProcInst	FMCSPICL	Intra-**S**erver	**P**rocess **I**nstance	CLeanup
CleanupWIs	FMCSWICL	Intra-**S**erver	**W**ork **I**tem	**CL**eanup
AutoResumeProcess	FMCSPIAR	Intra-**S**erver	**P**rocess **I**nstance	**A**uto **R**esume
CheckNotifItems	FMCSXXCE	Intra-**S**erver	**XX** (other)	**C**h**E**ck notifItems
Create Notifications	FMCSXXCF	Intra-**S**erver	**XX** (other)	**C**reate noti**F**ications
Expiration Notifications	FMCSXXCF	Intra-**S**erver	**XX** (other)	**E**xpiration **N**otificatons

SubProcess messages

Table 98. SubProcess messages

Message Type	MQWIH_ServiceStep	Class Type	Object Type	Action Type
CreateStartSub ProcInst	FMCSPSCS	Intra-**S**erver	**P**rocess**S**ub	**C**reate and **S**tart
SuspendSubProcInst	FMCSPSSU	Intra-**S**erver	**P**rocess**S**ub	**SU**spend
ResumeSubProcInst	FMCSPSRE	Intra-**S**erver	**P**rocess**S**ub	**RE**sume
TermSubProcInst	FMCSPSTE	Intra-**S**erver	**P**rocess**S**ub	**TE**rminate
DelSubProcInst	FMCSPSDL	Intra-**S**erver	**P**rocess**S**ub	**D**e**L**ete
SubProcInstError	FMCSPSER	Intra-**S**erver	**P**rocess**S**ub	**ER**orr
SubProcInstFinished	FMCSPSFI	Intra-**S**erver	**P**rocess**S**ub	**FI**nished

Internal Server messages

Table 99. Internal Server messages

Message Type	MQWIH_ServiceStep	Class Type	Object Type	Action Type
ChndTxnUpdate WorkItems	FMCSWIUD	Intra-**S**erver	**W**ork **I**tem	**U**p**D**ate
PhysDelProcInst	FMCSPIPD	Intra-**S**erver	**P**rocess **I**nstance	**P**hysical **D**elete
ChndTxnDel WorkItems	FMCSWIDL	Intra-**S**erver	**W**ork **I**tem	**D**e**L**ete

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-------------             -------Class--------
Action   Type       Name     Start                Service     Report
                                          DEFAULTS: DEF_SC
____  1  PC         %%%%0009 9
____  2    PC         EXCI*    17
____  3      PC         CICSEXCI 25               PES_SC
```

Figure 30. 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.

Appendix L. Error reporting

This appendix describe the following error reporting formats:

Error log record entries

The error log contains records that are created for every occurrence of an error. The layout of error log records held in the error log is described below:

Table 100. Error log record entries

Name	Description
Object ID	Uniquely identifies an error report created by the administration server.
Creation time	The time and date the error log entry was created in the database.
System name	The system name. Taken from system settings by the administration server.
System group name	The system group name of which the system is a member. Taken from system settings by the administration server.
Version number	The MQSeries Workflow version number.
Release number	The MQSeries Workflow release number.
Modification level number	The MQSeries Workflow modification level number.
Service pack (optional)	The MQSeries Workflow service pack level.
Component ID	The MQSeries Workflow component that has reported the error.
Component name	The MQSeries Workflow component that caused, or is related to this error.
Platform	Integer indicating the operating system platform on which the reporting MQSeries Workflow component is executing: 1. OS/2 2. AIX(R) 3. Windows 3.1 4. Windows NT 5. Windows 95 6. HP 7. Sun 8. OS/390 9. AS/400 **Note:** Not all these platforms are currently supported.
Exception type (optional)	Value of the thrown MQSeries Workflow exception.
System log ID	Identifier of the corresponding system log message entry. For a description of these entries, see "System log record entries" on page 218.
Message type (optional)	Identifier of the MQSeries Workflow message related to this event.
Detection time	The date and time that the error was detected.
Error file (optional)	Identifies the file within an MQSeries Workflow component where the error was detected.

Table 100. Error log record entries (continued)

Name	Description
Error line (optional)	Identifies the line in the file within an MQSeries Workflow component where the error was detected.
Error description	Summary of the error in the form of an exception notice.

System log record entries

System log records are created for all system events sent or received by the administration server. The layout of system log records is described below.

Table 101. System log record entries

Name	Description
Object ID	Uniquely identifies each system log entry.
Creation time	Specifies the time when the system log entry was created in the database.
System name	The system name.
System group	Contains the system group name of which the system is a member.
Message number	Identifier of the MQSeries Workflow system log message. It is coded description of the event that occurred.
