IBM MQSeries Workflow for OS/390

Customization and Administration

Version 3 Release 1

SC33-7030-00

IBM MQSeries Workflow for OS/390

Customization and Administration

Version 3 Release 1

SC33-7030-00

Note!

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

First Edition (March 1999)

This edition applies to Version 3, Release 1, Modification 0 of IBM MQSeries Workflow for OS/390 (product number 5655–A96) and to all subsequent releases and modifications until otherwise indicated in new editions.

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

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

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

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

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

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

Contents

Figures

Tables

About this book

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

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

Who should read this book

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

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

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

How this book is organized

- “Part 1. Customization” on page 1 describes how to customize MQSeries Workflow for OS/390. It contains the following chapters:
  - “Chapter 1. Planning your configuration” on page 3 provides tables to photocopy and complete for use during customization.
  - “Chapter 2. Things that you must do before starting customization” on page 13 describes post-installation tasks that must be performed once, and pre-customization tasks that must be performed each time you want to create a new MQSeries Workflow for OS/390 system.
  - “Chapter 3. Customizing MQSeries Workflow for OS/390” on page 17 guides you through the process necessary to customize an MQSeries Workflow for OS/390 system.
- “Part 2. System administration” on page 43 introduces the concepts and components of system administration in an MQSeries Workflow system and explains how to start and use the MQSeries Workflow for OS/390 administration console. Details regarding error logging and problem determination using a trace facility are also given.
  - “Chapter 4. Introduction to system administration” on page 45 gives an overview of the objects and tasks involved in administrating this product.

- "Chapter 5. Administration console tasks" on page 53 describes the command-driven administration interface that is used to start and stop the system and servers.
- "Chapter 6. Buildtime administration tasks" on page 59 describes the administration tasks connected with the Buildtime tool, and process models.
- "Chapter 7. Program execution" on page 67 covers all tasks relating to the program execution server, such as administering programs, users, mappings, and invocation types.
- "Chapter 8. Performance tuning" on page 81 describes some specific ways to improve system performance.
- "Chapter 9. Problem determination" on page 83 describes solutions to specific problems, and describes how to use the tracing facilities.

- ″Part 3″ contains the following appendixes:
  - "Appendix A. Program Execution Server directory" on page 93 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 C. Program mapping import tool syntax" on page 101 describes the database utility language used to modify the program mapper's database.
  - "Appendix E. FDL code page conversion tool" on page 107 contains details about how convert process model information between different code pages, in case your uploading method fails to preserve your FDL files.
  - "Appendix F. FDL import/export tool" on page 109 provides the syntax, options, and examples of the tool for importing and exporting process model information in the FDL file format.
  - "Appendix D. Naming and code page restrictions" on page 105 describes the restrictions for naming OS/390 objects in theMQSeries Workflow Buildtime process model.
  - "Appendix G. Customization parameter file" on page 115 contains a copy of the template file that must be customized each time a new Workflow system is generated.
  - "Appendix H. Machine profile" on page 117 describes contents of the profile file that determines the behavior of servers and tools when they are started.
  - "Appendix I. Environment variable file" on page 119 describes contents of the environment variable file that determines the behavior of servers and tools when they are started.
  - 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 125.

How to send your comments

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

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

Notices

Trademarks

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

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.

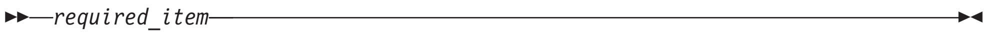
  The ──▸ symbol indicates that the statement syntax is continued on the next line.

  The ▸── symbol indicates that a statement is continued from the previous line.

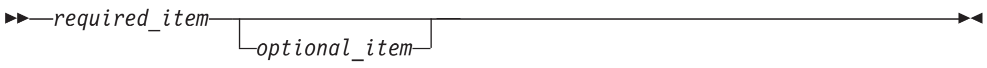
The ───►◄ symbol indicates the end of a statement.

Diagrams of syntactical units other than complete statements start with the ►─── symbol and end with the ───► symbol.

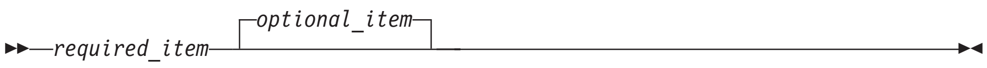
- Required items appear on the horizontal line (the main path).



- Optional items normally appear below the main path.

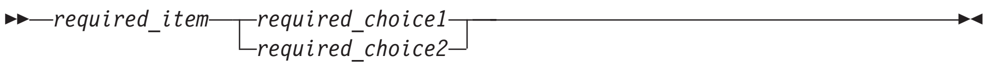


  If an optional item appears above the main path, that item has no effect on the execution of the statement and is used only for readability.

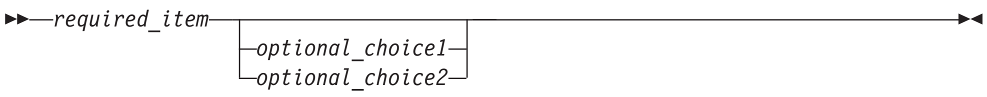


- If you can choose from two or more items, they appear vertically, in a stack.

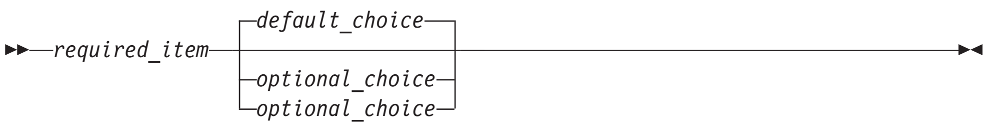
  If you *must* choose one of the items, one item of the stack appears on the main path.



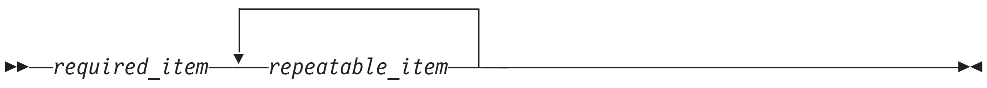
  If choosing one of the items is optional, the entire stack appears below the main path.



  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.

```
                  ┌─,───────────────┐
                  ▼                 │
►►──required_item───repeatable_item─┴────────────────────────────►◄
```

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

```
                  ┌─(5)─────────────┐
                  ▼                 │
►►──required_item───repeatable_item─┴────────────────────────────►◄
```

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

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

```
├──┤ A Fragment ├──────────────────────────────────────────────┤
```

A Fragment:

```
├──Keyword──value──────────────────────────────────────────────┤
```

Part 1. Customization

Chapter 1. Planning your configuration

Before starting to customize your MQSeries Workflow for OS/390 system, you should plan your configuration. This includes planning the following:

1. "Installation scope identifiers" on page 5
2. "System group scope identifiers" on page 6
3. "System scope identifiers" on page 7
4. "Flags and high level qualifiers" on page 8
5. "Subsystem identifiers" on page 9
6. "Customization identifiers" on page 9
7. "Evaluate database requirements" on page 10

We recommend that you copy and complete the following tables before starting customization. It may be necessary for the information to be agreed and exchanged between the following people:

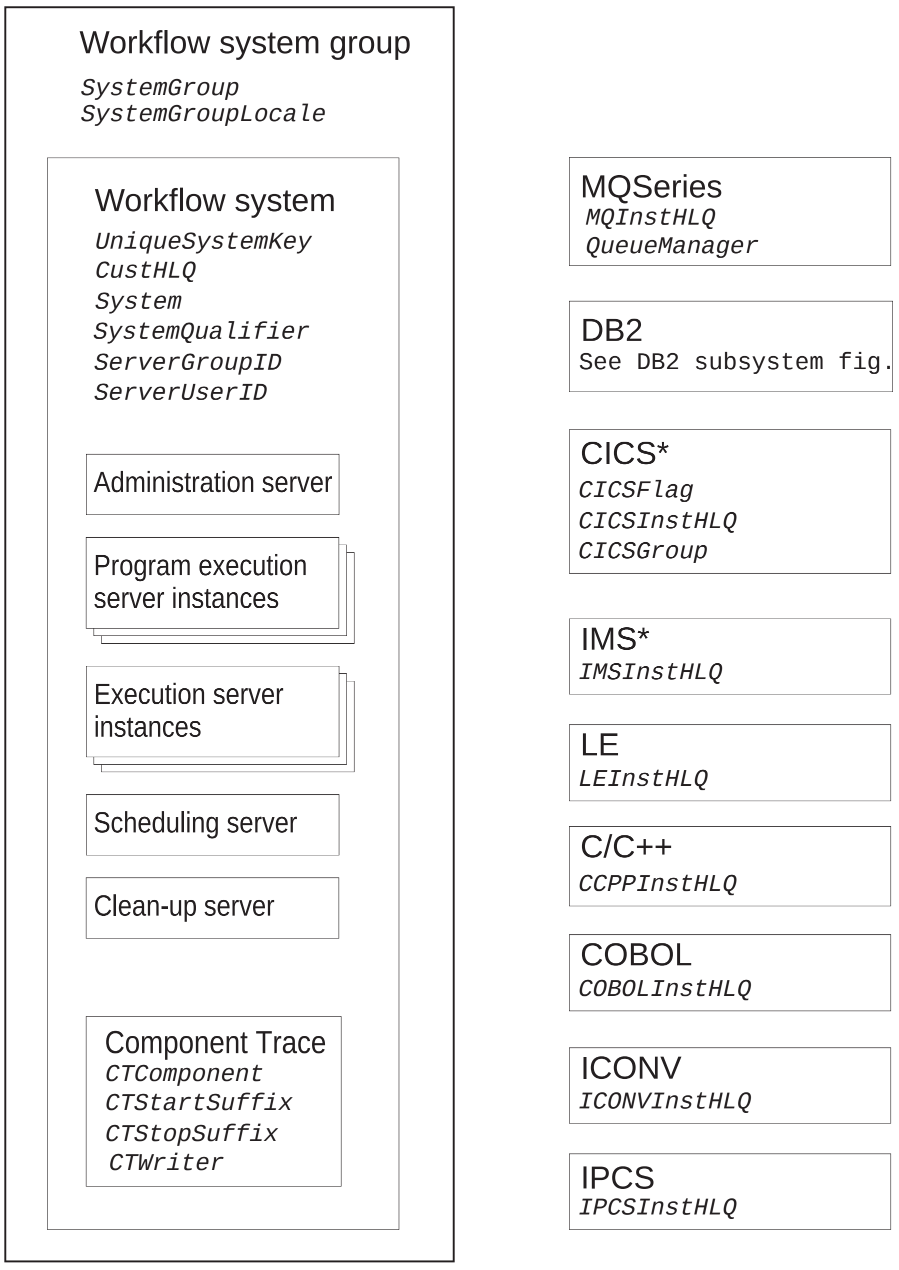
- OS/390 system administrator
- CICS administrator
- IMS administrator
- DB2 administrator
- RACF administrator
- MQSeries administrator
- MQSeries Workflow local area network (LAN) administrator
- MQSeries Workflow for OS/390 administrator

Deciding your MQSeries Workflow for OS/390 identifiers

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

The identifiers decided here will be entered into the customization parameter file during "Pre-customization" on page 15. During "Create input files for customization" on page 16, these parameters are automatically substituted in the customization jobs. The customization parameter file is listed in "Appendix G. Customization parameter file" on page 115.

The main system components and associated customization parameters are illustrated in Figure 1 on page 4 and Figure 2 on page 5.



* CICS, IMS, and COBOL are optional.

Figure 1. Customization parameters for a Workflow system

Multiple Workflow system groups can exist and can share one DB2 subsystem. A Workflow system group contains one system. Each Workflow system consists of servers for administration, program execution, execution, scheduling, and clean-up.

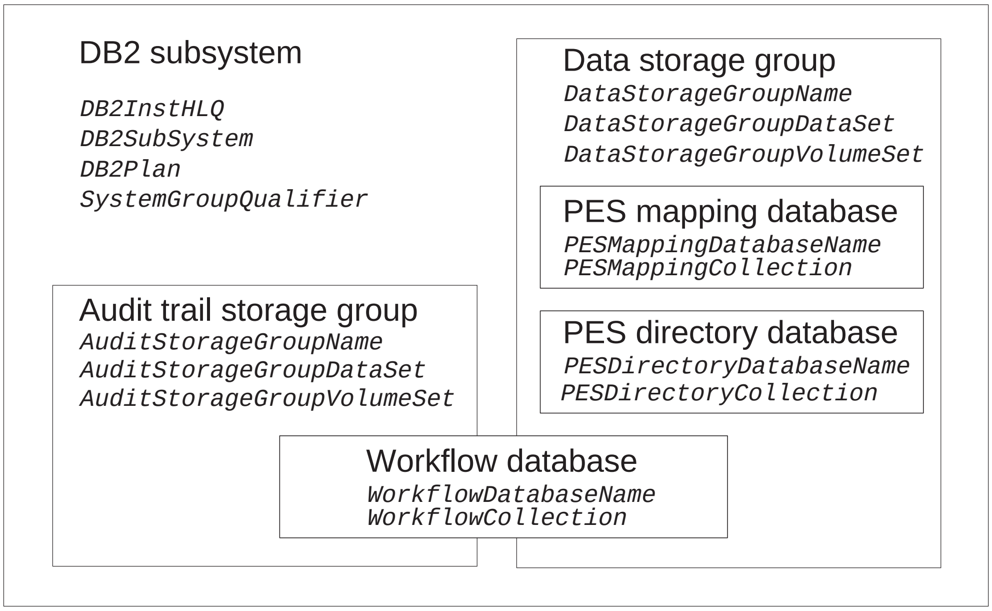


Figure 2. Customization parameters for DB2

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 OS/390 installation high level qualifier. This qualifier is determined when the product image is installed from tape, this is described in *MQSeries Workflow for OS/390: Program Directory*.
CustHLQ		`MQWFCHLQ`	The high level qualifier for the MQSeries Workflow for OS/390 system you want to customize.

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 OS/390 system group name. This name must be unique within your MQSeries Workflow domain.
SystemGroupLocale		MQWFSGLC	MQSeries Workflow for OS/390 system group locale setting. This is used to set the locale for all servers and tools. It selects the correct code page conversion. Use `'C'` for the default on your machine or a specific locale setting, for example, `'De_DE.IBM-273'` for German. The MQSeries Workflow for OS/390 servers and utilities load the active locale from the C environment variable `LC_ALL` to determine to which local code page MQSeries Workflow messages from remote clients should be converted. The `LC_ALL` environment variable is set and propagated to the MQSeries Workflow programs with the LE runtime option `ENVAR`.
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 OS/390 data will be stored. Specify the volume name or `'*'` for SMS managed volumes.
DataStorageGroup DataSetPrefix		DB2STGPW	DB2 storage group data set prefix for run-time data.
DataStorageGroup VolumeSet		DB2STGVW	DB2 storage group volume set for runtime data.
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.
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.

Table 3. System scope identifiers

Parameter	Your value	Name in customization parameter file	Description
UniqueSystemKey		MQWFUKEY	Unique key for an MQSeries Workflow for OS/390 system, may be up to eight uppercase characters long. This is the name given to the Workflow server start job, and must be unique within `SYS1.PROCLIB`. This key is used in the START command to start an administration server on the *System* associated with this key.
SystemQualifier		MQWFSYSP	MQSeries Workflow for OS/390 system qualifier used to prefix MQSeries Workflow for OS/390 object names, for example queue names and profile keys. This identifier may be up to eight uppercase characters long.
System		MQWFSYSN	MQSeries Workflow for OS/390 system name. This name must be unique within the system group. This is the system where the administration server is started when the start administration server command is issued: `START` *`UniqueSystemKey`*`.`*`AdminServerID`*.
ServerUserID		STTSKUID	The server started task RACF user ID used by all MQSeries Workflow for OS/390 servers. This is the default user ID that OS/390 programs will be run under, by the PES, as a result of MQSeries Workflow process activity requests for OS/390 program invocations. This user ID requires `EXECUTE` rights on *DB2Plan*
ServerGroupID		STTSKGRP	The server started task RACF group ID for all MQSeries Workflow for OS/390 servers.
CTComponent		CTRCNAME	CTRACE component name.
CTStartSuffix		CTRCPMS1	CTRACE PARMLIB member suffix (start writer). This value may be in the range 00..99.
CTStopSuffix		CTRCPMS2	CTRACE PARMLIB member suffix (stop writer). This value may be in the range 00..99.
CTWriter		CTRCWPRC	CTRACE writer procedure name. This must not be more than seven characters long.

