WebSphere MQ for z/OS

System Administration Guide

Version 7.0

IBM

WebSphere MQ for z/OS

IBM

System Administration Guide

Version 7.0

First edition (April 2008)

This edition of the book applies to:

- IBM WebSphere MQ for z/OS , Version 7.0

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

Contents

Figures

Tables

Chapter 1. Operating WebSphere MQ for z/OS

Operating WebSphere MQ for z/OS

Use these basic procedures to operate WebSphere® MQ for z/OS®.

You can also perform the operations described in this section using the WebSphere MQ Explorer, which is distributed with WebSphere MQ for Windows® and WebSphere MQ for Linux® (x86 platform). See Administration using the WebSphere MQ Explorer in the *System Administration Guide* for more information.

This chapter discusses the following topics:
- "Issuing commands"
- "Introducing the operations and control panels" on page 3
- "Using the operations and control panels" on page 11
- "Defining objects" on page 13
- "Defining a local queue" on page 13
- "Defining other types of objects" on page 14
- "Working with object definitions" on page 14
- "Working with namelists" on page 15

Issuing commands

You can control most of the operational environment of WebSphere MQ using the WebSphere MQ commands. WebSphere MQ for z/OS supports both the MQSC and PCF types of these commands. This book describes how to specify attributes using MQSC commands, and so it refers to those commands and attributes using their MQSC command names, rather than their PCF names. For details of the syntax of the MQSC commands, see WebSphere MQ Script (MQSC) Command Reference. For details of the syntax of the PCF commands, see WebSphere MQ Programmable Command Formats and Administration Interface. If you are a suitably authorized user, you can issue WebSphere MQ commands from:
- The initialization input data sets (described in the WebSphere MQ for z/OS System Setup Guide).
- A z/OS console, or equivalent, such as SDSF
- The z/OS master get command routine, MGCRE (SVC 34)
- The WebSphere MQ utility, CSQUTIL (described in "WebSphere MQ utility program (CSQUTIL)" on page 159.)
- A user application, which can be:
  - A CICS® program
  - A TSO program
  - A z/OS batch program
  - An IMS™ program

  See "Writing programs to administer WebSphere MQ" on page 20 for information about this.

Much of the functionality of these commands is provided in a user-friendly way by the operations and control panels, accessible from TSO and ISPF, and described in "Introducing the operations and control panels" on page 3.

Issuing commands from a z/OS console or its equivalent

You can issue all WebSphere MQ commands from a z/OS console or its equivalent. This means you can also issue WebSphere MQ commands from anywhere where you can issue z/OS commands, such as SDSF or by a program using the MGCRE macro.

The maximum amount of data that can be displayed as a result of a command typed in at the console is 32 KB.

Note:

1. You cannot issue WebSphere MQ commands using the IMS/SSR command format from an IMS terminal. This function is not supported by the IMS adapter.
2. The input field provided by SDSF might not be long enough for some commands, particularly those for channels.

Command prefix strings:

Each WebSphere MQ command must be prefixed with a command prefix string (CPF), as shown in Figure 1.

Because more than one WebSphere MQ subsystem can run under z/OS, the CPF is used to indicate which WebSphere MQ subsystem processes the command. For example, to start the queue manager for a subsystem called CSQ1, whose CPF is '+CSQ1', you issue the command +CSQ1 START QMGR from the operator console. This CPF must be defined in the subsystem name table (for the subsystem CSQ1). This is described in the WebSphere MQ for z/OS System Setup Guide. In the examples, the string '+CSQ1' is used as the command prefix.

Using the z/OS console to issue commands:

You can type simple commands from the z/OS console, for example, the DISPLAY command in Figure 1. However, for complex commands or for sets of commands that you issue frequently, the other methods of issuing commands are better.

```
+CSQ1 DISPLAY QUEUE(TRANSMIT.QUEUE.PROD) TYPE(QLOCAL)
```

Figure 1. Issuing a DISPLAY command from the z/OS console

Command responses:

Direct responses to commands are sent to the console that issued the command. WebSphere MQ supports the *Extended Console Support* (EMCS) function available in z/OS, and therefore consoles with 4-byte IDs can be used. Additionally, all commands except START QMGR and STOP QMGR support the use of Command and Response Tokens (CARTs) when the command is issued by a program using the MGCRE macro.

Issuing commands from the utility program CSQUTIL

You can issue commands from a sequential data set using the COMMAND function of the utility program CSQUTIL. This utility transfers the commands, as messages, to the *system-command input queue* and waits for the response, which is printed together with the original commands in SYSPRINT. For details of this, see "WebSphere MQ utility program (CSQUTIL)" on page 159.