Severity	Identifies the severity associated with the message: 0 (I): Information 4 (W): Warning 8 (E): Error
Message class	Identifies the class to which this message belongs: `SERVER` — Server events `USER` — User events `SYSTEM` — System events `PEA` — Program execution agent events `MQ` — Message queuing events `DB` — Database events
Event identifier (optional)	A string used to group the entries into related events. This could be a server name, a session ID, or a user defined string.
Message type (optional)	Identifier of the MQSeries Workflow message related to this event.
Error flag	If this flag is set to `true`, it indicates that error information has been written to the error log.
Server kind (optional)	If the message belongs to the class `SERVER`, this entry Identifies the server type for which the system log entry was created. Otherwise it contains `NULL`.
Server state (optional)	Identifies the server's state when the system log entry was created if the message belongs to the class `SERVER`. Otherwise it contains `NULL`.
UserID (optional)	If set, the name of the user whose request caused the syslog entry.
Exception (optional)	Contains the enumeration value for the exception thrown by an MQSeries Workflow component.
Message parameters (optional)	Contains additional parameters which are used to replace real message text.

Compact error reports

If an error can not be reported to the administration server, a compact error report is written to the error data set `FMCERR`*xx* of the server instance concerned. The following shows how a typical compact error reports may appear:

```
+-----------------------------------------------------------------+
|                 MQSeries Workflow 3.3 Error Report              |
|                                                                 |
|                                                                 |
|Report creation = 12/15/00 02:50:42 PM                           |
|FmcExtException, MsgID=12120, MsgParam=ADMINSVR, S1, SG34, , , , , , |
|Origin=File=FMC.A321COG1.CXX#(FMCAARR), Line=641,                |
|Function=FmcAdminReqRouter::InitExtensions()                     |
|RRSAF return code: 12; SQL reason code: f30093                   |
|                                                                 |
+-----------------------------------------------------------------+
```

Appendix M. Audit Trail

This chapter describes the audit trail and the command-line application used to clean up records from the audit trail held in the MQSeries Workflow database.

What is the audit trail?

When a process instance is executed, MQSeries Workflow writes information about each significant event into an audit trail. The audit trail is managed in the MQSeries Workflow relational database.

Whether an audit trail is written at all, and if so, how much is written into the audit trail, is controlled by the ***AUDIT_TO_DB*** or the ***AUDIT_TO_MQ*** option of the process instance. The ***AUDIT_TO_DB*** option replaces the previously used ***AUDIT*** option. With the ***AUDIT_TO_MQ*** option, you can write audit events as XML messages to an MQSeries queue.

The audit options are set during Buildtime and can take any one of the following values:

Audit trail OFF
Specifies that an audit trail is not created when an instance of the process runs.

Audit trail Condensed
Specifies that only important event information, for example, who issued terminate and resume requests for an activity instance, is written to the audit trail.

Audit trail Full
Specifies that all audit information for an event is written to the audit trail.

Audit trail Filter
Specifies which audit event information is written. You can specify these events by entering a string of audit event numbers, as listed in Table 103 on page 223.

For further information regarding the ***AUDIT*** option and how to set it, refer to the *IBM MQSeries Workflow: Getting Started with Buildtime* book.

Audit trail specifications are inherited from the domain level of MQSeries Workflow through the system group to the system and down to the process template. Each specification can be overwritten on a lower level.

The events are written into the audit trail of the MQSeries Workflow system on which the process instance was started.

Process instances are identified by the process instance name or the process instance identifier. Both are written into the audit trail. Object identifiers are stored in their external character format.

Applications that use standard SQL can be written to access the audit trail. Care must be taken to avoid any unintentional changes to the audit trail.

Each audit trail record is associated with a timestamp. This timestamp reflects the date and time the audit trail record was written. As such, it is filled by the underlying relational database management system. Since it is not guaranteed that all timestamps are unique, the sequence in which audit trail records with the same timestamp are retrieved is random.

Table 102 shows the structure of the audit trail in the relational database:

Table 102. Audit trail record layout

Field Name	Column name of table fmc.audit_trail	Type	Explanation
Timestamp	CREATED	TIMESTAMP Mandatory	Date and time the audit trail record is written.
Event	EVENT	INTEGER Mandatory	Type of event as indicated in Table 103 on page 223.
Process Name	PROCESS_NAME	VARCHAR (63) Mandatory	Name of the process instance.
Process Identifier	PROCESS_ID	IDENTIFIER Mandatory	Object identifier of the process instance.
Top-level Name	TOP_LVL_PROC_NAME	VARCHAR (63) Mandatory	Name of the top-level process instance if the process instance is executing as subprocess, or the same as in process name if the process instance is a top-level process instance.