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 16, 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 "Appendix G. Customization parameter file" on page 115.
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 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 file" on page 119. 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 OS/390.
QueueManager		`MQQMNAME`	Name of the MQSeries queue manager that is to be used by MQSeries Workflow for OS/390. **Note:** If you want to run CICS applications that use the MQSeries Workflow for OS/390 application program interface (API), this must be the same queue manager that is used by CICS.
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 OS/390 administrator. This user ID requires `EXECUTE` rights on *DB2Plan* to be able to execute the tools that update the runtime databases.
MQHostName		The TCP/IP host name of the OS/390 where the queue manager is installed. This value is required during “Customize the MQSeries client connection” on page 26.
applid		This value is required during “Customize CICS EXCI invocation” on page 30 and “Customize program execution server directory” on page 38.
netid		This value is required during “Customize IMS CPIC invocation” on page 34 and “Customize program execution server directory” on page 38.
luname		This value is required during “Customize IMS CPIC invocation” on page 34 and “Customize program execution server directory” on page 38.
CICSBridge InputQueue		This value is required during “Customize MQSeries CICS bridge invocation” on page 31 and “Customize program execution server directory” on page 38.
IMSBridge InputQueue		This value is required during “Customize MQSeries IMS bridge invocation” on page 36 and “Customize program execution server directory” on page 38.

Table 6. Customization identifiers (continued)

Parameter	Your value	Description
XCFGroupName		This value is required during “Customize MQSeries IMS bridge invocation” on page 36. 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 36. 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 36. It represents the MQSeries instance as a member in the XCF group *XCFGroupName*.
PESDirectory SourceFile		Your PES directory is based on a skeleton. After customizing the source file, any new changes will have to be made to your source file. This value is required during “Customize program execution server directory” on page 38.

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 12, 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 clean-up every	7 days
Average small container size	512 bytes
Average large container size	4096 bytes

More detailed database planning (optional)

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

This is a planning phase. You should not modify any of the files mentioned here, copies of these files will be generated during "Pre-customization" on page 15.

The values you decide on must be consistent with your values that you have already planned, especially those in Table 2 on page 6.

1. Print a copy of `*InstHLQ*.SFMCCNTL(FMCHJDBP)` 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 6)	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 12 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)									
			8	8	6	6	6	4	4	4	4	6
Database	-			•								
PESDIRTS	1											•
MAPPING	1											•
TEMPLT32	12		•									
TEMPLT04	12											•
CONTAINR	120										•	
PROCESS	12											•
WORKITEM	72					•						
NTFYITEM	12								•			
ACTWI	20							•				
PROCACT	12									•		
PROGACT	12									•		
BLOCKACT	12									•		
BLOCK	12									•		
STAFF04	16											•
STAFF32	16		•									
MPOOL	1		•									
TOPLGY32	20		•									
TEST	12											•
ADMIN	12											•
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 6.

Chapter 2. Things that you must do before starting customization

Before starting customization, you should check the following:

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

   Note: This manual assumes that you are using RACF for your security. If you are using a different security system, you must apply the equivalent security access controls for your system.
5. RACF authority to alter `PROCLIB` and `PARMLIB`.
6. The load library *InstHLQ*`.SFMCLINK` must be Advanced Program Facility (APF) authorized.
7. You should have configured the Resource Recovery Service (RRS) as described in *OS/390 MVS Programming: Resource Recovery*.
8. Then you are ready to perform "Post-installation" followed by
9. "Pre-customization" on page 15

Post-installation

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

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

Create the MMS message catalogs

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

Table 10. Create MMS message catalogs

Step number	Required or optional	Description	Action	Verification
1	Required	Update parameters in JCLs.	1. Edit *InstHLQ*.`SFMCCNTL(FMCHJMM1)` a. Replace <*MQWFIHLQ*> with your MQSeries Workflow for OS/390 installation high level qualifier, see *InstHLQ* in Table 1 on page 5. b. Replace <*MMSVOL*> with the volume name where the VSAM cluster for the MQSeries Workflow for OS/390 message catalogs should reside. c. Replace <*STORCLAS*> with the storage class of the volume where the VSAM cluster for the MQSeries Workflow for OS/390 message catalogs should reside. 2. Edit *InstHLQ*.`SFMCCNTL(FMCHJMM2)` a. Replace <*MQWFIHLQ*> with your MQSeries Workflow for OS/390 installation high level qualifier. 3. Edit *InstHLQ*.`SFMCPARM(FMCHYMMS)` a. Replace <*MQWFIHLQ*> with your MQSeries Workflow for OS/390 installation high level qualifier.	
2	Required	Create the VSAM clusters.	Submit JCL *InstHLQ*.`SFMCCNTL(FMCHJMM1)`	`rc = 0`
3	Required	Load the input files *InstHLQ*.`SFMCMSG (FMCHMxxx)` to the VSAM clusters.	Submit JCL *InstHLQ*.`SFMCCNTL(FMCHJMM2)`	`rc = 0`
4	Required	Copy MMS PARMLIB member.	Being careful not to overwrite an existing PARMLIB member: Copy *InstHLQ*.`SFMCPARM(FMCHYMMS)` to your system `PARMLIB`.	
5	Required	Rename the MMS PARMLIB member.	Being careful not to overwrite an existing PARMLIB member, rename the MMS PARMLIB member that you copied in step 4 to `MMSLST`*xx*.	
6	Required	Provide RACF profile.	Give the system address space MMS read access to the VSAM clusters created in step 2.	
7	Required	Access the PARMLIB member.	Either • Specify `MMS(`*xx*`)` on the `INIT` statement in `SYS1.PARMLIB(CONSOL`*nn*`)`, or • Issue the operator command `SET MMS=`*xx*	

Copy LPALIB member

To add the MQSeries Workflow for OS/390 LPALIB member to your system you must do the following:

Table 11. Copy LPALIB member

Step number	Optional or required	Description	Action
1	Required	Copy LPALIB member.	Being careful not to overwrite an existing LPALIB member: Copy *InstHLQ*.`SFMCLINK(FMCHXTRC)` to `SYS1.LPALIB`.

Pre-customization

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

Data set allocation

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

Table 12. Data set allocation

Step number	Required or optional	Description	Action	Verification
1	Required	Copy allocation job.	Copy the JCL *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 OS/390 customization high level qualifier, see *CustHLQ* in Table 1 on page 5).	
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` 8. *CustHLQ*.`GENPROC` 9. *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 *CustHLQ*.`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` for the copy step. The JCL also deletes any members in the libraries for the generated PROCLIB and PARMLIB members *CustHLQ*.`GENPROC` and *CustHLQ*.`GENPARM`. `rc=8` can be accepted for the delete step, it indicates that there was nothing to delete.
2	Required	Edit the customization parameter file.	Edit the customization parameter template member *CustHLQ*.`SFMCDATA(FMCHECIF)`, and enter your values from the tables in "Chapter 1. Planning your configuration" on page 3, as described in the comment sections of the file. **Note:** This file is described in "Appendix G. Customization parameter file" on page 115. 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`.	This requires `rc=0`. 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`.

When you have completed this stage, the JCL files that are required in the next chapter will contain all the customization parameters that you determined in "Chapter 1. Planning your configuration" on page 3.

Chapter 3. Customizing MQSeries Workflow for OS/390

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

- "System customization" is required.
- "Verify Workflow client sample application" on page 27 is optional.
- "Program execution customization" on page 29 is optional.

System customization

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

1. "General DB2 customization"
2. "Workflow DB2 customization" on page 18
3. "Program execution server directory DB2 customization" on page 19
4. "Program execution server mapping DB2 customization" on page 21
5. "MQSeries customization" on page 22
6. "OS/390 trace customization" on page 22
7. "CICS API support customization" on page 23
8. "IMS API support customization" on page 24
9. "Workflow server customization" on page 25
10. "LAN client customization" on page 25
11. "System customization verification" on page 26

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

General DB2 customization

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

```
GRANT SYSADM TO DB2AdminUserID
```

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

Table 14. General DB2 customization

Step number	Required or optional	Description	Action	Verification
1	Required	Bind the plan for the DB2 sample application `DSNTEP2`.	Submit JCL *CustHLQ*`.SFMCCNTL(FMCHJBTE)` **Note:** Use your value for *CustHLQ* from Table 1 on page 5.	`rc=0`

Table 14. General DB2 customization (continued)

Step number	Required or optional	Description	Action	Verification
2	Optional	If you want to change the buffer pool names and sizes:	Edit *CustHLQ*.SFMCDB2(FMCHJDBP), and change the VOLUMES parameter as necessary.	
	Required	Define buffer pools.	Submit JCL *CustHLQ*.SFMCCNTL(FMCHJDBP)	rc=0
3	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 buffer pool definitions.	
	Required	Create storage groups.	Submit JCL *CustHLQ*.SFMCCNTL(FMCHJDST)	rc=0

Workflow DB2 customization

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

Table 15. Workflow DB2 customization

Step number	Required or optional	Description	Action	Verification
1	Optional	If you want to change the default buffer pool name or the storage group for the database:	Edit *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*.

Table 15. Workflow DB2 customization (continued)

Step number	Required or optional	Description	Action	Verification
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.
5	Required	Be sure that RRS is active.	If RRS is not active, you can activate it by issuing the command: START RRS	
6	Required	Populate the Workflow database with initial settings.	Submit JCL *CustHLQ*.SFMCCNTL(FMCHJRBS)	rc=0. If you get a non-zero return code, you can roll back the complete action by deleting the contents of the Workflow tables using the job FMCHJEDC. After this step you have to start again with *Step number 5: Populate the Workflow database*.
7	Required	Populate the Workflow database with initial topology settings.	Submit JCL *CustHLQ*.SFMCCNTL(FMCHJRIB)	rc=0
8	Required	Run the DB2 utility RUNSTATS for the Workflow database.	Submit JCL *CustHLQ*.SFMCCNTL(FMCHJRST)	rc=4 can be accepted.
9	Required	Rebind the Workflow packages.	Submit JCL *CustHLQ*.SFMCCNTL(FMCHJBDB)	rc=4 can be accepted.
10	Required	Verify the Workflow database configuration.	Submit JCL *CustHLQ*.SFMCCNTL(FMCHJCCH)	Verify that each SELECT statement returns at least one row of data.

Program execution server directory DB2 customization

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

Table 16. Program execution directory DB2 customization

Step number	Required or optional	Description	Action	Verification
1	Optional	If you want to change the buffer pool names or storage group for the database:	Edit *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
5	Required	Import the PES directory.	Submit JCL *CustHLQ*.SFMCCNTL(FMCHJPIB) **Note:** For subsequent executions of this step, use FMCHJPIC	rc=0
6	Required	Run the DB2 utility RUNSTATS.	Submit JCL *CustHLQ*.SFMCCNTL(FMCHJSPD)	rc=0
7	Required	Rebind the PES directory packages.	Submit JCL *CustHLQ*.SFMCCNTL(FMCHJBPD)	rc=0
8	Required	Verify the PES directory database configuration.	Submit JCL *CustHLQ*.SFMCCNTL(FMCHJCPD)	Verify that the SELECT statement returns at least one row of data.

Program execution server mapping DB2 customization

This customization creates the PES mapping database that is used by the default program mapper. If you do not want to invoke any legacy applications that would require program mapping, you can skip this, and continue customization at "MQSeries customization" on page 22.

Table 17. 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*.
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

MQSeries customization

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

Table 18. 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 OS/390program execution.	Submit JCL *CustHLQ*.`SFMCCNTL(FMCHJPMQ)`	`rc=0`

OS/390 trace customization

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

Table 19. OS/390 trace customization

Step number	Required or optional	Description	Action
1	Required	Look up variable values.	Check Table 3 on page 7, and note the values that you planned for the following identifiers: 1. The two digits *nn* for *CTStartSuffix*. 2. The two digits *mm* for *CTStopSuffix*. 3. The value for *CTWriter*. **Note:** These are the values that you should have assigned to the variables <*CTRCPMS1*>, <*CTRCPMS2*>, and <*CTRCWPRC*> respectively, in file *CustHLQ*.`SFMCDATA(`*`FMCHECIF`*`)`
2	Required	Check system libraries.	Make sure that the system (or the sysplex) does not already contain the members: 1. `SYS1.PARMLIB(CTIFMC`*`nn`*`)`. 2. `SYS1.PARMLIB(CTIFMC`*`mm`*`)`. 3. `SYS1.PROCLIB(`*`CTWriter`*`)`. Where *nn*, *mm*, and *CTWriter* are the values from step 1.
3	Required	Copy PARMLIB members.	1. Copy the Component Trace Start PARMLIB member *CustHLQ*.`GENPARM(CTIFMC`*`nn`*`)` to `SYS1.PARMLIB` 2. Copy the Component Trace Stop PARMLIB member *CustHLQ*.`GENPARM(CTIFMC`*`mm`*`))` to `SYS1.PARMLIB`
4	Required	Copy PROCLIB member	1. Copy the trace writer *CustHLQ*.`GENPROC(`*`CTWriter`*`)` to `SYS1.PROCLIB`

Table 19. OS/390 trace customization (continued)

Step number	Required or optional	Description	Action
5	Required	Create the extended trace output data sets.	1. Edit data set *CustHLQ*.SFMCCNTL(FMCHJCTR) 2. Submit JCL *CustHLQ*.SFMCCNTL(FMCHJCTR)
6	Required	Provide RACF profiles.	Give SYS1.PROCLIB(*CTWriter*) update access to the trace data sets created in step 5.

CICS API support customization

If you want to use the MQSeries Workflow for OS/390 API and trace in CICS, then you must perform this customization. If you only want to use CICS legacy applications, or if you do not want to use CICS at all you can skip this customization, and continue at "IMS API support customization" on page 24.

Before starting this customization, you should ensure that CICS uses the same MQSeries queue manager that the MQSeries Workflow for OS/390 system uses and should perform a CICS shutdown.

Table 20. 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 *SystemQualifier*.SCEESAMP(CEECCSD) 2. Enable the C/C++ feature in CICS. **Note:** A sample that may help you with this task is located in *SystemQualifier*.SCLBSAM(CLB3YCSD)	
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 OS/390 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 9:	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..	

Table 20. CICS API support customization (continued)

Step number	Required or optional	Description	Action	Verification
5	Required	Update CICS CSD with file definitions for C++ and MQSeries Workflow for OS/390.	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	Make MQSeries CICS stubs available in CICS.	Make the MQSeries CICS Stubs `IMQB23IC` and `IMQS23IC` available to CICS. For more information see *MQSeries for OS/390: System Management*.	
7	Required	Restart CICS.	Restart CICS.	

8	Required	Make the C/C++ group CLB and the MQSeries Workflow for OS/390 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 OS/390 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 OS/390 group.	

9	Required	Verify profile access.	1. Logon to CICS. 2. Perform: `CEMT I FI(FMCHEUPR)` 3. One file should be displayed. Try to open the file by typing ″`OPE`″ over ″`CLO`″ (and pressing enter). If this works without resulting in an error message, the profile access has been established. If you get an error message, retry the previous steps for enabling CICS API support. If this does not help, contact your IBM representative. 4. You can now close the file again by typing ″`CLO`″ over ″`OPE`″ Since CICS will then disable the file, type ″`ENA`″ over ″`UNE`″ (UNEnabled).	

IMS API support customization

This makes MQSeries Workflow for OS/390 DLLs available to IMS so that programs using the MQSeries Workflow for OS/390 container API can be executed in IMS. If you only want to use IMS legacy applications, or if you do not want to use IMS at all you can skip this customization, and continue at "Workflow server customization" on page 25.