Introducing the operations and control panels

You can use the WebSphere MQ operations and control panels to perform administration tasks on WebSphere MQ objects. You use these panels for defining, displaying, altering, or deleting WebSphere MQ objects. Use the panels for day-to-day administration and for making small changes to objects. If you are setting up or changing many objects, you should use the COMMAND function of the CSQUTIL utility program.

The operations and control panels support the controls for the channel initiator (for example, to start a channel or a TCP/IP listener), for clustering, and for security. They also enable you to display information about threads and page set usage.

The panels work by sending MQSC type WebSphere MQ commands to a queue manager, through the system command input queue.

Note:
1. There are no panels for the direct manipulation of topic objects or subscriptions. Using a command facility, you can adminster publish/subscribe definitions and other system controls that are not directly available from other panels.
2. You cannot issue the WebSphere MQ commands directly from the command line in the panels.
3. To use the operations and control panels, you must have the correct security authorization; this is described in the WebSphere MQ for z/OS System Setup Guide.

Invoking the operations and control panels

If the ISPF/PDF primary options menu has been updated for WebSphere MQ, you can access the WebSphere MQ operations and control panels from that menu. For details about updating the menu, see the WebSphere MQ for z/OS System Setup Guide.

You can access the WebSphere MQ operations and control panels from the TSO command processor panel (usually option 6 on the ISPF/PDF primary options menu). The name of the exec that you run to do this is CSQOREXX. It has two parameters; `thlqual` is the high-level qualifier for the WebSphere MQ libraries to be used, and `langletter` is the letter identifying the national language libraries to be used (for example, E for U.S. English). The parameters can be omitted if the WebSphere MQ libraries are permanently installed in your ISPF setup. Alternatively, you can issue CSQOREXX from the TSO command line.

These panels are designed to be used by operators and administrators with a minimum of formal training. Read these instructions with the panels running and try out the different tasks suggested.

Note: While using the panels, temporary dynamic queues with names of the form SYSTEM.CSQOREXX.* are created.

Rules for the operations and control panels

The WebSphere MQ Script (MQSC) Command Reference manual defines the general rules for WebSphere MQ character strings and names. However, there are some rules that apply only to the operations and control panels:

- Do not enclose strings, for example descriptions, in single or double quotes.
- If you need to use a quote mark in a description or other text field, for example:

```
This is Maria's queue
```

      use just one quote. The panel processor doubles them for you to pass them to WebSphere MQ. However, if it has to truncate your data to do this, it does so.

- You can use uppercase or lowercase characters in most fields, and they are folded to uppercase characters when you press Enter. The exceptions are:
  - Storage class names and Coupling Facility structure names, which must start with uppercase A through Z and be followed by uppercase A through Z or numeric characters.
  - Certain fields that are not translated. These include:
    - Application ID
    - Description
    - Environment data
    - Object names (but if you use a lowercase object name, you might not be able to enter it at a z/OS console)
    - Remote system name
    - Trigger data
    - User data
- In names, leading blanks and leading underscores are ignored. Therefore, you cannot have object names beginning with blanks or underscores.
- Underscores are used to show the extent of blank fields. When you press Enter, trailing underscores are replaced by blanks.
- Many description and text fields are presented in multiple parts, each part being handled by WebSphere MQ independently. This means that trailing blanks *are retained* and the text is not contiguous.

Blank fields:

When you specify the **Define** action for a WebSphere MQ object, each field on the define panel contains a value. See the general help (extended help) for the display panels for information on where WebSphere MQ gets the values. If you type over a field with blanks, and blanks are not allowed, WebSphere MQ puts the installation default value in the field or prompts you to enter the required value.

When you specify the **Alter** action for a WebSphere MQ object, each field on the alter panel contains the current value for that field. If you type over a field with blanks, and blanks are not allowed, the value of that field is left unchanged.

Objects and actions

The operations and control panels offer you many different types of object and a number of actions that you can perform on them. The actions are listed on the initial panel and enable you to manipulate the objects and display information about them. These objects include all the WebSphere MQ objects, together with some extra ones. The objects fall into the following categories.

- Queues, processes, authentication information objects, namelists, storage classes and CF structures
- Channels
- Cluster objects
- Queue manager and security
- Connections
- System

Queues, processes, authentication information objects, namelists, storage classes and CF structures:

These are the basic WebSphere MQ objects. There can be many of each type. They can be listed, listed with filter, defined, and deleted, and have attributes that can be displayed and altered, using the LIST or DISPLAY, LIST with FILTER, DEFINE LIKE, MANAGE, and ALTER actions. (Objects are deleted using the MANAGE action.)