Top-level Identifier	TOP_LVL_PROC_ID	IDENTIFIER Mandatory	Object identifier of the top-level process instance if the process is executing as subprocess, or the same as in process identifier if the process instance is a top-level process instance.
Parent Process Name	PARENT_PROC_NAME	VARCHAR (63) Optional	Name of the parent process instance if the process instance is executing as a subprocess.
Parent Process Identifier	PARENT_PROC_ID	IDENTIFIER Optional	Object identifier of the parent process instance if the process instance is executing as a subprocess.
Process Model Name	PROC_TEMPL_NAME	VARCHAR(32) Mandatory	Name of the process model.
Process Model Valid from Date	TEMPL_VALID_FROM	TIMESTAMP Optional	The date the associated process model becomes valid.
Block Names	BLOCK_NAMES	VARCHAR(254) Optional	The concatenated names of all blocks in which the activity is contained in. The various names are separated by a dot.
User ID	USER_NAME	VARCHAR(32) Optional	ID of the user associated with the event that caused the audit trail to be written. If the audit trail record is written by the MQSeries Workflow system, this field is not filled.
Second user ID	SECOND_USER_NAME	VARCHAR(32) Optional	ID of the second user associated with the event that caused the audit trail to be written.

Table 102. Audit trail record layout (continued)

Field Name	Column name of table fmc.audit_trail	Type	Explanation
Activity Name	ACTIVITY_NAME	VARCHAR(32) Optional	If the audit trail entry is associated with an activity, the field contains the name of the activity. If the audit trail entry is associated with a control connector, the field contains the name of the activity that is the source of the control connector.
Activity Type	ACTIVITY_TYPE	INTEGER Optional	If the audit trail record is written for an activity, the field contains the type of the activity as defined in Table 104 on page 226.
Activity Status	ACTIVITY_STATE	INTEGER Optional	If the audit trail record is written for an event associated with an activity, the field contains the status of the activity encoded as shown in Table 105 on page 226.
Second Activity Name	SECOND_ACT_NAME	VARCHAR(32) Optional	If the audit trail is written for an event associated with a control connector, the field contains the name of the target activity.
Command Parameters	COMMAND_ PARAMETERS	VARCHAR(1024) Optional	If the event is the start of a program activity, the field contains the actual parameters passed when invoking the program.
Associated Object	ASSOCIATED_OBJECT	IDENTIFIER Optional	Contains the identifier of the object associated with the event. Can be used to locate the object in the MQSeries Workflow database.
Object Description	OBJECT_DESCRIPTION	VARCHAR(254) Optional	Contains the description of the object associated with the event.
Program Name	PROGRAM_NAME	VARCHAR(32) Optional	If the event is the start of a program activity, the field contains the name of the program.
Activity Return Code	ACTIVITY_RC	LONG Optional	Return code of the activity.

The contents of each audit trail record depends on the event. Table 103 shows the contents of each field.

The audit trail level field indicates which events are written to the audit trail when either full or condensed audit trailing is active. If full audit trailing is active, all audit trail records are written. If condensed audit trailing is active, only the events listed with C are written.

Table 103. Audit trail record contents

Code	Audit Trail Level	User ID	Second User ID	Associated object	Event
21000	C	Process starter		Process instance	Process started
21001		Issuer of suspend command		Process instance	Process suspended

Table 103. Audit trail record contents (continued)

Code	Audit Trail Level	User ID	Second User ID	Associated object	Event
21002		Issuer of resume command		Process instance	Process resumed
21003				Process instance	Process notification sent
21004	C			Process instance	Process ended normally
21005	C			Process instance	Process terminated
21006	C			Activity instance	Activity ready
21007	C	User on whose behalf the activity is started		Activity instance	Activity started
21008				Activity instance	First activity notification sent
21009		Target of transfer	Source of transfer	Work item	Work item transferred
21010			User for whom work item is created	Work item	Work item created
21011	C	User on whose behalf the activity was executed		Activity instance	Activity ended normally
21012	C	Issuer of force-finish command		Activity instance	Activity force-finished
21013		Issuer of restart command		Activity instance	Activity restarted
21014	C	Issuer of finish command		Activity instance	Activity exited manually
21015					Block started
21016					Block ended
21017		Issuer of create command		Process instance	Process created
21018	C	Issuer of create and start command		Process instance	Process created and started
21019		Issuer of restart command		Process instance	Process restarted
21020		Issuer of delete command		Process instance	Process deleted
21021	C	Issuer of cancel checkout command	.	Activity instance	Checkout of activity canceled
21022	C	Issuer of checkout command		Activity instance	Checkout activity
21023		Issuer of checkin command		Activity instance	Checkin activity
21024				Activity instance	Second notification for activity sent
21025	C			Process instance	Process ended normally and deleted
21026	C	Issuer of terminate command		Process instance	Process terminated and deleted

Table 103. Audit trail record contents (continued)