Table 21. 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 SYS1.PROCLIB. For each Workflow system, you must copy a template into SYS1.PROCLIB, and then customize it. This is necessary to start the MQSeries Workflow for OS/390 servers as a started task.

Table 22. 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 SYS1.PROCLIB, where *UniqueSystemKey* is your value in Table 3 on page 7.	
2	Required	Assign RACF user ID and group to the Workflow server started task. **Note:** This is the *ServerUserID*, see Table 3 on page 7	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 only if you want console messages to be in uppercase, otherwise optional	Modify the Workflow server start job definitions.	1. Edit SYS1.PROCLIB(*UniqueSystemKey*). 2. 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). 3. If you want, you can modify the DD statements for the stdout and stderr output, and the simple trace output. 4. If you want to, you can modify the sysout class for stdout, stderr, and simple trace output.	
5	Required only if you want servers and tools to give MMS messages in uppercase	Modify the language setting in the machine profile.	Edit the machine profile *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	
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 6, and *ServerUserID* in Table 3 on page 7.	

LAN client customization

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

1. "Customize the MQSeries client connection" on page 26
2. "Customize the MQSeries Workflow client" on page 26

Note: It is very important that you check the files called Readme.1st and Readme.xxx (where xxx is your language code) on the *MQSeries Workflow Version 3.1.2 CD*.

Customize the MQSeries client connection

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

Table 23. 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	Define the connection to MQSeries on OS/390 by setting the environment variable `MQSERVER`	**For Windows 95/NT and OS/2 clients** `set MQSERVER=`*QueueManager*`.CL.TCP/TCP/`*MQHostName* **For UNIX clients** `export MQSERVER=`*QueueManager*`.CL.TCP/TCP/`*MQHostName* where *QueueManager* is your value from Table 5 on page 9, and *MQHostName* is your value from Table 6 on page 9. **Note:** By setting the variable `MQSERVER`, existing settings for `MQCHLLIB` and `MQCHLTAB` will be over-ruled.
3	Optional	Test the MQSeries client connectivity.	Execute the sample program `IMQWRLDC` that is provided on the MQSeries CD.

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 24. Customize the MQSeries Workflow client

Step number	Required or optional	Description	Action
1	Required	Install an MQSeries Workflow client.	Install an MQSeries Workflowclient from the *MQSeries Workflow Version 3.1.2 CD* as described in *IBM MQSeries Workflow: Installation Guide*.
2	Required	Update MQSeries Workflow machine profile.	Issue the following command in the MQSeries Workflow client's binary directory: `fmczchk -c prf:m,FMLSegmentation,N`

Now you have customized the MQSeries Workflow client.

System customization verification

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

Table 25. System customization verification

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 7, and *AdminServerID* is a made-up name used to identify the administration server for the system identified by *UniqueSystemKey*.	`rc=0`
2	Required	Start the runtime client.	Double-click on the runtime client icon.	You are prompted for a user ID and password.
3	Required	Logon.	Logon using the user ID `ADMIN`, and the password ″`password`″.	If no error message is displayed, the verification is complete.
4	Required	Logoff.	Logoff the runtime client.	
5	Required	Stop the OS/390 administration server.	On the OS/390 system console, issue the command `MODIFY` *`AdminServerID`*`,STOP ADM` where *AdminServerID* is the name you used when starting the server.	`rc=0`

Verify Workflow client sample application

This customization verification stage is optional, if you wish, you can skip to "Program execution customization" on page 29.

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

Table 26. 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 7, and *AdminServerID* is a made-up name used to identify the administration server.	`rc=0`

Table 26. Verify Workflow client sample application (continued)

Step number	Required or optional	Description	Action	Verification
2	Required	Import the Workflow sample process model.	Submit JCL *InstHLQ*.SFMCCNTL(FMCHJFDL) **Note:** The user performing this step should have DB2 SYSADM rights.	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.	rc=0
4	Required	Start the runtime client.	Double-click on the runtime client icon.	You are prompted for the client's user ID and password.
5	Required	Logon.	Logon using the user ID ADMIN, and the password "password".	
6	Required	Start a new process instance.	1. Look up your process templates. 2. Create an instance of the process template named Edit. 3. Refresh the instance window. **Note:** For more information about using the runtime client, refer to *IBM MQSeries Workflow: Getting Started with Runtime*.	
7	Required	Start a new workitem.	1. Start the process instance. 2. Refresh the workitem window.	An editor should be displayed. A new workitem named Edit_Activity should appear in the workitem window.
8	Required	Terminate the workitem.	1. Close the editor. 2. Refresh the instance and workitem windows.	The workitem named Edit_Activity is finished, and the instance has terminated.
9	Required	Logoff.	Log off the runtime client.	
10	Required	Stop the execution server.	On the OS/390 system console, issue the command: `MODIFY` *AdminServerID*`,STOP EXE`	rc=0
11	Required	Stop the OS/390 administration server.	On the OS/390 system console, issue the command: `MODIFY` *AdminServerID*`,STOP ADM`	rc=0

Program execution customization

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

Table 27. Customizing program execution invocation types

Program type	Invocation type	Customization required
CICS	EXCI	"Customize CICS EXCI invocation" on page 30
	MQSeries CICS Bridge	"Customize MQSeries CICS bridge invocation" on page 31
IMS	CPIC	"Customize IMS CPIC invocation" on page 34
	MQSeries IMS Bridge	"Customize MQSeries IMS bridge invocation" on page 36

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 38
2. "Configure program execution samples" on page 39
3. "Verify program execution samples" on page 40

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 28. 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 number 4: Enable user IDs* 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 number 4: Enable user IDs* 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	Define the RACF profiles required to give the program execution server authority to run CICS applications using EXCI calls.	1. Identify the RACF user ID of the PES (see *ServerUserID* in Table 3 on page 7), the ID of the of the CICS server region (see *applid* in Table 6 on page 9), 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.	

Table 28. Customize CICS EXCI invocation (continued)

Step number	Required or optional	Description	Action	Verification
4	Required	Enable user IDs.	If you set ATTACHSEC =IDENTIFY in step number 1.1.d above then the invoked applications will be run using the RACF user ID *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 7.	

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 29. 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 9) with attributes: a. SHARE b. MSGDLVSQ(FIFO) c. DEFPERSIST(YES) d. HARDENBO

Table 29. Customize MQSeries CICS bridge invocation (continued)

Step number	Required or optional	Description	Action

3	Required only if security checks are required.	Define user ID under which the CICS bridge (monitor) is started as a surrogate user ID of all RACF user IDs for which program execution requests to the CICS bridge should be issued	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 RACF user IDs to be allowed to run CICS bridge invocations. This must be the user ID under which the CICS bridge (monitor task) is started.
			mqwf_uid
			RACF 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.

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 RACF user IDs for which CICS bridge request should be issued (RACF user IDs corresponding to MQSeries Workflow user IDs). 2. Give `WRITE` access to the CICS bridge request queue to the *ServerUserID* (see your value in Table 3 on page 7). 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 RACF user IDs for which the bridge should put reply messages on that queue. 5. Give `WRITE` access to all reply to queues to the user ID of the CICS bridge monitor. 6. Give `WRITE` access to the dead-letter queue to all RACF user IDs for which requests should be issued and to the user ID of the CICS bridge monitor.
5	Required only if the target CICS requires security checks for RACF user IDs of MQSeries Workflow users.	Define RACF authority access to CICS programs to RACF user IDs of MQSeries Workflow users.	1. Identify the RACF 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.

Table 29. Customize MQSeries CICS bridge invocation (continued)

Step number	Required or optional	Description	Action
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 RACF 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 9. 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 OS/390 does not support passwords the authority levels, `VERIFY_UOW` and `VERYFY_ALL` are not supported by CICS bridge invocations.

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 30. 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 `LUADD`s 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 the `SYS1.PARMLIB` this can be used to run the `ASCH` address space. 2. If there is no `ASCHPMxx PARMLIB` member you must create one. For details on how to create one, see *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 30. Customize IMS CPIC invocation (continued)

Step number	Required or optional	Description	Action
4	Required	Define a system base APPC LU for VTAM that is enabled for protected conversation support.	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 RACF 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 OS/390 RACF user IDs by issuing the RACF command `PERMIT <ims_lu> CLASS(APPL) ID(<user_id>) ACCESS(READ)` repeatedly for each user ID or a RACF group ID. 3. Activate the above made definitions by issuing the RACF command `SETROPTS CLASSACT(APPL) RACLIST(APPL)` 4. To activate the definitions, issue the RACF command `SETROPTS RACLIST(APPL) REFRESH` **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 30. 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 `ASCHPM`*xx* containing the configuration of the ASCH. 3. If APPC/MVS is already running and the changes in `APPCPM`*xx* 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 31. 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 9) 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 31. 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 9), and with the XCF member name of the IMS (see *XCFMemberIMS* in Table 6 on page 9).
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 9) 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 information into the PES directory database.

Table 32. Customize program execution server directory

Step number	Required or optional	Description	Action	Verification
1	Required	Add EXCI, CPIC, and MQSeries bridge connection parameters to the PES directory.	Edit *CustHLQ*.`SFMCDATA(FMCHEDTP)`, 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 <*queuemanager*> to the name of the MQSeries queue manager you want to use (see your value for *QueueManager* in Table 5 on page 9). b. Change <*queuename*> to the name of the MQSeries CICS bridge input queue, see your value for *CICSBridgeInputQueue* in Table 6 on page 9. 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 <*queuemanager*> to the name of the MQSeries queue manager you want to use (see your value for *QueueManager* in Table 5 on page 9). b. Change <*queuename*> to the name of the MQSeries IMS bridge input queue, see your value for *IMSBridgeInputQueue* in Table 6 on page 9.	
2	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 19.

Now you are ready to perform "Configure program execution samples" on page 39.

Configure program execution samples

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

Table 33. 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.	Copy the following files to 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 7 to execute the sample programs, and all resources required by them.

Now you are ready to perform “Verify program execution samples” on page 40.

Verify program execution samples

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

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

Table 34. Verify program execution samples

Step number	Required or optional	Description	Action	Verification
1	Required	Ensure that the necessary subsystems are running.	If necessary, start DB2, MQSeries *QueueManager*, CICS, or IMS.	
2	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 7, and *AdminServerID* is a made-up name used to identify the administration server.	`rc=0`
3	Required	Start the MQSeries Workflow for OS/390 system.	On the OS/390 system console, issue the command: `MODIFY` *`AdminServerID`*`,START`	`rc=0`
4	Required	Start the runtime client.	Double-click on the runtime client icon.	You are prompted for the client's user ID and password.
5	Required	Logon.	Logon using the user ID `ADMIN`, and the password "`password`". **Note:** For more information on using the runtime client, refer to *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.
6	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

Table 34. Verify program execution samples (continued)

Step number	Required or optional	Description	Action	Verification
7	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
8	Required	Create workitems.	1. Select all process instance items and start them. 2. Click OK for every window that appears. 3. Create new worklist.	You will see workitems for: 1. CICSMapping_Activity 2. CICSContainer_Activity 3. IMSMapping_Activity 4. IMSContainer_Activity
9	Required	Start new workitems.	1. Start new workitems a. CICSMapping_Activity b. CICSContainer_Activity c. IMSMapping_Activity d. IMSContainer_Activity 2. Refresh the workitem list.	After successful completion, each workitem returns "**Finished**".

10	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`	

11	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`	

Table 34. Verify program execution samples (continued)

Step number	Required or optional	Description	Action	Verification

12	Optional	Check results of **IMSMapping _Activity**.	The latest `SYSxxxx` section of your IMS region job should contain:
			`FMCH3IMS: MQWF Program Execution Customization`
			`LastName:   Smith`
			`FirstName:  John`
			`Zip:        12345`
			`Salary:     1000.42`
			`Tax:        15.5`
			`Customer LastName : EINSTEIN`
			`Customer FirstName : ALBERT`
			`Customer PhoneNumber : 3048`
			`Customer LastName : NEWTON`
			`Customer FirstName : ISAAK`
			`Customer PhoneNumber : 4041`
			`Customer LastName : KOHL`
			`Customer FirstName : HELMUT`
			`Customer PhoneNumber : 5154`
			`New Salary: 1080.45`
			`ISRT: CEETDLI successful`	

13	Optional	Check results of **IMSContainer _Activity**.	The latest `SYSxxxx` section of your IMS region job should contain:
			`FMCH3ICS: MQWF Program Execution Customization`
			`OutContainer Name = SimpleDS`
			`InContainer Name = SimpleDS`
			`Set OutputContainer long value ml rc = 0`
			`Main: Set OutputContainer rc = 0`	

14	Required	Logoff.	Log off the runtime client.	
15	Required	Stop the MQSeries Workflow for OS/390 system.	On the OS/390 system console, issue the command: `MODIFY` *AdminServerID*`,STOP`	`rc=0`
16	Required	Stop the OS/390 administration server.	On the OS/390 system console, issue the command: `MODIFY` *AdminServerID*`,STOP ADM`	`rc=0` Congratulations, you have now configured and verified your MQSeries Workflow for OS/390 system.

Part 2. System administration

Chapter 4. Introduction to system administration

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

Objects you will need to administer or use

The administration of MQSeries Workflow for OS/390 requires that you use the following MQSeries Workflowand 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 clean-up server
- Zero or more program execution server instances.

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

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

Administration server
The component that performs administration functions within an MQSeries Workflow system. For OS/390 the administration server must be started manually from the system console, as described in "Starting the administration server and administration console" on page 53.

Administration console
The MQSeries Workflow for OS/390 administration console accepts administration commands for starting and stopping systems and servers. You can also use it to display how many of server instances are running. For more information, see "Chapter 5. Administration console tasks" on page 53.

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

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

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

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

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

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

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

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

Program mapping
The program execution server provides a default program mapper. You can define mapping rules for legacy programs so that they can be invoked. How to write program mapping rules is described in *MQSeries Workflow for OS/390: Programming*. How to administrate program mapping is described in "Administering program mapping" on page 74.

Buildtime
The MQSeries Workflow Buildtime is used to define process models and system configurations. Buildtime runs on a Windows workstation, you will use it to define the services to be made available to the MQSeries Workflow activities. Buildtime exports the process models in a format that is known as MQSeries Workflow Definition Language (FDL). You can use Buildtime to define the number of instances of execution and program execution servers that are started when the system is started up. You will have to transfer the FDL file to your OS/390 system, and import it into the MQSeries Workflow for OS/390 database using the import tool. See "Chapter 6. Buildtime administration tasks" on page 59. For more information about Buildtime see *IBM MQSeries Workflow: Getting Started with Buildtime*.

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 30 and "Customize MQSeries CICS bridge invocation" on page 31.

DB2
DB2 databases are used as repository for Workflow process models and to store the in-flight state of created and running process instances. Each MQSeries Workflow for OS/390 system group needs its own database, and contains one system. Multiple MQSeries Workflow for OS/390 system groups can share one DB2 subsystem.

IMS
If you want to make IMS programs available to MQSeries Workflow activities, you may have to install and configure an MQSeries IMS bridge or the CPIC invocation type. For information about setting up IMS invocation types see "Customize IMS CPIC invocation" on page 34 and "Customize MQSeries IMS bridge invocation" on page 36.

MQSeries
MQSeries Workflow for OS/390 uses MQSeries message transportation, and requires a MQSeries for OS/390 queue manager.

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.

System trace
You can use the system trace facility for problem determination. For information about tracing see "The MQSeries Workflow for OS/390 system trace facility" on page 87.

Address spaces
During normal operation, you should only administer servers and systems using the administration console. On OS\390, servers run in address spaces. Several servers may run in the same address space, but for each server type a different address space is used. For a single-instance server like the administration server this means that it runs alone in an address space. Some of the servers are multiple-instance servers: If the workload requires it, additional server instances of this server type can be started. The number of server instances that can run in a single address space depends on the size of the instances. The maximum number of server instances that shall be started in one address space is a tuning parameter, which can be changed when tuning the performance (see "Changing the number of server instances per address space" on page 81). If more than this maximum number of server instances is started, additional address spaces will be used.