This category consists of the following objects:

QLOCAL	Local queue
QREMOTE	Remote queue
QALIAS	Alias queue for indirect reference to a queue
QMODEL	Model queue for defining queues dynamically
QUEUE	Any type of queue
QSTATUS	Status of a local queue
PROCESS	Information about an application to be started when a trigger event occurs
AUTHINFO	Authentication information: definitions required to perform Certificate Revocation List (CRL) checking using LDAP servers
NAMELIST	List of names, such as queues or clusters
STGCLASS	Storage class
CFSTRUCT	Coupling Facility (CF) structure
CFSTATUS	Status of a CF structure

Channels:

Channels are used for distributed queuing. There can be many of each type, and they can be listed, listed with filter, defined, deleted, displayed, and altered. They also have other functions available using the START, STOP and PERFORM actions. PERFORM provides reset, ping, and resolve channel functions.

This category consists of the following objects:

CHANNEL	Any type of channel
SENDER	Sender channel
SERVER	Server channel
RECEIVER	Receiver channel
REQUESTER	Requester channel
CLUSRCVR	Cluster-receiver channel
CLUSSDR	Cluster-sender channel
SVRCONN	Server-connection channel
CLNTCONN	Client-connection channel
CHSTATUS	Status of a channel connection

Cluster objects:

Cluster objects are created automatically for queues and channels that belong to a cluster. The base queue and channel definitions can be on another queue manager. There can be many of each type, and names can be duplicated. They can be listed, listed with filter, and displayed. PERFORM, START, and STOP are also available through the LIST actions.

This category consists of the following objects:

CLUSQ	Cluster queue, created for a queue that belongs to a cluster
CLUSCHL	Cluster channel, created for a channel that belongs to a cluster
CLUSQMGR	Cluster queue manager, the same as a cluster channel but identified by its queue manager name

Cluster channels and cluster queue managers do have the PERFORM, START and STOP actions, but only indirectly through the DISPLAY action.

Queue manager and security:

Queue manager and security objects have a single instance. They can be listed, and have attributes that can be displayed and altered (using the LIST or DISPLAY, and ALTER actions), and have other functions available using the PERFORM action.

This category consists of the following objects:

MANAGER	Queue manager – the PERFORM action provides suspend and resume cluster functions
SECURITY	Security functions – the PERFORM action provides refresh and reverify functions

Connection:

Connections can be listed, listed with filter and displayed.

This category consists only of the connection object, CONNECT.

System:

A collection of other functions. This category consists of the following objects:

SYSTEM   System functions

CONTROL   Synonym for SYSTEM

The functions available are:

LIST or   Display queue-sharing group, distributed queuing, page set, or data set
DISPLAY   usage information.

PERFORM   Refresh or reset clustering

START   Start the channel initiator or listeners

STOP   Stop the channel initiator or listeners

Actions:

The actions that you can perform for each type of object are shown in the
following table:

Table 1. Valid operations and control panel actions for WebSphere MQ objects

Object	Alter	Define like	Manage (1)	List or Display	List with Filter	Perform	Start	Stop
AUTHINFO	X	X	X	X	X			
CFSTATUS				X				
CFSTRUCT	X	X	X	X	X			
CHANNEL	X	X	X	X	X	X	X	X
CHSTATUS				X	X			
CLNTCONN	X	X	X	X	X			
CLUSCHL				X	X	X(2)	X(2)	X(2)
CLUSQ				X	X			
CLUSQMGR				X	X	X(2)	X(2)	X(2)
CLUSRCVR	X	X	X	X	X	X	X	X
CLUSSDR	X	X	X	X	X	X	X	X
CONNECT				X	X			
CONTROL				X		X	X	X
MANAGER	X			X		X		
NAMELIST	X	X	X	X	X			
PROCESS	X	X	X	X	X			
QALIAS	X	X	X	X	X			
QLOCAL	X	X	X	X	X			
QMODEL	X	X	X	X	X			
QREMOTE	X	X	X	X	X			
QSTATUS				X	X			
QUEUE	X	X	X	X	X			
RECEIVER	X	X	X	X	X	X	X	X
REQUESTER	X	X	X	X	X	X	X	X

Table 1. Valid operations and control panel actions for WebSphere MQ objects (continued)

Object	Alter	Define like	Manage (1)	List or Display	List with Filter	Perform	Start	Stop
SECURITY	X			X		X		
SENDER	X	X	X	X	X	X	X	X
SERVER	X	X	X	X	X	X	X	X
SVRCONN	X	X	X	X	X		X	X
STGCLASS	X	X	X	X	X			
SYSTEM				X		X	X	X
Note:								
1. Provides **Delete** and other functions								
2. Using the **List or Display** action								

Object dispositions

You can specify the *disposition* of the object with which you need to work. The disposition signifies where the object **definition** is kept, and how the object behaves.