Code	Audit Trail Level	User ID	Second User ID	Associated object	Event
21027	C	Issuer of terminate command		Activity Instance	Activity terminated
21028	C	Issuer of create with start time command		Process instance	Process created with future start time
21030		Issuer of delete work item command	Owner of work item	Work item	Work item deleted
21031	C	Issuer of force restart work item command		Activity instance	Activity force restarted
21032		User on whose behalf the activity was executed		Activity instance	Activity implementation completed
21034					Control connector evaluated to true
21037	C	Issuer of suspend command		Process instance	The specified user has issued a suspend process command.
21038	C	Issuer of terminate process command		Process instance	The specified user has issued a terminate process command.
21039	C	Issuer of execute command		Process instance	The specified user has issued an execute command.
21040	C	Issuer of resume command		Process instance	The specified user has issued a resume process command.
21041		User who has processed the activity		Activity instance	Activity automatically restarted as exit condition evaluated to false.
21044	C	Issuer of terminate activity request		Activity instance	The specified user has issued a terminate process activity command.
21052	C	Issuer of import request		Process instance	Process instance imported
21053	C	Issuer of import request		Activity instance	Activity instance imported
21056				Process instance	Block ended and loop back to the beginning because the exit condition failed.
21080	C			Activity instance	Activity state is set to inError as the activity imlementation failed.
21081	C			Activity instance	Activity expired

The following table shows the encoding for activity types.

Table 104. Audit trail activity type encoding

Code	Activity Type
21100	Program activity
21101	Process activity
21102	Block activity
21103	Information activity
21104	Bundle activity

The following table shows the encoding for activity states. If an event is associated with a state change, the target state is recorded in the audit trail record.

Table 105. Audit trail activity state encoding

Code	Activity State
21200	Ready
21201	Running
21202	Finished
21203	CheckedOut
21204	Force-Finished
21205	Terminated
21206	Suspended
21207	InError
21208	Executed
21209	Skipped
21210	Deleted
21211	Suspending
21212	Terminating
21213	Expired

How to analyze the audit trail

All information about the audit trail is stored in the DB2 relational database table **fmc.audit_trail**. For a detailed desription of the audit trail structure, refer to Table 102 on page 222.

There are several SQL queries you can perform to analyze the audit trail. To perform these queries, you need a DB2 client that can access the MQ Workflow database as, for example, the administration server.

To analyze the audit trail you can use SPUFI on S/390 or a DB2 client (DRDA) on the distributed side that is connected to the DB2 subsystem on the host.

To establish a connection with the MQ Workflow database, enter the following command:

`db2 connect to` *`name`* where *name* is the name of your database.

When you have established the database connection, you can run any query for the specified database to get the information you need. Following you find several examples for queries that can be helpful for analyzing the audit trail.

The following query provides information about which process models have been executed and how often they have been executed:

```
SELECT PROC_TEMPL_NAME, COUNT(*) AS COUNT
  FROM SystemGroupPrefix.AUDIT_TRAIL
  WHERE EVENT IN (21000, 21018)
  GROUP BY PROC_TEMPL_NAME
  WITH UR;
```

The following query provides information about which programs have been executed and how often they have been executed.

```
SELECT PROGRAM_NAME, COUNT(*) AS COUNT
  FROM SystemGroupPrefix.AUDIT_TRAIL
  WHERE EVENT = 21007
    AND PROGRAM_NAME IS NOT NULL
  GROUP BY PROGRAM_NAME
  WITH UR;
```

The following query provides information about the number of work items created within a specific time frame. In the example, the query returns all work items that were created in January 2000.

```
SELECT COUNT(*) AS COUNT
  FROM SystemGroupPrefix.AUDIT_TRAIL
  WHERE EVENT = 21010
    AND CREATED BETWEEN '2000-01-01-00.00.00.000000'
    AND '2000-01-31-23.59.59.999999'
    WITH UR;
```

The following query requires FULL audit trail. It provides information about which blocks have been restarted and how often they have been restarted.

```
SELECT PROC_TEMPL_NAME, BLOCK_NAMES, COUNT(*) AS COUNT
  FROM SystemGroupPrefix.AUDIT_TRAIL
  WHERE EVENT = 21056
  GROUP BY PROC_TEMPL_NAME, BLOCK_NAMES
  WITH UR;
```

To end the connection with the database, enter the following command:

`db2 disconnect` *`name`* where *name* is the name of your database.