Administration in an MQSeries Workflow system

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

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

An MQSeries Workflow system has a tiered structure:

- **Tier 1 — Client tier**

  Tier 1 contains the MQSeries Workflow system clients, application programming interfaces, and Buildtime. They use the MQSeries client and MQSeries Workflow APIs to connect with the second tier.
- **Tier 2 — Server tier**

  Tier 2 can be split into two parts. The first part contains all the various MQSeries Workflow servers. This is the working center where all the scheduling, distribution, clean-up, administration, server communication, and execution is done. The administration server is located in this tier. It communicates with the other components in the system using MQSeries techniques. The server tier also contains the administration console.

  The second part contains the MQSeries Workflow database. This holds system status and setup information for the complete system. The administration server accesses system tables within the database and returns the contents back to the administration console when requested to do so. The database is automatically updated by the administration server in response to system events.

The administration console is the user interface to the administration server and is used by a system administrator to request services from the administration server, see “Chapter 5. Administration console tasks” on page 53. For details on how to use the administration utility to administer the MQSeries Workflow system on AIX, and Windows NT, see *IBM MQSeries Workflow: Administration Guide*.

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

System administration client/server components

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

Figure 3 on page 49 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.

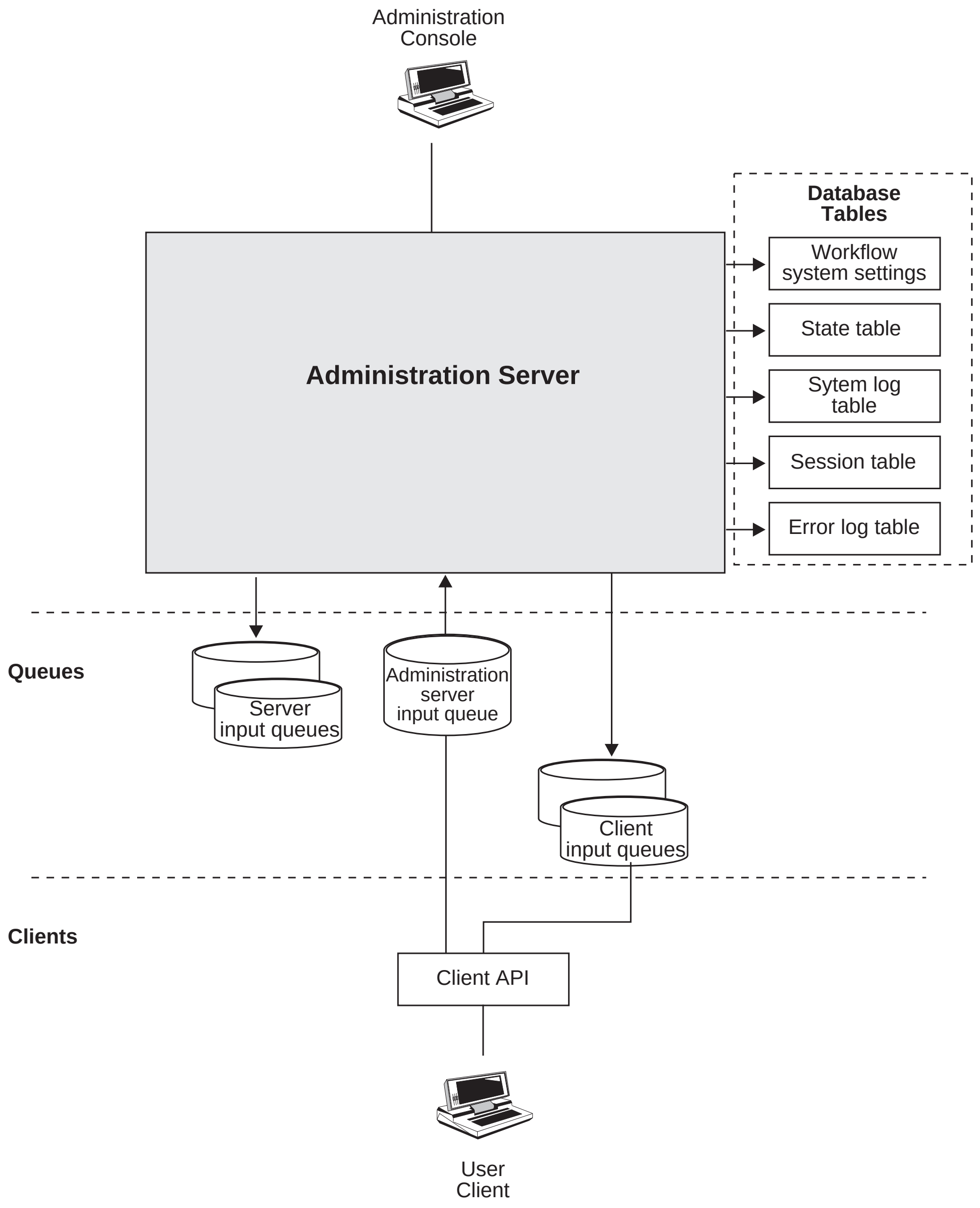


Figure 3. Implementation of the administration component in an MQSeries Workflow system

The administration server

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

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

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

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

Database tables

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

- The administration server state table, which lists the administration server state properties, and the operational status of system servers.
- Property tables for all servers.
- The system properties table in which properties that determine the behavior of the system are contained
- The system group properties table in which properties that determine the behavior of the system group and some system properties are contained.
- The domain properties table contains properties that determine the behavior of the domain and some system group and system properties.
- A session table in which a session record is created for each authorized user after logon.

The administration console

The administration console provides a command line interface to the administration server. It allows the OS/390 administrator to start and stop MQSeries Workflow systems and servers, and to query how many server instances are running. The tasks that can be performed from the OS/390 administration console are described in “Chapter 5. Administration console tasks” on page 53.

Overview of administration tasks

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

System and server administration tasks

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

Table 35. System and server administration tasks

Task	System console command	Admin console command
"Starting the administration server and administration console" on page 53	•	
"Stopping the administration server" on page 53	•	•
"Starting the system" on page 54		•
"Stopping the system" on page 54		•
"Restarting the system" on page 55		•
"Starting servers" on page 56		•
"Stopping servers" on page 57		•
"Restarting servers" on page 57		•
"Displaying the number of instances of a server" on page 58		•

System console

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

Administration console

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

Program and user administration tasks

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

Table 36. Program and user administration tasks: tool dependencies

Task	Buildtime	PES directory	RACF	Program mapping
"Defining process models" on page 59	•			
"Uploading process models to the host" on page 65				
"Importing and exporting process models" on page 65				
"Enabling an OS/390 program to be executed as a program activity" on page 72	•	•	•	•
"Disabling a program" on page 73	•			
"Enabling an OS/390 program to run as safe application" on page 73	•			
"Authorizing a user to access an OS/390 program" on page 73		•	•	
"Revoking a user's access to OS/390 programs" on page 74		•	•	
"Importing a program mapping definition" on page 74				•
"Enabling a program's mapping" on page 76	•	•		
"Disabling a program's mapping" on page 77	•			

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

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

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

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

Chapter 5. Administration console tasks

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

The following sections describe three groups of commands:

- “Administration server commands”
- “System commands” on page 54
- “Server commands” on page 55

Administration server commands

Starting the administration server and administration console

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

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

```
START UniqueSystemKey.AdminServerID
```

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 7. It must be unique within the OS/390 image, and not more than 8 characters long.

AdminServerID
: A name you will use to identify the administration server when you issue console commands. It must be unique within the OS/390 image, and not more than 8 characters long. As there is only one administration server per Workflow system, it is recommended that you construct the name from the letters `FMCA` and some unique characters from the system name, for example `FMCASYS1`.

For example, `START MQWFS1.FMCASYS1`.

Only one administration server can be started for each system. After executing this command you can start the system, see “Starting the system” on page 54.

Stopping the administration server

You can stop the administration server by issuing the following command on the system console:

```
STOP AdminServerID
```

This has the same effect as the stop server command:

```
MODIFY AdminServerID,STOP ADM
```

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

For example, `STOP FMCASYS1`.

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

System commands

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

- "Starting the system"
- "Stopping the system"

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 59 for more information.

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

```
MODIFY AdminServerID,START
```

or the short form

```
F AdminServerID,S
```

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

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

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

Stopping the system

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

```
MODIFY AdminServerID,STOP
```

or the short form

```
F AdminServerID,P
```

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

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

This stops all servers (except for the administration server) that are running on the Workflow system. How to stop the administration server is described in "Stopping the administration server" on page 53.

Restarting the system

Changes made to the machine profile or the environment variable file will only affect new server instances and tools that are started. If you want such changes to affect all running server instances then you must restart the whole system, including the administration server in the following sequence:
1. "Stopping the system" on page 54
2. "Stopping the administration server" on page 53
3. "Starting the administration server and administration console" on page 53
4. "Starting the system" on page 54
5. Then start extra server instances if required, see "Starting servers" on page 56.

which requires the command sequence:
```
MODIFY AdminServerID,STOP
STOP AdminServerID
START UniqueSystemKey.AdminServerID
MODIFY AdminServerID,START
MODIFY AdminServerID,START ServerType [INST(NumberOfInstances)]
```

Server commands

You can start and stop any MQSeries Workflow for OS/390 servers from the administration console. You can also query how many server instances are running. These tasks are described in:
- "Starting servers" on page 56
- "Stopping servers" on page 57
- "Restarting servers" on page 57
- "Displaying the number of instances of a server" on page 58

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 37. Server types

ServerType	**Server**	**Instantiation type**
`ADM`	Administration server.	Single instance
`SCH`	Scheduling server	Single instance
`CLE`	Clean-up server.	Single instance
`EXE`	Execution server.	Multiple instance
`PES`	Program execution server.	Multiple instance

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

Starting servers

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

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

or the short form

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

AdminServerID
: The administration server that is to start the server.

ServerType
: The type of server to be started. You cannot start the administration server using this command, see "Starting the administration server and administration console" on page 53.

NumberOfInstances
: This optional parameter specifies how many new instances of the server should be started. The default value is specified in the server settings in Buildtime. Single instance server types can have only one instance. Multiple instance server types can have many instances.

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 execution servers

To activate the MQSeries Workflow for OS/390 system at least one execution server must be running. It is possible to start multiple instances of the execution server to share the work load. For example, if you want to start 2 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)
```

Starting program execution server instances

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

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

Stopping servers

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