The disposition is significant only if you are working with any of the following object types:

* queues
* channels
* processes
* namelists
* storage classes
* authentication information objects

If you are working with other object types, the disposition is disregarded.

Permitted values are:

Q   QMGR. The object definitions are on the page set of the queue manager and are accessible only by the queue manager.

C   COPY. The object definitions are on the page set of the queue manager and are accessible only by the queue manager. They are local copies of objects defined as having a disposition of GROUP.

P   PRIVATE. The object definitions are on the page set of the queue manager and are accessible only by the queue manager. The objects have been defined as having a disposition of QMGR or COPY.

G   GROUP. The object definitions are in the shared repository, and are accessible by all queue managers in the queue-sharing group.

S   SHARED. This disposition applies only to local queues. The queue definitions are in the shared repository, and are accessible by all queue managers in the queue-sharing group.

A   ALL. If the action queue manager is either the target queue manager, or *, objects of **all** dispositions are included; otherwise, objects of QMGR and COPY dispositions only are included. This is the default.

Choosing a queue manager

While you are viewing the initial panel, you are not connected to any queue manager. However, as soon as you press Enter, you are connected to the queue manager, or a queue manager in the queue-sharing group named in the **Connect name** field. You can leave this field blank; this means you are using the default queue manager for batch applications. This is defined in CSQBDEFV (see the WebSphere MQ for z/OS System Setup Guide for information about this).

Use the **Target queue manager** field to specify the queue manager where the actions you request are to be performed. If you leave this field blank, it defaults to the queue manager specified in the **Connect name** field. You can specify a target queue manager that is not the one you connect to. In this case, you would normally specify the name of a remote queue manager object that provides a queue manager alias definition (the name is used as the *ObjectQMgrName* when opening the command input queue). To do this, you must have suitable queues and channels set up to access the remote queue manager.

The **Action queue manager** allows you to specify a queue manager that is in the same queue-sharing group as the queue manager specified in the **Target queue manager** field to be the queue manager where the actions you request are to be performed. If you specify * in this field, the actions you request are performed on all queue managers in the queue-sharing group. If you leave this field blank, it defaults to the value specified in the **Target queue manager** field. The **Action queue manager** field corresponds to using the CMDSCOPE command modifier described in the WebSphere MQ Script (MQSC) Command Reference.

Queue manager defaults:

If you leave any queue manager fields blank, or choose to connect to a queue-sharing group, a secondary window appears when you press Enter. This window confirms the names of the queue managers you will be using. Press Enter to continue. When you return to the initial panel after having made some requests, you find fields filled in with the actual names.

Queue manager levels:

The Operations and Control panels work satisfactorily only with queue managers that are running on z/OS, and whose command level matches that of the panels, currently 600 or 700.

If these conditions are not met, it is likely that actions work only partially, incorrectly, or not at all, and that the replies from the queue manager are not recognized.

If the action queue manager is not at command level 700, some fields are not displayed, and some values cannot be entered. A few objects and actions are disallowed. In such cases, a secondary window appears asking for you to confirm that you want to proceed.

Using the function keys

To use the panels, you must use the function keys or enter the equivalent commands in the command area. The function keys have special settings for WebSphere MQ. (This means that you cannot use the ISPF default values for the function keys; if you have previously used the KEYLIST OFF ISPF command

anywhere, you must type KEYLIST ON in the command area of any operations and control panel and then press Enter to enable the WebSphere MQ settings.)

These function key settings can be displayed on the panels, as shown in Figure 2 on page 11. If the settings are not shown, type PFSHOW in the command area of any operations and control panel and then press Enter. To remove the display of the settings, use the command PFSHOW OFF.

The function key settings in the operations and control panels conform to CUA® standards. Although you can change the key setting through normal ISPF procedures (such as the KEYLIST utility) you are not recommended to do so.

Note: Using the PFSHOW and KEYLIST commands affects any other logical ISPF screens that you have, and their settings remain when you leave the operations and control panels.

Processing your actions:

Press Enter to carry out the action requested on a panel. The information from the panel is sent to the queue manager for processing.

Each time you press Enter in the panels, WebSphere MQ generates one or more operator messages. If the operation was successful, you get confirmation message CSQ9022I, otherwise you get some error messages.

Displaying WebSphere MQ user messages:

Press function key F10 in any panel to see the WebSphere MQ user messages.