Appendix N. Migrating from a previous release

If you want to migrate from Version 3.1, contact the Workflow service team.

In order to migrate your existing MQSeries Workflow for OS/390 Version 3.2 systems, you must create new MQSeries Workflow for z/OS systems and migrate the Runtime database. Therefore some of the steps executed during the creation of a new system will also be performed in order to migrate your existing systems. Before starting to migrate your existing MQSeries Workflow for z/OS systems, you shold plan your MQSeries Workflow for z/OS identifiers.

Note: In this chapter, the term **existing system** (or installation) refers to the Version 3.2 installation that you want to migrate from. The term **new system** (or installation) refers to the Version 3.3 installation you want to create.

Planning your migration

If you have more than one system group, you should migrate one system group completely before you start to migrate the next system group. To migrate a system group, you will keep your existing database, and will install and configure new systems to replace the existing systems in your system group. The following instructions often refer you to the standard customization descriptions in "Part 1. Customization" on page 1.

Since the Runtime database belongs to a system group, you must migrate all the systems in a system group during the production outage time for the system group.

Migrating an existing MQSeries Workflow for OS/390 system group requires the following steps:

1. "Decide your new MQSeries Workflow for z/OS identifiers" on page 230
2. "Before starting migration" on page 231, which includes:
   a. Checking that you have the prerequisites and authorities to be able to perform migration.
   b. Install the new release of MQSeries Workflow for z/OS as described in the *MQSeries Workflow for z/OS: Program Directory* using a new *InstHLQ*.
3. "Phase 1: Existing system group is functional" on page 232, consists of:
   a. "Premigration for each system in the system group" on page 232
4. "Phase 2: Production outage" on page 233, consists of:
   a. "Stop your existing system group" on page 233
   b. "Migrate each OS/390 image" on page 234
   c. "Migrate the databases for the system group" on page 234
   d. "Migrate and verify each system in the system group" on page 235
5. "Phase 3: New system group is functional (except program execution)" on page 235, consists of:
   a. "Program execution migration for each system in the system group" on page 235
6. "Phase 4: New system group is fully-functional" on page 235, consists of:
   a. "LAN client migration (optional)" on page 235

Decide your new MQSeries Workflow for z/OS identifiers

You will use SMP/E to install the product into the libraries image from the tape to the location that is specified by the MQSeries Workflow for z/OS installation high level qualifier *InstHLQ*. Each time that you want to migrate an existing MQSeries Workflow for OS/390 system you must specify a new customization high level qualifier *CustHLQ*. It determines where the new MQSeries Workflow for z/OS system-specific files are copied and customized.

Many identifiers are taken from your existing MQSeries Workflow for OS/390 setup. You only have to plan the identifiers listed in Table 106.

Migration identifiers for each Workflow system

The following identifiers are required for migrating from an existing MQSeries Workflow for OS/390 system. You shold copy and complete this table for each MQSeries Workflow for OS/390 system you want to migrate. Some of the identifiers decided here will be entered into the customization parameter file, and will be automatically substituted in the customization jobs.

Table 106. Migration identifiers for each Workflow system

Parameter	Your value	Name in the migration parameter files	Description
InstHLQ		MQWFIHLQ	MQSeries Workflow for z/OS installation high level qualifier. This qualifier is determined when the product image is installed from tape, this is described in *MQSeries Workflow for z/OS: Program Directory*. **Note:** This should be different to the installation HLQ for your existing installation.
OldCustHLQ		MQWFMHLQ	The data set containing the MQSeries Workflow for OS/390 customization parameter file FMCHECIF for your **existing** system that is to be migrated. **Note:** This is normally the *CustHLQ* for your existing system.
CustHLQ		MQWFCHLQ	The high level qualifier for the **new** MQSeries Workflow for z/OS system you want to customize. This should be different to the customization HLQ for your existing installation.
CICSGroup		CICSGRPN	CICS group name used for program execution server invocation. **Note:** If you want to use the same CICS system for your new MQSeries Workflow for z/OS system, you must choose a group name that is different from the one you used for your existing MQSeries Workflow for OS/390 system.
ISPFHLQ		—	ISPF installation high level qualifier.
ISPFLAN		—	ISPF language, for example, ENU for U.S. English.
DB2BPIDX		—	DB2 buffer pool name for the new indexes that will be created during the migration of an existing MQSeries Workflow for OS/390 system group.

Before starting migration

On each system in the system group that you want to migrate, ensure the following:

1. You have OS/390 Version 2.7 or higher installed.
2. You have MQSeries for OS/390 Version 2.1 or higher installed, and you will be using the same queue manager in the same MQSeries cluster.