```
MODIFY AdminServerID,STOP ServerType
```

or the short form

```
F AdminServerID,P ServerType
```

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

ServerType
: The type of server to be stopped.

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

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

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

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 84.

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, followed by 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 76.

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

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

Restarting the administration server

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

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

- If you want to activate changes made to the server properties in the process model.

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

For example, if your administration server ID is `FMCASYS1`, and your *UniqueSystemKey* is `MQWFS1` (see your value in Table 3 on page 7), then

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

will restart the administration server.

Displaying the number of instances of a server

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

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

or the short form

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

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

ServerType
: The type of server instances to be counted.

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

Chapter 6. Buildtime administration tasks

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

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

Defining process models

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

Note: To avoid code page conversion problems when uploading FDL to the host, your Buildtime object names should conform to the guidelines that are described in "Appendix D. Naming and code page restrictions" on page 105.

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 60
- "Defining the connection between a program activity and the PES" on page 64

After you have changed properties, and exported the process model, you will have an FDL file that must be imported into the Workflow database, as described in "Importing and exporting process models" on page 65.

Defining server properties

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

Table 38. 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
User support (PES specific)	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. **Note:** In Buildtime, the **Program** option may be displayed as **Starter**	Agent

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

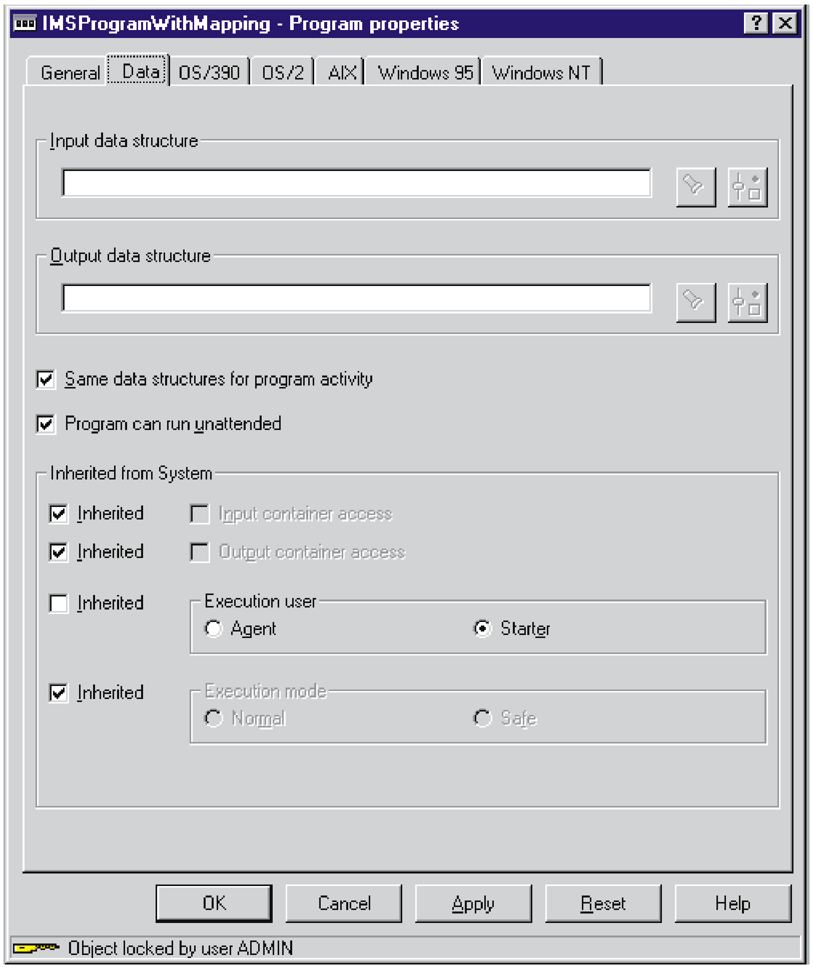
Table 39. Server properties that should not be changed

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
Start time	This setting is ignored on OS/390.	
Stop time	This setting is ignored on OS/390.	

Defining program properties

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

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


IMSProgramWithMapping - Program properties

General | Data | OS/390 | OS/2 | AIX | Windows 95 | Windows NT

Input data structure

Output data structure

[✓] Same data structures for program activity

[✓] Program can run unattended

Inherited from System

[✓] Inherited [ ] Input container access

[✓] Inherited [ ] Output container access

[ ] Inherited Execution user: Agent [✓ Starter]

[✓] Inherited Execution mode: Normal / Safe

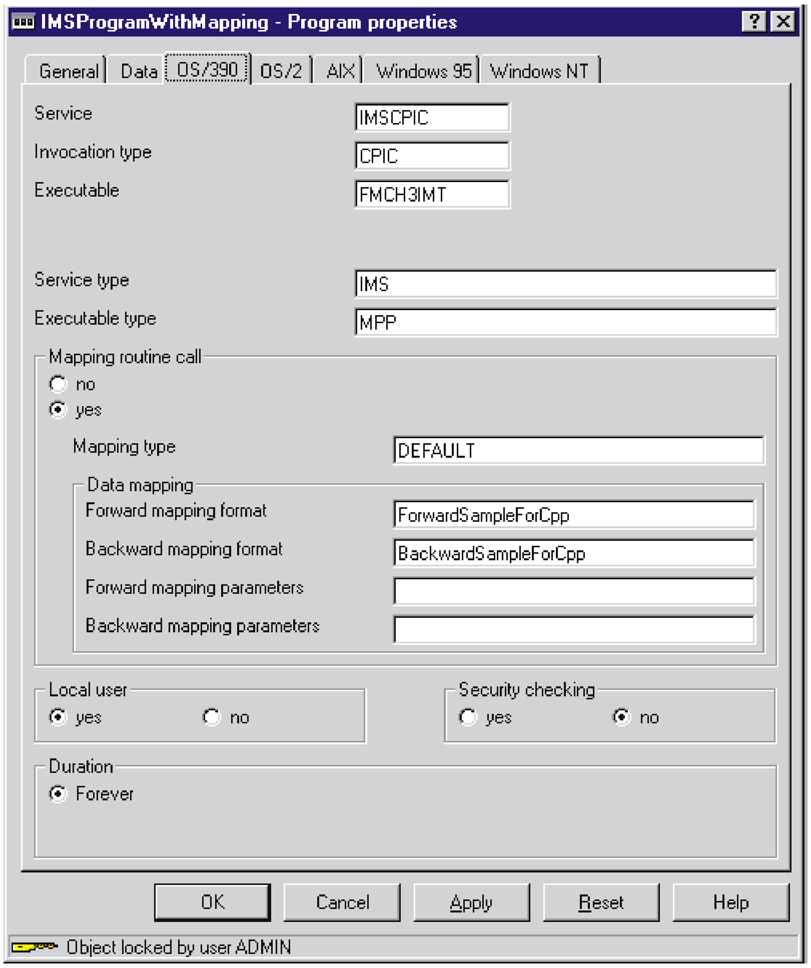
OK | Cancel | Apply | Reset | Help

Object locked by user ADMIN

Figure 4. Program properties: Data page

Figure 4 shows the program properties data page for the an example program named `IMSProgramWithMapping`. The option **Execution user = Starter** means that the PES has to execute the program using the execution user ID of the starter of the activity. This option requires the PES setting **User support = Starter**. In this case, the PES has to map the Workflow user ID of the starter of the request to an execution user ID known to OS/390. See “Adding a new service definition and the related user resolution information” on page 70.

For more information about these settings, see “Program execution security” on page 79.


IMSProgramWithMapping - Program properties

General | Data | OS/390 | OS/2 | AIX | Windows 95 | Windows NT

Service: IMSCPIC
Invocation type: CPIC
Executable: FMCH3IMT
Service type: IMS
Executable type: MPP

Mapping routine call: no [✓ yes]
Mapping type: DEFAULT
Data mapping
Forward mapping format: ForwardSampleForCpp
Backward mapping format: BackwardSampleForCpp
Forward mapping parameters:
Backward mapping parameters:

Local user: [✓ yes] no
Security checking: yes [✓ no]
Duration: [✓ Forever]

OK | Cancel | Apply | Reset | Help

Object locked by user ADMIN

Figure 5. 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 40. 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.

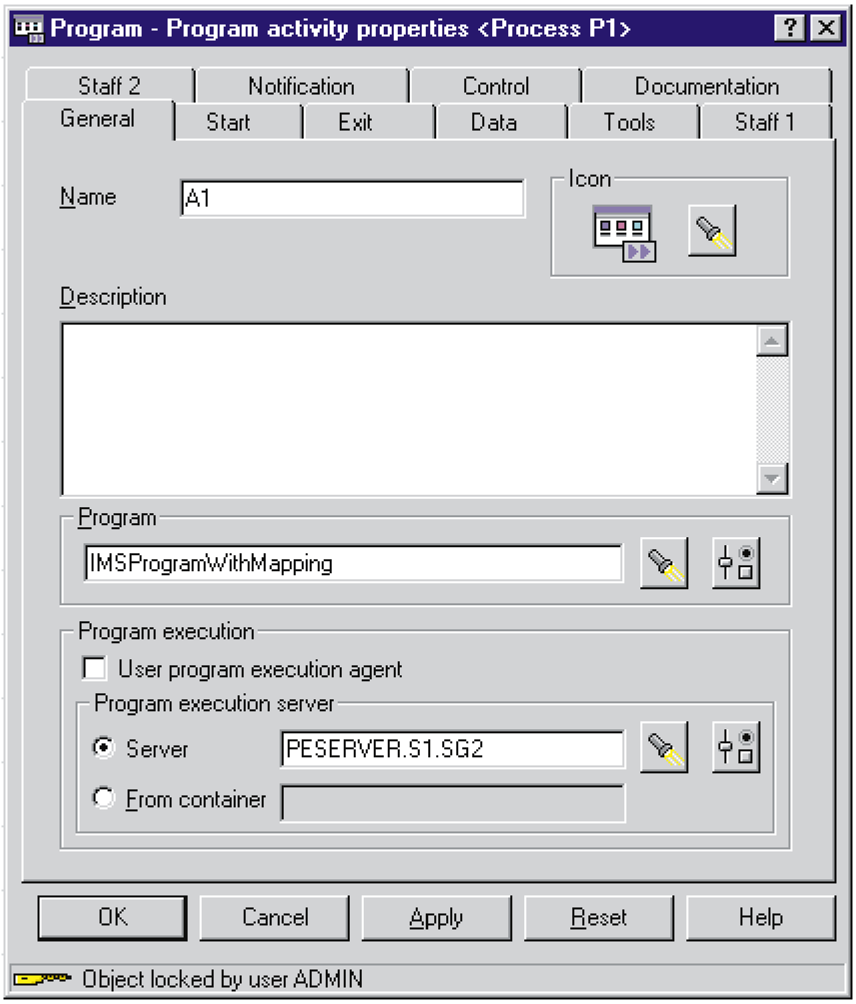
Table 40. Program properties: OS/390 page settings (continued)

OS/390 program property	**Description**
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 OS/390: 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 OS/390: 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 OS/390: 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 80.
Duration	For OS/390 programs, this value is set to Forever.

Defining the connection between a program activity and the PES

After an OS/390 program has been defined, it must be associated with a program activity, and the program execution server. This is done on the screen that is shown in Figure 6. 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.



Name: A1

Program: IMSProgramWithMapping

[ ] User program execution agent

(●) Server: PESERVER.S1.SG2

( ) From container:

Object locked by user ADMIN

Figure 6. Program Activity Properties: connecting a program to the PES

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

1. Click on the flashlight button to find the program name and select the program,
2. Clear the **User program execution agent** check box.
3. Click on the flashlight button to find the program execution server and select `PESERVER`.
4. Press **OK**, 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 107.

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 105.

Importing and exporting process models

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

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

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

For more information about the export and translation options, see “FDL import/export tool’s syntax” on page 109, “FDL export examples” on page 113, and “Translate examples” on page 114.

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 111)
   b. Specify the input and output files using the predefined `DD`-names `FMCIIMP`, `FMCIEXP`, `FMCICMD`, and `FMCILOG`, as illustrated in “FDL import examples” on page 112.
   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 7. 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 7 on page 68.

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

PES directory
Which invocation and mapping exits can be used by the PES at run-time 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 sucessfully. 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 69 and "Appendix A. Program Execution Server directory" on page 93.

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 70. The PES can perform security checks to ensure that only authorized users can run a program, as described in "Program security" on page 80.

Run-time 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 distinguishes between recoverable and non-recoverable errors. Recoverable errors are returned as an error indication and processing continues with the next request. Unrecoverable errors cause the PES instance to terminate, and an error indication is written to `FMCERR`.

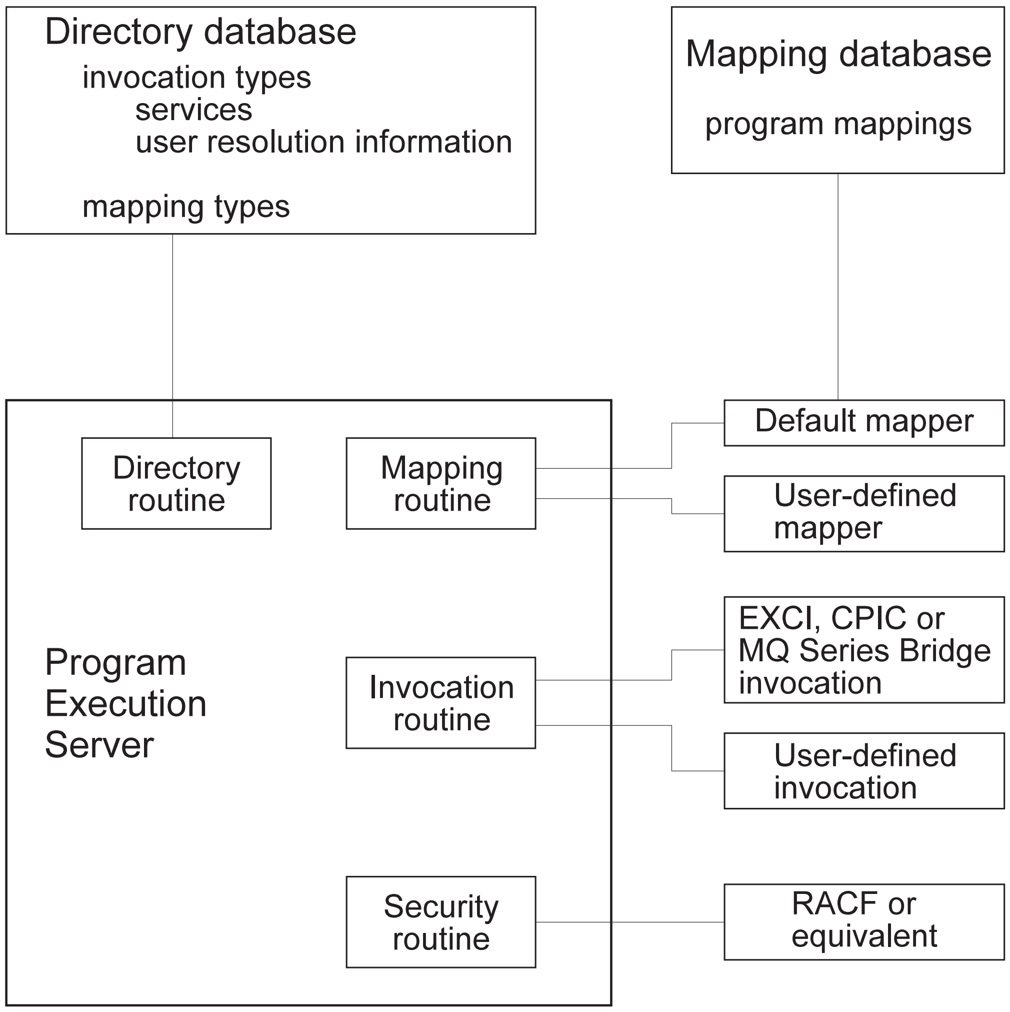


Figure 7. OS/390 Program execution server: component structure

The PES processes a synchronous request in the following way:

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

Invocation types supported

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

CICS program types supported

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

IMS program types supported

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

Mapping types supported

MQSeries Workflow for OS/390 supplies a program mapping type named `DEFAULT` which should be able to handle most mapping requirements. A user type can be defined to handle special cases that are not covered by the `DEFAULT` mapping types' interface types. For more information about how to define mappings and user types, see *MQSeries Workflow for OS/390: Programming*. The mapping type to be used for each program is defined in the process model as shown in Figure 5 on page 62.

User-defined invocation and mapping types

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

If you define your own invocation and mapping types, they must be defined consistently in the process model and in the PES directory, as explained in more detail in "PES directory dependencies on the process model's OS/390 program definitions" on page 97.

Administering the Program Execution Server directory

The PES directory provides persistent runtime information for the OS/390 program execution server. It is a DB2 database, and it must be located in the same DB2 subsystem as the MQSeries Workflow for OS/390 database. The directory is a read-only database; any data retrieved from the directory is not cached within the program execution server. The directory may be updated while the server is running.

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 93.

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 71.

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 99, and
- "PES directory import examples" on page 99.

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 71. For more information about the PES directory structure see "Appendix A. Program Execution Server directory" on page 93.

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 95 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 99.
   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)

Administering programs

Program administration consists of the following tasks:
- "Enabling an OS/390 program to be executed as a program activity"
- "Enabling an OS/390 program to run as safe application" on page 73
- "Disabling a program" on page 73
- "Authorizing a user to access an OS/390 program" on page 73
- "Revoking a user's access to OS/390 programs" on page 74

Enabling an OS/390 program to be executed 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 60
2. "Creating a program mapping"
3. "Defining a security profile" on page 73

Creating a program mapping

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

If you have created a program mapping, you will have to activate the mapping in the process model. This task is described in "Enabling a program's mapping" on page 76.

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 60
2. "Uploading process models to the host" on page 65
3. "Importing and exporting process models" on page 65

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 80.

Enabling an OS/390 program to run as safe application

When an IMS 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 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.

Setting **Execution mode** in the program's process model definition to **Normal** disables the safe-mode execution of the program. In normal execution mode the program is guaranteed to be executed at least once.

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 5 on page 62), 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 80.
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 70.

4. Reload the PES directory as described in "Importing the PES directory" on page 71 .

Revoking a user's access to OS/390 programs

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

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

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

Administering program mapping

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

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

After you have created a program's mapping definitions:

- You must import the mapping definitions into the mapping database as described in "Importing a program mapping definition".
- A program mapping in the mapping database only becomes active after the mapping has been enabled as described in "Enabling a program's mapping" on page 76.

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 57.

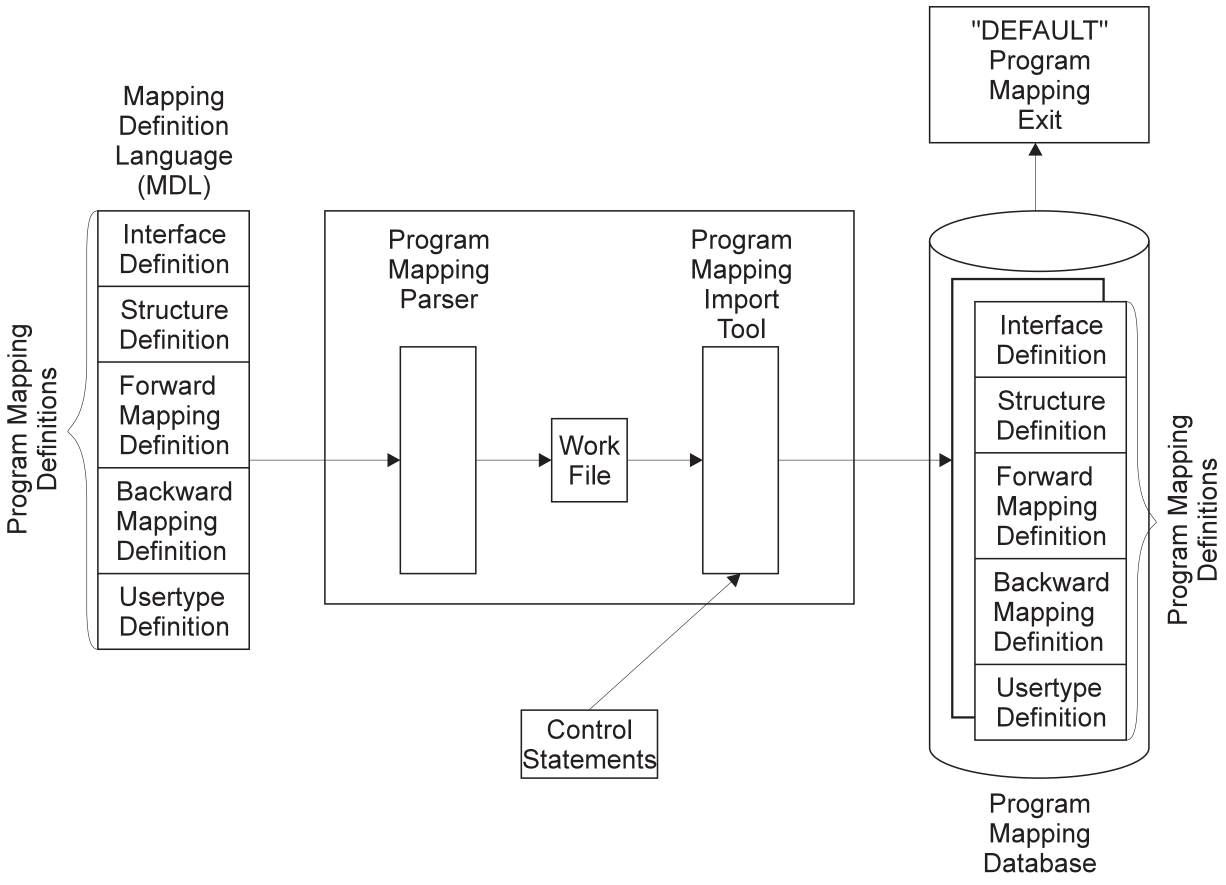

"DEFAULT" Program Mapping Exit
Mapping Definition Language (MDL)
Program Mapping Definitions
Interface Definition
Structure Definition
Forward Mapping Definition
Backward Mapping Definition
Usertype Definition
Program Mapping Parser
Work File
Program Mapping Import Tool
Control Statements
Interface Definition
Structure Definition
Forward Mapping Definition
Backward Mapping Definition
Usertype Definition
Program Mapping Definitions
Program Mapping Database

Figure 8. Program mapping definition process and components

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

1. Create the program mapping definitions as described in *MQSeries Workflow for OS/390: Programming*.
2. Customize a copy of *CustHLQ*.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 41. 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 60:
      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 65.

3. Import the process model on the host, as described in “Using the FDL import/export tool” on page 65.
4. If mapping definitions were modified, then restart the PES as described in “Restarting the program execution server” on page 57.

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 5 on page 62.
  2. Upload the new process model to the host, as described in “Uploading process models to the host” on page 65.
  3. Import the process model on the host, as described in “Using the FDL import/export tool” on page 65.

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 101.
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 OS/390: Programming*.
2. Provide the mapping exit DLL to MQSeries Workflow for OS/390. Either copy the DLL into the data set *CustHLQ*.SFMCLOAD, or concatenate the data set name, where the mapping exit DLL is stored, to SFMCLOAD.
3. Define the new mapping exit in the PES directory. Replicate the entry called (KEYTOMAPPING1) in the PES directory template, choose the next free number (e.g. 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 71.
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 5 on page 62.
6. Import the new process model into the Workflow database as described in "Importing and exporting process models" on page 65.

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 5 on page 62.
2. Import the changes, as described in "Importing and exporting process models" on page 65.
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 71.

Administering invocation types

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

Enabling an invocation type

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

1. The invocation type must be defined in the PES directory as described in "Adding a user-defined invocation type" on page 71.
2. The invocation exit DLL for MQSeries Workflow for OS/390 must be copied into *InstHLQ*`.SMFCLOAD`, or the data set containing that DLL must be concatenated to the DD statement `FMCSVLIB` in the JCL procedure *CustHLQ*`.SFMCPROC (FMCHPSRV)`.
3. The service and its connection parameters must be defined in the PES directory as described in "Adding a new service definition and the related user resolution information" on page 70.

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 5 on page 62.
2. Import the changes, as described in "Importing and exporting process models" on page 65.

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 71.

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 4 on page 61.

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 5 on page 62. 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**, run-time 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 5 on page 62.

The following combinations of settings are meaningful:

Table 42. 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 79 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 70.

Program security

If a program is defined with **Security checking=Yes**, then it must have a security profile defined for the executable. The profile name is *SystemQualifier.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 5 on page 62.

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 8. Performance tuning

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

- "Changing the number of running server instances"
- "Changing the number of server instances per address space"

Note: Some other performance issues are covered in "User response times are unacceptably long" on page 86.

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 and the PES are started with five server instances.

You can start additional servers using the administration console, as described in "Starting servers" on page 56. 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 each server type that can be started in one address space is defined in the machine profile, see "Appendix H. Machine profile" on page 117.

This only applies to the multiple instance servers:

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

The initial value for the execution server and the program execution server is five server instances per address space. This means that starting six 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.

Chapter 9. Problem determination

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

1. If you have problems with servers, see "Server problems".
2. If you have problems with resources or performance, see:
   a. "Resource and performance problems" on page 86
   b. "Chapter 8. Performance tuning" on page 81
3. If none of the previous solutions apply, you may decide to use one of the following:
   a. "The MQSeries Workflow for OS/390 system trace facility" on page 87
   b. "Tracing in CICS" on page 89
4. If you got an SVC dump, see "What do I do if I get an SVC dump?" on page 89.

Server problems

The administration server cannot be started

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 server 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

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`.

One or more program execution server instances terminate, the request is still in state running

The request has caused an unrecoverable PES error. These errors include:

- Program execution directory inconsistencies, for instance caused by wrong user-defined keys.
- Mismatches between process model program definitions and PES directory contents. For instance a service name defined for a program is not defined in the PES directory.
- PES is unable to establish a connection to a service. This might be caused by invalid connection parameters specified in the PES directory for this service.

If an error occurs, an error description (error ID and message text) is always written to `FMCOUT`, if tracing is turned on it is also written to `FMCTRC`. Refer to the error description corresponding to the error ID in *MQSeries Workflow for OS/390: Messages*. Correct the error and restart the request.

A dump is written before all server instances are started

Are there too many server instances per address space?

If less than the specified number of server instances are started, and then a dump is written, the value for the number of server instances started per address space should be reduced. This is described in "Changing the number of server instances per address space" on page 81.

Cannot stop servers

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.

Did you wait long enough?

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

Do your transactions take longer than 30 seconds?

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

If this problem persists, you can try increasing the value for the **WaitBetweenQInhibitAndAllowed** setting in the machine profile *CustHLQ*`.SFMCDATA(FMCHEMPR)`, see "Appendix H. Machine profile" on page 117 for more information. After changing the machine profile you must perform"Restarting the administration server" on page 57.

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 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 57.

Changes made to the machine profile are not activated

Have you restarted the administration server?

Changes to the machine profile will only become active after the administration server is restarted, as described in "Restarting the administration server" on page 57. All existing server instances will continue to use the old machine profile settings. After the restart all new server instances will use the new machine profile settings.

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

Resource and performance problems

User response times are unacceptably long

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

Is tracing turned on?

Operation is significantly slower when tracing is active. Turning tracing off is described in "Turning tracing off" on page 89.

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 56.

Are too many server instances running?

In this case restart fewer servers instances, as described in "Restarting servers" on page 57.

Does the workload exceed your system's capacity?

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

Invalid password

Are you using an old version of the runtime client?

This can happen when an old runtime client tries to connect to a newer administration server. You should install the MQSeries Workflow Version 3.1.2 runtime client.

Running out of spool space

Is tracing turned on?

This problem can be caused by the trace facility. Check which servers have been started with trace turned on. Trace entries are written even when the servers are idle. Reduce the trace level as described in "Turning tracing on" on page 87 (or turn it off) and restart the server type that was being traced.

The MQSeries Workflow for OS/390 system trace facility

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

The trace facility provides two types of tracing:

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

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

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

1. "Turning tracing on"
2. "Restarting the component and reproducing the problem" on page 88
3. "Viewing the trace" on page 88
4. "Turning tracing off" on page 89

Turning tracing on

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

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

   Note: This trace level significantly reduces the performance of the system, and requires a large quantity of disk space.
2. Edit the environment variable file *CustHLQ*`.SFMCDATA(FMCHEENV)`
   a. If you want extended trace, set
   ```
   FMC_SIMPLE_TRACE_ONLY=NO
   ```

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

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

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

Restarting the component and reproducing the problem

After starting the trace:

1. Run the component that is causing the problem.
2. Reproduce the problem.
3. If extended trace has been activated:
   a. Issue the following command on the OS/390 system console:
      ```
      TRACE CT,,COMP=CTComponent,PARM=CTIFMCmm
      ```
      where *CTComponent* and *mm* are your values for the CTRACE component name, and the two character CTRACE suffix *CTStopSuffix* — these values were defined in Table 3 on page 7.
   b. Run the JCL *CustHLQ*.SFMCCNTL(FMCHJTRC). This converts the format of the extended trace data sets into the same format as the simple trace.

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

Table 43. Extended trace format converter return codes

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

Viewing the trace

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

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

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

Turning tracing off

To deactivate the trace:

- Edit data set *CustHLQ*.SFMCDATA(FMCHEMPR) and specify

  MQWorkflowMachine.*SystemQualifier*.FMC_TRACE_CRITERIA:**0**,0000,00000000

If you have been tracing a server, it will continue writing trace data until it is stopped or restarted, as described in "Stopping servers" on page 57 and "Restarting servers" on page 57.

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

Tracing in CICS

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

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

The type of information that is provided, and the parameters for the MQSeries Workflow for OS/390 trace are described in "The MQSeries Workflow for OS/390 system trace facility" on page 87. For more information about the CICS trace facilities see *CICS/ESA: Problem Determination Guide* and *CICS Transaction Server for OS/390: CICS Problem Determination*.

What do I do if I get an SVC dump?

Create a problem summary

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

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

Part 3. 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 97.

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 94
- "Mapping section" on page 94
- "Security section" on page 94

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 95 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`.

Note: Multiple parameter assignments are separated by a semicolon (’;’).

User resolution: User resolution information is only required if a program is to run under a local user ID associated with the MQSeries Workflow user starting the execution request. This only applies to programs that are defined in the process model with **Execution user=Yes** and **Local user=Yes**. In this case, you have to add **userID/executionUserID** pairs to provide a mapping from each MQSeries Workflow *userID* who may access that service, mapping on to the OS/390 *executionUserID* that the program is to run under.

The reason for having this mapping is that MQSeries Workflow user IDs may be up to 32 characters long, whereas OS/390 user IDs are restricted to 8 characters.

Mapping section

The **mapping** section defines the program mapping types that the program execution server supports. For each mapping type, it defines the DLL name of the exit that is used by the mapping type, and the initialization parameters. The standard program mapping type defines the default mapper that is provided with MQSeries Workflow for OS/390.

Security section

The **security** section is reserved for future use and must not be modified.

PES directory template

A PES directory template file is provided in *CustHLQ*.SFMCDATA(FMCHEDTP). It contains definitions for the invocations types (EXCI, CPIC, MQCICS, and MQIMS) and the DEFAULT mapping type. It contains:

- A service section for each invocation type.
- A user section for each service section.

These have to be completed during program execution customization. The template contains the following:

```
;//*******************************************************************
;//*
;//* Description: Program Execution Server Directory Template
;//*
;//*******************************************************************
; Area of PES directory

(directory)

   programExecution         =keyToAreaOfPES ; Area of PES1

(keyToAreaOfPES1)
   pesName                  =PESERVER
   invocation               =keyToInvocation
   security                 =keyToSecurity
   mapping                  =keyToMapping

; Invocation section of PES1

(keyToInvocation1)
   type                     =EXCI
   exitName                 =FMCH0IEC
   exitParameters           =
   service                  =invocation1Service

; Service section of PES1

(invocation1Service1)
   type                     =CICS
   name                     =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 5 on page 62. 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 44. 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 45. PES directory import tool's return codes

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

PES directory import examples

The following JCL examples illustrate the use of the import options and DD statements.

Importing a PES directory source file

This example job imports the source file that is specified using the DD name FMCDIMP, creating the new entries in the directory.

```
//FMCHJPIF EXEC PROC=FMCHPBAT,PROGRAM=FMCH1PIT,
//             PARM='ENVAR("_CEE_ENVFILE=DD: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 102
- "Deleting a program mapping definition" on page 102
- "Listing program mapping definitions" on page 102

All control statements have the same format: **keyword type element**

1. The first word is a **keyword** which defines the action.
2. The second word defines which **type** of program mapping definition should be processed.

   Note: Valid types are: `STRUCTURE`, `INTERFACE`, `USERTYPE`, `BACKWARDMAPPING`, and `FORWARDMAPPING`.
3. The third word defines which **element** of this type should be processed.

   Note: You can use the wildcard character '*'. You can combine wildcard control statements with non–wildcard control statements. The wildcard control statements can be used to select all elements of the mapping definition or mapping database without explicitly naming them.

C and C++ style comments are allowed. Single line comments may start with the characters '`//`', and multi-line comments begin with '`/*`', and end with '`*/`'.

Creating a new program mapping definition

To create a new entry, specify:

```
CREATE EntryType EntryName
```

To create all entries of a given type, specify:

```
CREATE EntryType *
```

Note: If an entry already exists, the activity is rolled back, and you will get an error message.

Replacing an existing program mapping definition

To replace a specific entry for *EntryType* and *EntryName*, use the control statement:

```
REPLACE EntryType EntryName
```

To replace all entries for *EntryType*, use the control statement:

```
REPLACE EntryType *
```

Note: If the entry does not exist, the database transaction is rolled back, and you will get an error message.

Inserting a program mapping definition

To insert a specific entry for *EntryType* and *EntryName*, use the control statement:

```
INSERT EntryType EntryName
```

For example, `INSERT USERTYPE UT1`

Note: If the entry already exists, it will be overwritten.

To insert all entries for *EntryType*, use the control statement:

```
INSERT EntryType *
```

Deleting a program mapping definition

To delete a specific entry for *EntryType* and *EntryName*, use the control statement:

```
DELETE EntryType EntryName
```

To delete all entries for *EntryType*, use the control statement:

```
DELETE EntryType *
```

Note: If the entry does not exist, the database transaction is rolled back, and you will get an error message.

You can delete the whole database with the control statements:

```
DELETE USERTYPE *
DELETE FORWARDMAPPING *
DELETE BACKWARDMAPPING *
DELETE INTERFACE *
DELETE STRUCTURE *
```

Listing program mapping definitions

You can list all entries for a given type, with the control statement:

```
LIST EntryType *
```

This statement lists the entries by name and type in alphabetical order.

Control statement execution

The control statements are executed on the program mapper's database as a transaction. If any of the statements fail, the whole transaction is rolled back, and an error is returned. The control statements are not necessarily executed in the order that they are defined in the control member. The control statements are executed in the following sequence:

1. Any command for **forward mapping** definitions, in alphabetical order of the forward mapping name, followed by forward mapping commands that use the wildcard.

2. Any command for **backward mapping** definitions, in alphabetical order of the backward mapping name, followed by backward mapping commands that use the wildcard.
3. Any command for **structure** definitions, in alphabetical order of the structure name, followed by structure commands that use the wildcard.
4. Any command for **interface** definitions, in alphabetical order of the interface name, followed by interface commands that use the wildcard.
5. Any command for **user type** definitions, in alphabetical order of the user type name, followed by user type commands that use the wildcard.
6. **List** commands in alphabetical order of definitions for forward mapping, backward mapping, structure, interface, and user type.

Example control statements

The following example creates all user types, mapping definitions, and replaces usertype UT1:

```
REPLACE USERTYPE UT1          // Replace existing UT1
CREATE USERTYPE *             // Create all other usertypes
```

Appendix D. Naming and code page restrictions

MQSeries Workflow for OS/390 exploits the iconv function set of the C/C++ Compiler on OS/390 by converting incoming messages to to Unicode (UCS-2) and then to the local codepage. MQSeries Workflow for OS/390 relies on the converters available on the system and does not provide them as part of the product. Please see the *OS/390 C/C++ Programming Guide* for a list of supported unicode converters. There may be more converters available as PTF's.

You should verify that the code pages installed on all cooperating MQSeries Workflow platforms allow correct character conversion for all code points for a message round-trip.

MQSeries Workflow for OS/390 requires that certain naming restrictions are followed.

Naming Buildtime objects

The names given to MQSeries Workflow for OS/390 objects in the Buildtime should conform to the rules for naming MQSeries objects. If you follow these rules, no code page conversion problems should occur when you transfer the FDL file to the host. Names should only contain the following characters:

- Uppercase A-Z
- Lowercase a-z
- Numerics 0-9
- Period (.)
- Forward slash (/)
- Underscore (_)
- Percent sign (%)
- Parentheses (())

If you use object names which 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 107.

Restrictions for passwords in CICS

Passwords specified in the Logon API call from CICS programs must only include characters contained in codepage IBM-1047.

Appendix E. FDL code page conversion tool

If you have code page conversion problems when uploading your process model information, you can upload the FDL file as a binary image and then use the tool `FMCH1CNV` to convert the FDL file between particular source and target code pages.

Using the FDL code page conversion tool

In order to use this tool you have to transfer the FDL file as binary image to the host. The FDL file should be stored in a data set that has a variable record format. To use the `fmch1cnv` tool you should do the following:

1. Customize the JCL *CustHLQ*`.SFMCCNTL(FMCHJCNV)`
   a. Specify the options the conversion tool should use, see "Options".
   b. Specify the input and output files using the predefined `DD`-names `FMCCIMP` and `FMCCEXP`.
   c. If you want to use a specific log file instead of `SYSOUT`, you should specify a data set for the `DD`-name `FMCCLOG`.
2. Submit the JCL *CustHLQ*`.SFMCCNTL(FMCHJCNV)`

Options

You can start the conversion tool `FMCH1CNV` using the following options:

Option	Argument	Description
s	source code page	Name of the code set in which the input data is encoded. If you omit this option, the code page used is taken from the input file.
t	target code page	Name of the code set to which the output data is to be converted. If you omit this option, the local code page of your system determined at run-time is used.

Notes:

1. An equal sign (=), a comma (,), a colon (:), or a blank character can be used as an option delimiter.
2. The predefined `DD`-names: `FMCCIMP`, `FMCCEXP`, and `FMCCLOG` must be used to specify the input file, output file, and log file.
3. The record length of the output data set must have at least the same length as the longest line of the FDL input file transferred to the host.
4. The FDL conversion tool writes information, warning, and error messages to the log file that is specified by the `DD`-name `FMCCLOG`. By specifying `//FMCCLOG DD SYSOUT=*` the messages are written to `SYSOUT`.
5. Please see the *OS/390 C/C++ Programming Guide* for a list of supported code set converters.

Return codes

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

Table 46. FDL code page conversion tool's return codes

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

Appendix F. FDL import/export tool

Process model information is created in the Buildtime tool, and exported in FDL file format. You must use the import/export tool to:

- Import process model information into the Workflow database.
- Translate 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