3. You have DB2 for OS/390 Version 5.1 or higher installed, and you will be using the same subsystem and databases as for your existing system.
4. To perform migration, you must have DB2 SYSADM rights.
5. You should have configured the DB2 for OS/390 CLI support (FMID JOB5517) as described in DB2 for OS/390 V5 Call Level Interface Guide and Reference.
6. IBM Resource Access Control Facility (RACF) authority to alter the MQSeries Workflow for z/OS 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.
7. RACF authority to alter `PROCLIB` and `PARMLIB`.
8. Install the new release of MQSeries Workflow for z/OS as described in the *MQSeries Workflow for z/OS: Program Directory* using the new *InstHLQ*.
9. Perform "After installing MQSeries Workflow for z/OS" on page 19, for each OS/390 image.
10. The load library *InstHLQ*`.SFMCLINK` must be Advanced Program Facility (APF) authorized.
11. You should have configured the Resource Recovery Service (RRS) as described in OS/390 MVS Programming: Resource Recovery.

Migrate an existing MQSeries Workflow for OS/390 system group

This section describes the migration tasks necessary to migrate an existing MQSeries Workflow for OS/390 Version 3.2 system group to Version 3.3. To migrate a system group, you must migrate the Runtime database, and create new systems to replace the existing systems.

Migration phases

The complete migration of a system group can be divided into four time frames:

Phase 1: Existing system group is functional
Some migration actions can be performed while your existing system group remains functional.

Phase 2: Production outage
When neither your existing systems nor your new systems are functional. During this phase, you migrate the system group's databases, and migrate and verify all the systems in the system group. During this phase you can switch back to the old system group by restoring the old database.

Note: You must not run non-migrated servers with the migrated database.

Phase 3: New system group is functional (except program execution)
When the new system is functional, but you are not able to invoke CICS

and/or IMS programs. During this phase, you enable the CICS and IMS APIs, and verify the program execution migration.

Phase 4: New system group is fully-functional
When the new system group is fully-functional, you can start to migrate your LAN clients.

Phase 1: Existing system group is functional

You can perform "Premigration for each system in the system group" while your existing system group is still running.

Premigration for each system in the system group

Before starting the migration, you must perform the premigration tasks, described in Table 107, for each system in the system group.

These tasks create the necessary libraries, and resources, and copy files from the installation image (*InstHLQ*) to the location for the new system that is to be customized (*CustHLQ*). Some configuration parameter values are also copied from the configuration parameter file for your existing system (*OldCustHLQ*) to automatically generate the migration jobs.

Table 107. Premigration for each system in the system group

Step number	Description	Action	Verification
1	Allocate data sets.	Perform the actions described in "Data set allocation" on page 23.	
2	Copy customization templates.	1. Copy the JCL *InstHLQ*.SFMCCNTL(FMCHJCCT) 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.
3	Precustomize the customization parameter file with some of the values from the existing system that you want to migrate.	1. Copy the job *CustHLQ*.SFMCCNTL (FMCHJCUM) to a private partioned data set. 2. Edit your copy of FMCHJCUM as described in the comment header. **Note:** This job will read values from your existing configuration parameter file *OldCustHLQ*.SFMCDATA(FMCHECIF). 3. Submit your copy of FMCHJCUM. 4. Check the job output (step IKJEFT01 / DD statement SYSTSPRT) for error messages. This program replaces a subset of the customization parameter of the member *CustHLQ*.SFMCDATA(FMCHECIF) with the values that were defined for your existing system. **Note:** All customization parameters which are replaced in this step are marked with the uppercase character 'C'. The following is an example for the system group parameter after replacing with the value SG1: `'MQWFSGNM' ='SG1'C`	

Table 107. Premigration for each system in the system group (continued)

Step number	Description	Action	Verification
4	Edit the customization parameter file.	Edit the customization parameter template member *CustHLQ*.SFMCDATA(FMCHECIF), and enter your values from the tables in "Planning your migration" on page 229, as described in the comment sections of the file. **Note:** Some of the parameters were already replaced with values from your existing system during the previous step, these are marked with the uppercase character 'C'. This file is described in "Customization parameter file for a primary system" on page 191. From now on, this member will contain your customization parameters. This member is used as an input file for the generation process in the next step.	