```
                 ┌──────────────────┐
                 ▼                  │
►►──FMCH0IBA─────┴───┬─┤ Logon ├──┬─┴──────────────────────────────────────────►◄
                     ├─┤ Import ├─┤
                     └─┤ Export ├─┘
```

Logon:

```
├──┬─ -p── =──password──┬──────────────────────────────────────────────────────┤
   └─ -u── =──userid────┘
```

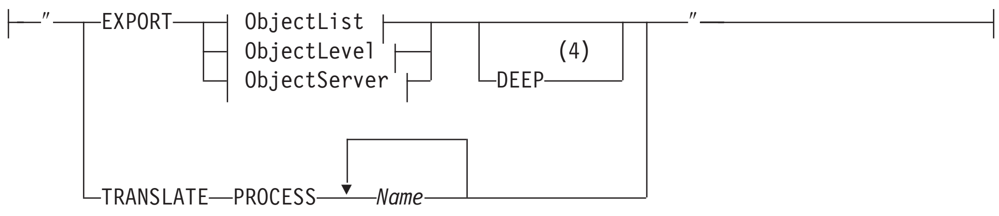
Import:

```
        (1)
├──┬─ -i────────┬───────────────────────────────────────────────────────────────┤
   ├─ -o────────┤
   └─ -t────────┘
```

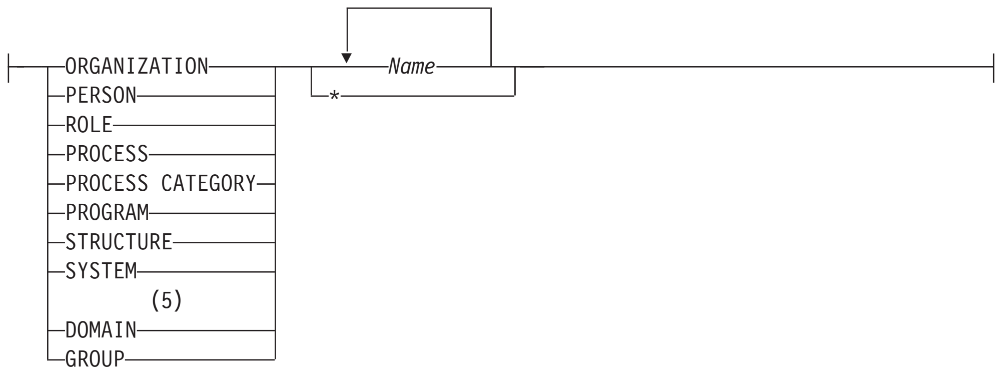
Export:

```
        (2)
├──┬─ -e──────────────────────────────────────────┬─────────────────────────────┤
   └─ -c =──┬─┤ EntityManagingCommand ├──┬─────────┘
            │     (3)                    │
            └─@──────────────────────────┘
```

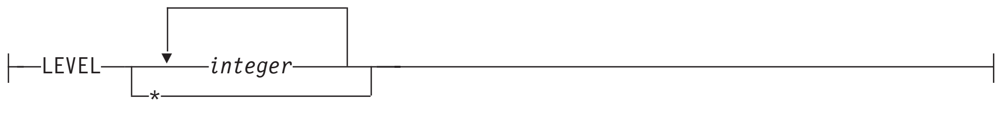
EntityManagingCommand:



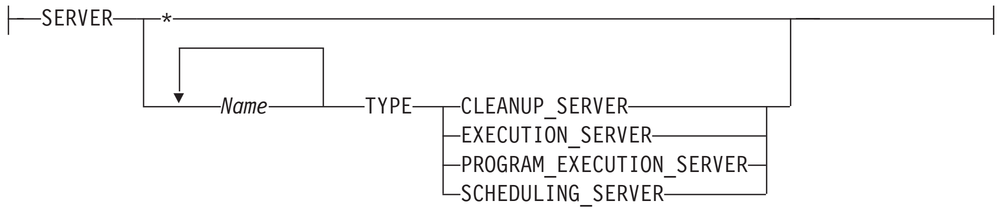
ObjectList:



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.

- 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 a process model, however, only in import mode. Any error messages that occur while translating a process model can be found in a separate log file, called *uid*.**XLATE.LOG**.
u	*userid*	This is the logon user ID for the Workflow database.

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 47. FDL import/export tool's return codes

Value	Description	Effect of modifications to the database
0	Successful execution	Any database modifications have been completed.
1	Information	
2	Warning	
4	Validation error	The tool has made a rollback of the transaction. The database remains unchanged.
8	Syntax error	
12	Error	
16	Input error	
20	Severe error	
24	Internal error	

Examples

FDL import examples

The following JCL examples show the use of the import options and DD statements. The example JCL jobs start the import tool, log on using the user ID *uid* and a password *pwd*:

To import an FDL file

This JCL job imports the FDL file that is specified by the DD name FMCIIMP .

```
//FMCHJRIF EXEC PROC=FMCHPBAT,PROGRAM=FMCH0IBA,
//            PARM='ENVAR("_CEE_ENVFILE=DD:FMCHEENV")/
//            -u uid -p pwd -i -o'
//*
//FMCIIMP DD  DISP=SHR,DSN=CustHLQ.SFMCDATA(FDL)
//FMCILOG DD  SYSOUT=*
//*
```

To import an FDL file and translate the contained process models

This job imports the FDL file that is specified by the DD name FMCIIMP and translates the imported process models.

```
//FMCHJRIF EXEC PROC=FMCHPBAT,PROGRAM=FMCH0IBA,
//            PARM='ENVAR("_CEE_ENVFILE=DD:FMCHEENV")/
//            -u uid -p pwd -i -o -t'
//*
//FMCIIMP DD  DISP=SHR,DSN=CustHLQ.SFMCDATA(FDL)
//FMCILOG DD  SYSOUT=*
//*
```

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

This job imports the FDL file that is specified by the DD name FMCIIMP. All information, warning, and error messages will be written to the log file that is specified by the DD name FMCILOG.

```
//FMCHJRIF EXEC PROC=FMCHPBAT,PROGRAM=FMCH0IBA,
//            PARM='ENVAR("_CEE_ENVFILE=DD:FMCHEENV")/
//            -u uid -p pwd -i -o'
//*
//FMCIIMP DD  DISP=SHR,DSN=CustHLQ.SFMCDATA(FDL)
//FMCILOG DD  DISP=SHR,DSN=CustHLQ.SFMCDATA(LOG)
//*
```

FDL export examples

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

To export all workflow entities

This JCL job export the entities from the Workflow database to the output file specified by the DD name FMCIEXP.

```
//FMCHJRIF EXEC PROC=FMCHPBAT,PROGRAM=FMCH0IBA,
//             PARM='ENVAR("_CEE_ENVFILE=DD:FMCHEENV")/
//             -u uid -p pwd -e'
//*
//FMCIEXP  DD  DISP=SHR,DSN=CustHLQ.SFMCDATA(OUT)
//FMCILOG  DD  SYSOUT=*
//*
```

To export all people

This JCL job export the definitions for all persons from the Workflow database to the output file specified by the DD name FMCIEXP.

```
//FMCHJRIF EXEC PROC=FMCHPBAT,PROGRAM=FMCH0IBA,
//            PARM='ENVAR("_CEE_ENVFILE=DD:FMCHEENV")/
//            -u uid -p pwd -e -c"EXPORT PERSON*"'
//*
//FMCIEXP  DD  DISP=SHR,DSN=CustHLQ.SFMCDATA(OUT)
//FMCILOG  DD  SYSOUT=*
//*
```

To export an individual process (deep)

This JCL job exports the definitions for the process process1 and all nested subprocesses of this process from the Workflow database to the output file specified by the DD name FMCIEXP.

```
//FMCHJRIF EXEC PROC=FMCHPBAT,PROGRAM=FMCH0IBA,
//            PARM='ENVAR("_CEE_ENVFILE=DD:FMCHEENV")/
//            -u uid -p pwd -e -c"EXPORT PROCESS process1 DEEP"'
//*
//FMCIEXP  DD  DISP=SHR,DSN=CustHLQ.SFMCDATA(OUT)
//FMCILOG  DD  SYSOUT=*
//*
```

To export Workflow entities using a command file

his job export entities from the Workflow database to the file specified by the DD name FMCIEXP, using the commands in the file that is specified by the DD name FMCICMD.

```
//FMCHJRIF EXEC PROC=FMCHPBAT,PROGRAM=FMCH0IBA,
//            PARM='ENVAR("_CEE_ENVFILE=DD:FMCHEENV")/
//            -u uid -p pwd -e -c @'
//*
//FMCICMD DD  DISP=SHR,DSN=CustHLQ.SFMCDATA(CMD)
//FMCIEXP  DD  DISP=SHR,DSN=CustHLQ.SFMCDATA(OUT)
//FMCILOG  DD  SYSOUT=*
//*
```

Translate examples

The following JCL examples show how to translate existing process models that have already been imported into the database. The example JCL jobs start the import tool, log on using the user ID *uid* and a password *pwd*:

To translate existing models

This job translates an existing process model in the MQSeries Workflow for OS/390 run-time database with the process name *process1*.

```
//FMCHJRIF EXEC PROC=FMCHPBAT,PROGRAM=FMCH0IBA,
//            PARM='ENVAR("_CEE_ENVFILE=DD:FMCHEENV")/
//            -u uid -p pwd
//            -c "TRANSLATE PROCESS process1"'
//*
//FMCILOG DD  SYSOUT=*
//*
```

To translate existing process models using a command file

This job translates the existing process models using the commands in the file that is specified by the DD name FMCICMD.

```
//FMCHJRIF EXEC PROC=FMCHPBAT,PROGRAM=FMCH0IBA,
//            PARM='ENVAR("_CEE_ENVFILE=DD:FMCHEENV")/
//            -u uid -p pwd -c @'
//*
//FMCICMD DD  DISP=SHR,DSN=CustHLQ.SFMCDATA(CMD)
//FMCILOG DD  SYSOUT=*
//*
```

Appendix G. Customization parameter file

All the customization parameters for an MQSeries Workflow for OS/390 system are entered into the customization parameter file. Each time that you create a new MQSeries Workflow for OS/390 system, you must copy and complete this file. This is done during pre-customization.

The customization parameter file template is *CustHLQ*.`SFMCDATA(FMCHECIF)`. During pre-customization (in step 2 of the task "Create input files for customization" on page 16) you will to enter your system's customization parameters from the tables in "Chapter 1. Planning your configuration" on page 3. The following step then generates the JCLs necessary to customize the Workflow system that is defined in this file.

The template file contains the following:

```
***********************************************************************
****** LIST OF PARAMETERS TO CUSTOMIZE                           ******
***********************************************************************

***********************************************************************
**** PARAMETERS FOR SCOPE INSTALLATION                             ****
***********************************************************************

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

***********************************************************************
**** PARAMETERS FOR SCOPE DOMAIN                                   ****
***********************************************************************

*NOT USED AT THE MOMENT

***********************************************************************
**** PARAMETERS FOR SCOPE SYSTEM GROUP                             ****
***********************************************************************

 MQWFSGNM ='MQWFSGNM'   *** MQWF System Group Name
 MQWFSGLC ='MQWFSGLC'   *** MQWF System Group Locale Setting
                        ***      (is used to set the locale for
                        ***       all servers and tools, use
                        ***       'C' for the default on your machine
                        ***       or a specific locale setting, e.g.
                        ***       De_DE.IBM-273 for German)

 DB2SGPRE ='DB2SGPRE'   *** DB2 System Group Object Qualifier
                        ***     (is used for the DB2 objects created
                        ***      for this System Group e.g.
                        ***      'DB2SGPRE'.TABLENAME)

 **** DB2 Storage Group for Workflow Data ****
 DB2STGNW ='DB2STGNW'   *** DB2 Storage Group Name
 DB2STGPW ='DB2STGPW'   *** DB2 Storage Group Dataset Name Prefix
 DB2STGVW ='DB2STGVW'   *** DB2 Storage Group Volume
                        ***     (VOLUME for volume name or
                        ***      '*' for SMS managed volumes)

 **** DB2 Storage Group for Audit Trail Data ****
 DB2STGNA ='DB2STGNA'   *** DB2 Storage Group Name
 DB2STGPA ='DB2STGPA'   *** DB2 Storage Group Dataset Name Prefix
 DB2STGVA ='DB2STGVA'   *** DB2 Storage Group Volume
```

```
                        ***     (VOLUME for volume name or
                        ***      '*' for SMS managed volumes)

 **** DB2 Databases ****
 DB2DBNAM ='DB2DBNAM'   *** DB2 Database Name
 DB2MDBNM ='DB2MDBNM'   *** DB2 PES Mapping Database Name
 DB2PDBNM ='DB2PDBNM'   *** DB2 PES Directory Database Name

 **** DB2 Collections ****
 DB2DBCOL ='DB2DBCOL'   *** DB2 Database Collection Name
 DB2MDCOL ='DB2MDCOL'   *** DB2 PES Mapping Database Collection Name
 DB2PDCOL ='DB2PDCOL'   *** DB2 PES Directory Database Collection Name

 DB2PLANN ='DB2PLANN'   *** DB2 Plan Name

***********************************************************************
**** PARAMETERS FOR SCOPE SYSTEM                                   ****
***********************************************************************

 MQWFUKEY ='MQWFUKEY'   *** MQWF Unique System Key
                        ***     (e.g. for started task name)
 MQWFSYSP ='MQWFSYSP'   *** MQWF System Prefix
                        ***     (e.g. part of MQSeries object naming
                        ***      and profile keys)
 MQWFSYSN ='MQWFSYSN'   *** MQWF System Name

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

 CTRCNAME ='CTRCNAME'   *** CTRACE Component Name
 CTRCPMS1 ='CTRCPMS1'   *** CTRACE Parmlib Member Suffix (Start Writer)
 CTRCPMS2 ='CTRCPMS2'   *** CTRACE Parmlib Member Suffix (Stop Writer)
 CTRCWPRC ='CTRCWPRC'   *** CTRACE Writer Procedure Name (max. 7 chars)

***********************************************************************
**** PARAMETERS FOR CUSTOMIZATION                                   ****
***********************************************************************

 **** CICS ****

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

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

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

 **** High Level Qualifiers ***

 DB2INHLQ ='DB2INHLQ'   *** DB2 Installation High Level Qualifier
 MQPREFIX ='MQPREFIX'   *** MQSeries Installation High Level Qualifier
 LELIBPFX ='LELIBPFX'   *** Language Environment High Level Qualifier
 CLIBRPFX ='CLIBRPFX'   *** C/C++ Installation High Level Qualifier
 CBLIBPFX ='CBLIBPFX'   *** Cobol Installation High Level Qualifier
 IMSLIBPX ='IMSLIBPX'   *** IMS Installation High Level Qualifier
 ICONVPFX ='ICONVPFX'   *** ICONV Installation High Level Qualifier
 IPCSPRFX ='IPCSPRFX'   *** IPCS Installation High Level Qualifier