5	Generate all the jobs necessary to customize this system.	1. Copy the job *CustHLQ*.SFMCCNTL(FMCHJCUS) to a private partitioned data set. 2. Edit your copy of FMCHJCUS as described in the comment header. **Note:** This job will read values from your new configuration parameter file *CustHLQ*.SFMCDATA(FMCHECIF). 3. Submit your copy of FMCHJCUS. 4. Check the job output (step IKJEFT01 / DD statement SYSTSPRT) for error messages. The program performs some syntax checking on the length and value of the variables you specified in the file *CustHLQ*.SFMCDATA(FMCHECIF). The program then substitutes your values for variables in the customization files. Some PROCLIB and PARMLIB members are also copied with new names to the library *CustHLQ*.GENPROC and *CustHLQ*.GENPARM.	
6	Define the new MQSeries resources.	Submit the job *CustHLQ*.SFMCCNTL(FMCHJMMQ)	rc=0
7	Update resources for trace.	Perform the actions described in "Trace customization" on page 30.	

When you have completed this stage, the JCL files that are required in the following sections will contain all the customization parameters that you decided in "Planning your migration" on page 229, and any that are taken from your existing system.

Phase 2: Production outage

During the production outage phase, you must perform the following tasks:

1. "Stop your existing system group"
2. "Migrate each OS/390 image" on page 234
3. "Migrate the databases for the system group" on page 234
4. "Migrate and verify each system in the system group" on page 235

Stop your existing system group

This phase starts by stopping all MQSeries Workflow for OS/390 components, as described in Table 108 on page 234.

Table 108. Stop your existing system group

Step number	Description	Action
1	Stop clients and servers.	1. Stop all clients that are connected to MQSeries Workflow for OS/390 systems in the system group. 2. Verify that the queues used by MQSeries Workflow for OS/390 systems are empty. **Note:** You can tell this from the current queue depth field. 3. Use the administration console to stop all MQSeries Workflow for OS/390 systems in the system group.

Migrate each OS/390 image

Table 109 describes how to migrate the MMS message catalog and component trace. This must be done for each OS/390 image affected by the migration.

Table 109. Migrate each OS/390 image

Step number	Description	Action
1	Deactivate the old MMS catalog.	Enter the command: `SET MMS=no`
2	Update the MVS Message Services (MMS) message catalog.	Perform the actions described in "Create the MMS message catalogs" on page 19.
3	Enable the TRACE system address space to access the new MQSeries Workflow start/stop exit routine.	Concatenate the MQSeries Workflow for z/OS LPA library *InstHLQ*.SFMCLPA to your LPA library concatenation list.

Migrate the databases for the system group

To migrate the existing databases for a system group, you must perform the actions described in Table 110, using the *CustHLQ* value for your primary system.

Table 110. Migrate the databases for the system group

Step number	Description	Action	Verification
1	Back up the Workflow databases.	Make a back up of your existing Workflow databases by exporting the DDL and unloading the tables from the MQSeries Workflow for OS/390 databases.	
2	Migrate the Workflow Runtime database.	To migrate your existing Workflow Runtime database: 1. Edit the JCL *CustHLQ*.SFMCCNTL (FMCHJMDC) as described in the comment header. 2. Submit the JCL *CustHLQ*.SFMCCNTL (FMCHJMDC)	rc=0. This program performs the complete migration including schemata changes and necessary changes to the database content.
3	Bind the new Workflow Runtime database packages.	Submit JCL *CustHLQ*.SFMCCNTL(FMCHJBDB).	rc=4 can be accepted.
4	Bind the new PES directory database packages.	Submit JCL *CustHLQ*.SFMCCNTL(FMCHJBPD). **Note:** The contents of the PES directory will be migrated.	rc=0.

Table 110. Migrate the databases for the system group (continued)

Step number	Description	Action	Verification
5	Bind the new PES mapping database packages.	Submit JCL *CustHLQ*.SFMCCNTL(FMCHJBMA).	rc=0.

Migrate and verify each system in the system group

To migrate and verify each system, perform the actions described in Table 111 for each system in the system group.

Table 111. Migrate and verify each system in the system group

Step number	Description	Action
1	Migrate the system.	Perform the actions described in "Workflow server customization" on page 33.
2	Verify the system.	"System customization verification" on page 40.

Phase 3: New system group is functional (except program execution)

In this phase, you perform the program execution migration.

Program execution migration for each system in the system group

You must migrate and verify the program execution for each system in the system group, as described in Table 112.

Table 112. Program execution migration for each system

Step number	Description	Action
1	Enable CICS API support.	If the system uses the CICS API, perform the actions described in "CICS API support customization" on page 31.
2	Enable IMS API support (make Workflow DLLs available to IMS).	If the system uses the IMS API, make sure that all members *InstHLQ*.SFMCLOAD(FMCH3xxx) (with the prefix "FMCH3") are in your IMS PGMLIB library.
3	Enable Java API support.	If the system uses the Java API, perform the actions described in "Customize Java-API support" on page 36.
4	Verify program execution.	To verify program execution, perform the following: 1. "Configure program execution samples" on page 52 2. "Verify program execution samples" on page 52

Phase 4: New system group is fully-functional

By the time you reach this phase, your migrated system group (and the systems in it) should be fully-functional. If required, you can proceed with "LAN client migration (optional)".

LAN client migration (optional)

Your existing LAN clients should work with the new servers, so migrating the clients is only necessary if you require the new or improved functionality that the

new client software provides. To migrate a LAN client, and verify that it works with your migrated Workflow system, perform the actions described in “LAN client customization” on page 33.

Appendix O. 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, other countries, or both:

AIX	IBM	OS/390
AS/400	IMS	RACF
CICS	IMS/ESA	VTAM
CICS/ESA	MQSeries	WebSphere
DB2	MVS	z/OS
DB2 Universal Database	MVS/ESA	

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of the Open Group in the United States and other countries.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

Glossary

This glossary defines terms and abbreviations used in this and other MQSeries Workflow for z/OS 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 server. The MQSeries Workflow component that performs administration functions within an MQSeries Workflow system. Administration server commands that are issued on the system console are passed onto a particular administration server. It accepts commands for starting and stopping systems and servers, it so allows the number of server instances to be queried.

administration server ID. This name is used when issuing administration server 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 z/OS*, 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 151.

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.

audit trail. When a process instance is executed, MQSeries Workflow writes information about each significant event into an audit trail. For more details, see "Appendix M. Audit Trail" on page 221.

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. Deines 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 209.

cleanup 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 WLM does not actively manage the system. This mode is not supported by MQSeries Workflow. 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 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 the 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*.

H

hold queue. Messages that cannot be delivered are placed on a hold queue. How to display, replay, or delete these messages is described in "Hold queue commands" on page 94.

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 user ID 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 z/OS is named 'DEFAULT'.

MDL. See *mapping definition language*.

message queuing. A communication technique provided my 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 refering to the MQSeries Workflow product family.

MQSeries Workflow for z/OS. This product; (previously known as MQSeries Workflow for OS/390) 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.

N

notification exit. A notification exit is the executable that is called by the program execution server when it is about to invoke a program, when it has successfully invoked a program, or when an error has occurred during program invocation.

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 z/OS 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 z/OS 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 147.

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 cleanup 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 z/OS handles two types of user IDs, (1) MQSeries Workflow user IDs. (2) *Execution user ID*s.

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 defintion*. For more details see "What is OS/390 Workload Manager?" on page 145.

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.

Z

z/OS. Since the new version of OS/390 is named z/OS, the new version of MQSeries Workflow for OS/390 is named MQSeries Workflow for z/OS.

Bibliography

To order any of the following publications, contact your IBM representative or IBM branch office.

MQSeries Workflow for z/OS publications

This section lists the publications included in the MQSeries Workflow for z/OS library.

- *MQSeries Workflow for z/OS: Customization and Administration*, SC33-7030, explains how to customize and administer an MQSeries Workflow for z/OS system.
- *MQSeries Workflow for z/OS: Programming*, SC33-7031, explains the C, C++, Java, and Cobol application programming interfaces (APIs), and the program exits.
- *MQSeries Workflow for z/OS: Messages and Codes*, SC33-7032, explains the MQSeries Workflow for z/OS system messages and codes.
- *MQSeries Workflow for z/OS: Program Directory*, GI10-0483, explains how to install MQSeries Workflow for z/OS.

MQSeries Workflow publications

This section lists the publications included in the MQSeries Workflow library.

- *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 Queue Manager Clusters*, SC34-5349.
- *MQSeries Clients*, GC22-1632.

Workflow publications

- *Production Workflow — Concepts and Techniques*, Frank Leymann, Dieter Roller, Prentice-Hall, 1999
- *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

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

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

IBM MQSeries Workflow for z/OS
Customization and Administration
Version 3.3

Publication No. SC33-7030-05

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

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. _____

Readers' Comments — We'd Like to Hear from You
SC33-7030-05

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

PLACE
POSTAGE
STAMP
HERE

IBM Deutschland Entwicklung GmbH
Information Development, Dept. 0446
Schoenaicher Strasse 220
71032 Boeblingen
Germany

Fold and Tape | **Please do not staple** | Fold and Tape

SC33-7030-05

Cut or Fold
Along Line

Program Number: 5655-BPM

Printed in Denmark by IBM Danmark A/S

SC33-7030-05

Spine information:

IBM MQSeries Workflow for z/OS **Customization and Administration** Version 3.3