 **** Subsystems ****

 DB2SSYSN ='DB2SSYSN'   *** DB2 Subsystem Name
 MQQMNAME ='MQQMNAME'   *** MQSeries Queue Manager Name
 CICSGRPN ='CICSGRPN'   *** CICS Group Name
```

Appendix H. Machine profile

Each MQSeries Workflow for OS/390 system has a machine profile in *CustHLQ*.`SFMCDATA(FMCHEMPR)`. This profile contains system settings that affect the operation of MQSeries Workflow for OS/390 servers and tools. Some of the values are substituted automatically during customization, these must not be changed. Changes made to the machine profile will affect new server instances and tools that are started. If you want the changes to affect all running server instances then you must restart the system as described in “Restarting the system” on page 55.

Table 48. Machine profile settings

Variable	Value may be changed?	Description
`System`	No	This value should be your value for *System* in Table 3 on page 7. This value is substituted from the customization parameter file, see “Appendix G. Customization parameter file” on page 115.
`SystemGroup`	No	This value should be your value for *SystemGroup* in Table 2 on page 6. This value is substituted from the customization parameter file, see “Appendix G. Customization parameter file” on page 115.
`DatabaseName`	No	This value should be your value for *WorkflowDatabaseName* in Table 2 on page 6. This value is substituted from the customization parameter file, see “Appendix G. Customization parameter file” on page 115.
`DbPlan`	No	This value should be your value for *DB2Plan* in Table 2 on page 6. This value is substituted from the customization parameter file, see “Appendix G. Customization parameter file” on page 115.
`DbSubSystem`	No	This value should be your value for *DB2SubSystem* in Table 5 on page 9. This value is substituted from the customization parameter file, see “Appendix G. Customization parameter file” on page 115.
`ExecutionServer OperationMode`	No	For future use.
`APITimeOut`	Tune carefully	API timeout 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, as described in “Turning tracing on” on page 87 and “Turning tracing off” on page 89. Valid values are between **`0`**`,0000,00000000` (no trace) and **`99`**`,FFFF,FFFFFFFF` (full trace).
`Language`	Yes	The three letter language code selects which language version of the MMS messages the servers will send to the OS/390 system console. Valid values are: **ENU** For mixed-case U.S. English. This is the default value. **ENP** For uppercase U.S. English. This option may be required if you are using a double-byte character set. If other languages become available in the future, they will be found as *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 clean-up servers that will be started per address space is one.

Table 48. Machine profile settings (continued)

Variable	Value may be changed?	Description
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 81.
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 81.
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. “Cannot stop servers” on page 84 describes a situation when you may wish to change this value.

After customization, your machine profile will look like the following, with your values from “Chapter 1. Planning your configuration” on page 3 automatically substituted for the identifiers shown in italics:

```
*******************************************************************
*
* Description: MQ WorkFlow machine profile.
*
*******************************************************************
*
MQWorkflowMachine.SystemQualifier.System:System
MQWorkflowMachine.SystemQualifier.SystemGroup:SystemGroup
MQWorkflowMachine.SystemQualifier.DatabaseName:WorkflowDatabaseName
MQWorkflowMachine.SystemQualifier.DbPlan:DB2Plan
MQWorkflowMachine.SystemQualifier.DbSubSystem:DB2SubSystem
MQWorkflowMachine.SystemQualifier.ExecutionServerOperationMode:Standalone
MQWorkflowMachine.SystemQualifier.APITimeOut:180000
MQWorkflowMachine.SystemQualifier.FMLConnectName:SystemGroup.System,QueueManager
MQWorkflowMachine.SystemQualifier.FMLConnectDelayTime:30
MQWorkflowMachine.SystemQualifier.FMC_TRACE_CRITERIA:00,0000,00000000
MQWorkflowMachine.SystemQualifier.Language:ENU
MQWorkflowMachine.SystemQualifier.AdminSvrsPerAS:1
MQWorkflowMachine.SystemQualifier.ClnupSvrsPerAS:1
MQWorkflowMachine.SystemQualifier.DistSvrsPerAS:1
MQWorkflowMachine.SystemQualifier.ExeSvrsPerAS:5
MQWorkflowMachine.SystemQualifier.GwySvrsPerAS:1
MQWorkflowMachine.SystemQualifier.ModelSvrsPerAS:1
MQWorkflowMachine.SystemQualifier.PESvrsPerAS:5
MQWorkflowMachine.SystemQualifier.SchedSvrsPerAS:1
MQWorkflowMachine.SystemQualifier.ServerStartProc:UniqueSystemKey
MQWorkflowMachine.SystemQualifier.WaitBetweenQInhibitAndAllowed:30
```

Appendix I. Environment variable file

Each MQSeries Workflow for OS/390 system has an environment variable file in *CustHLQ*.SFMCDATA(FMCHEENV). This file contains system settings that affect the operation of MQSeries Workflow for OS/390 servers and tools. Some of the values are substituted automatically during customization, these must not be changed. Changes made to the environment variable file will affect new server instances and tools that are started. If you want the changes to affect all running server instances then you must restart the system as described in "Restarting the system" on page 55.

Table 49. Environment variable file settings

Variable	Value may be changed?	Description
_ICONV_UCS2_PREFIX	No	Your value for *ICONVInstHLQ* in Table 4 on page 8.
LC_ALL	No	Selects the codepage to be used by the Workflow servers and tools. This is set to your value for *SystemGroupLocale*, see Table 2 on page 6 for more details.
FMC_SIMPLE_TRACE_ONLY	Yes	Activates simple tracing in newly started servers and tools. Valid values are YES or NO. For more information, see "Turning tracing on" on page 87.
FMC_SYSTEM_QUALIFIER	No	This is set to your value for *SystemQualifier* in Table 3 on page 7.
FMC_ELAPSED_TIME	No	This must be set to YES.
FMC_IENV	No	This must be set to 1.

After customization, your machine profile will look like the following, with your values from "Chapter 1. Planning your configuration" on page 3 automatically substituted for the identifiers shown in italics:

```
_ICONV_UCS2_PREFIX=ICONVInstHLQ
LC_ALL=SystemGroupLocale
FMC_SIMPLE_TRACE_ONLY=NO
FMC_SYSTEM_QUALIFIER=SystemQualifier
FMC_ELAPSED_TIME=YES
FMC_IENV=1
```

Glossary

This glossary defines terms and abbreviations used in this and other MQSeries Workflow for OS/390 publications. If you do not find the term you are looking for, refer to the index or the *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994.

A

administration console. The MQSeries Workflow component that accepts commands for starting and stopping systems and servers. It also allows the number of server instances to be queried.

administration server. The MQSeries Workflow component that performs administration functions within an MQSeries Workflow system.

administration server ID. This name is used when issuing administration console commands. It identifies which administration server will execute a given command. This identifier must be unique within the OS/390 image, and not more than 8 characters long.

activity. One of the steps that make up a process model. This can be a program activity, process activity, or block activity.

API. See *application programming interface*.

application programming interface. An interface provided by the MQSeries Workflow workflow manager that enables programs to request services from the MQSeries Workflow workflow manager. The services are provided synchronously.

B

backward mapping. Conversion of output data created by an OS/390 legacy application into an MQSeries Workflow container. This conversion is performed by the *program execution server*'s *program mapper*.

backward mapping definition. Part of the *MDL* which connects an *interface* definition and *structure* definition.

bridge. See *MQSeries bridges*.

Buildtime. An MQSeries Workflow component with a graphical user interface for creating and maintaining workflow process models, administering resources, and the system network definitions.

C

CICS bridge. See *MQSeries bridges*.

cleanup server. The MQSeries Workflow component that physically deletes information in the MQSeries Workflow run-time database, which had only been deleted logically.

container API. An MQSeries Workflow API that allows programs executing under the control of MQSeries Workflow to obtain data from the input and output container of the activity and to store data in the output container of the activity. See also *data container*.

CPIC. An *invocation type* that allows the *program execution server* to run an application synchronously on an IMS service. CPIC is based on IMS/APPC.

D

data container. Storage for the input and output data of an activity or process. See also *container API*.

data structure. A named entity that consists of a set of data structure members. Input and output containers are defined by reference to a data structure, and adopt the layout of the referenced data structure type.

data structure member. One of the variables of which a *data structure* is composed.

'DEFAULT' mapping. The *mapping type* provided by IBM.

domain. A set of MQSeries Workflow system groups which have the same meta-model, share the same staff information, and topology information. Communication between the components in the domain is via message queuing.

E

EXCI. An *invocation type* that allows the *program execution server* to run an application synchronously on a CICS service. EXCI is based on the CICS External CICS Interface provided by CICS Version 4.1 and higher to allow non-CICS applications to call programs running under CICS.

executable. The name of the program as defined to the service system.

execution user ID. The OS/390 user ID used by the *program execution server* to execute a program request.

execution server. The MQSeries Workflow component that performs the processing of process instances at runtime.

export tool. A utility program for retrieving information from the Workflow database and making it available in MQSeries Workflow Definition Language (FDL) format.

F

FDL. The MQSeries Workflow definition language used to exchange MQSeries Workflow information between MQSeries Workflow system groups. The language is used by the import and export function of MQSeries Workflow and contains the workflow definitions for staff, programs, data structures, and topology. This allows non-MQSeries Workflow components to interact with MQSeries Workflow. See also *export tool* and *import tool*.

forward mapping. Conversion of MQSeries Workflow containers into a format accepted by an OS/390 legacy application. This conversion is performed by the *program execution server*'s *program mapper*.

forward mapping definition. Part of the *MDL* which connects a *structure* definition and *interface* definition.

I

import tool. A utility program that accepts information in the Workflow definition language (FDL) format and places it in a Workflow database.

IMS bridge. See *MQSeries bridges*.

interface. The definition of the data structure accepted by an OS/390 CICS or IMS legacy application. This definition is used by the *'DEFAULT' mapping* exit to convert the data to (and from) an MQSeries Workflow program's *structure*.

interface definition. Part of the *MDL* which defines the interface used by a legacy application.

interface element. Part of an *interface* definition. An interface element has a name, a type, and a cardinality. It is mapped on to a *structure element* by a *mapping rule*.

invocation exit. The DLL specified by the *invocation type*. The exit is based on an invocation protocol like CICS *EXCI* or the *MQSeries* CICS and IMS *bridges*.

invocation protocol. The way the PES connects to a service like CICS or IMS in order to invoke a program on that service system.

invocation type. The name used to identify the invocation exit to use. An invocation type must be defined in the program execution server directory, where it is associated with one or more services. In the process model, an invocation type must also be associated with each program that the PES is to be able to invoke.

L

local user. The RACF user under which the program is executed.

M

mapping definition language. The language used to define *mapping rules* for the *'DEFAULT' mapping* exit.

mapping exit. Used by the PES to convert data between MQSeries Workflow and legacy applications. The exit is identified by a mapping type defined in the *PES directory* and in *Buildtime*. The exit is only called if mapping has been enabled in *Buildtime*.

mapping rules. Part of a *forward mapping* or *backward mapping* definition that defines the mapping between individual *interface elements* and *structure elements*. Mapping rules are defined using the *mapper definition language*.

mapping type. The name used to identify which mapping exit to use. The mapping type is defined in the *PES directory* and must match the *Buildtime* definitions for the legacy application. The mapping type provided with MQSeries Workflow for OS/390 is named 'DEFAULT'.

MDL. See *mapping definition language*.

message queuing. A communication technique provided 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 OS/390. This product; extending IBM's business process automation to the OS/390 platform. This term is always used to distinguish it from MQSeries Workflow for other platforms.

P

PES. See *program execution server*.

PES directory. See *program execution server directory*.

process activity. An activity that is part of a process model. When a process activity is executed, an instance of the process model is created and executed.

process definition. See *process model*.

process model. A set of processes represented in a process model. The processes are represented in graphical form in the process diagram. The process model contains the definitions for staff, programs, and data structures associated with the activities of the process. After having translated the process model into a process template, the process template can be executed over and over again.

program execution server. The MQSeries Workflow for OS/390 component that manages the invocation of programs running on OS/390.

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

program mapping import tool. Component of the MQSeries Workflow program mapping exit which reads the result of the program mapping parser and inputs the compiled program mapping definitions into the program mapping DB.

program mapping parser. Component of the MQSeries Workflow for OS/390 program mapping exit which parses the MDL and creates an intermediate file which is used by the program mapping import tool.

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.

structure. The definition of the MQSeries Workflow structure passed into or out of an *activity* implementation. This definition is used by the *'DEFAULT' mapping* exit to convert the data to (and from) a legacy application's *interface*.

structure element. Part of a *structure* definition. A structure element has a name, a type, and a cardinality. It is mapped on to an *interface element* by a *mapping rule*.

system. The smallest MQSeries Workflow unit within an MQSeries Workflow domain. It consists of a set of the MQSeries Workflow servers: one administration server, one or more execution server instances, and zero or more program execution server instances, and optionally, one scheduling server and/or one clean-up server.

system group. Each system group needs its own database, and contains one *system*. Multiple system groups can share the same DB2 subsystem.

T

translate. The action that converts a process model into a run-time process template.

U

user ID. An alphanumeric string that uniquely identifies an MQSeries Workflow user. MQSeries Workflow for OS/390 handles two types of user IDs, (1) MQSeries Workflow user IDs. (2) *Execution user 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.

W

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*.

Bibliography

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

MQSeries Workflow for OS/390 publications

This section lists the publications included in the MQSeries Workflow for OS/390 library.

- *MQSeries Workflow for OS/390: Customization and Administration*, SC33-7030, explains how to customize and administer an MQSeries Workflow for OS/390 system.
- *MQSeries Workflow for OS/390: Programming*, SC33-7031, explains the C and Cobol application programming interfaces (APIs), and the program exits.
- *MQSeries Workflow for OS/390: Messages*, SC33-7032, explains the MQSeries Workflow for OS/390 system messages.
- *MQSeries Workflow for OS/390: Program Directory*, GI10-0483, explains how to install MQSeries Workflow for OS/390.

MQSeries Workflow publications

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

- *IBM MQSeries Workflow: List of Workstation Server Processor Groups*, GH12-6357, lists the processor groups for MQSeries Workflow.
- *IBM MQSeries Workflow: Concepts and Architecture*, GH12-6285, explains the basic concepts of MQSeries Workflow. It also describes the architecture of MQSeries Workflow and how the components fit together.
- *IBM MQSeries Workflow: Getting Started with Buildtime*, SH12-6286, describes how to use Buildtime of MQSeries Workflow.
- *IBM MQSeries Workflow: Getting Started with Runtime*, SH12-6287, describes how to get started with the Client.
- *IBM MQSeries Workflow: Programming Guide*, SH12-6291, explains the application programming interfaces (APIs).
- *IBM MQSeries Workflow: Installation Guide*, SH12-6288, contains information and procedures for installing and customizing MQSeries Workflow.
- *IBM MQSeries Workflow: Administration Guide*, SH12-6289, explains how to administer an MQSeries Workflow system.

Workflow publications

- *IBM Systems Journal, Vol. 36. No. 1, 1997 by Frank Leymann, Dieter Roller*, you can also refer to the Internet: http://www.almaden.ibm.com/journal/sj361/leymann.html
- *Workflow Handbook 1997, published in association with WfMC*, edited by Peter Lawrence

Other useful publications

- *MQSeries Clients*, GC22-1632.
- *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.

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

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

IBM MQSeries Workflow for OS/390
Customization and Administration
Version 3 Release 1

Publication No. SC33-7030-00

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
SC33-7030-00

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
Department 3248
Schoenaicher Strasse 220
71032 Boeblingen
Germany

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

SC33-7030-00

Cut or Fold
Along Line

Program Number: 5655–A96

Printed in Denmark by IBM Danmark A/S

SC33-7030-00