Cancelling your actions:

On the initial panel, both F3 and F12 exit the operations and control panels and return you to ISPF. No information is sent to the queue manager.

On any other panel, press function keys F3 or F12 to leave the current panel **ignoring any data you have typed since last pressing Enter**. Again, no information is sent to the queue manager.
- F3 takes you straight back to the initial panel.
- F12 takes you back to the previous panel.

Getting help:

Each panel has help panels associated with it. The help panels use the ISPF protocols:
- Press function key F1 on any panel to see general help (extended help) about the task.
- Press function key F1 with the cursor on any field to see specific help about that field.
- Press function key F5 from any field help panel to get the general help.
- Press function key F3 to return to the base panel, that is, the panel from which you pressed function key F1.
- Press function key F6 from any help panel to get help about the function keys.

If the help information carries on into a second or subsequent pages, a **More** indicator is displayed in top right of the panel. Use these function keys to navigate through the help pages:

- F11 to get to the next help page (if there is one).
- F10 to get back to the previous help page (if there is one).

Using the command line

You never need to use the command line to issue the commands used by the operations and control panels because they are available from function keys, as described above. The command line is provided to allow you to enter normal ISPF commands (like PFSHOW).

The command line is initially displayed at the bottom of the panels, regardless of what ISPF settings you have. If you prefer it to be at the top, use the SETTINGS ISPF command from any of the operations and control panels to change it. The settings are remembered for subsequent sessions with the operations and control panels.

Using the operations and control panels

Figure 2 shows the panel that is displayed when you start a panel session.

```
                   IBM WebSphere MQ for z/OS - Main Menu

Complete fields. Then press Enter.

Action  . . . . . . . . . . 1      0. List with filter   4. Manage
                                   1. List or Display    5. Perform
                                   2. Define like        6. Start
                                   3. Alter              7. Stop
                                   8. Command
Object type . . . . . . . . CHANNEL       +
Name  . . . . . . . . . . . *
Disposition . . . . . . . . A   Q=Qmgr, C=Copy, P=Private, G=Group,
                                 S=Shared, A=All

Connect name  . . . . . . . MQ1C  - local queue manager or group
Target queue manager  . . . MQ1C
           - connected or remote queue manager for command input
Action queue manager  . . . MQ1C  - command scope in group
Response wait time  . . . . 30    5 - 999 seconds

(C) Copyright IBM Corporation 1993,2008. All rights reserved.

Command ===>
 F1=Help     F2=Split    F3=Exit     F4=Prompt    F9=SwapNext F10=Messages
F12=Cancel
```

Figure 2. The WebSphere MQ operations and control initial panel

From this panel you can perform actions such as:

- Choose the local queue manager you want and whether you want the commands issued on that queue manager, on a remote queue manager, or on another queue manager in the same queue-sharing group as the local queue manager. Overtype the queue manager name if you need to change it.
- Select the action you want to perform by typing in the appropriate number in the **Action** field.

- Specify the object type that you want to work with. Press function key F1 for help about the object types if you are not sure what they are.
- Specify the disposition of the object type that you want to work with.
- Display a list of objects of the type specified. Type in an asterisk (*) in the **Name** field and press Enter to display a list of objects (of the type specified) that have already been defined on the action queue manager. You can then select one or more objects to work with in sequence. All the actions are available from the list.

Note: You are recommended to make choices that result in a list of objects being displayed, and then work from that list. Use the **Display** action, because that is allowed for all object types.

Using the Command Facility

Use the editor to enter or amend MQSC commands to be passed to the queue manager.

From the primary panel, CSQOPRIA, select option **8 Command**, to start the Command Facility.

You are presented with an edit session of a sequential file, *prefix*.CSQUTIL.COMMANDS, used as input to the CSQUTIL COMMAND function; see "Issuing commands to WebSphere MQ (COMMAND)" on page 171.

You do not need to prefix commands with the command prefix string (CPF).

You can continue MQSC commands on subsequent lines by terminating the current line with the continuation characters **+** or **-**. Alternatively, use lineedit mode to provide long MQSC commands or the values of long attribute values within the command.

lineedit

To use lineedit, move the cursor to the appropriate line in the edit panel and use **F4** to display a single line in a scrollable panel. A single line can be up to 32 760 bytes of data.

To leave lineedit:
- **F3 exit** saves changes made to the line and exits
- **F12 cancel** returns to the edit panel discarding changes made to the line.

To discard changes made in the edit session, use **F12 cancel** to terminate the edit session leaving the contents of the file unchanged. Commands are not executed.

Executing commands

When you have finished entering MQSC commands, terminate the edit session with **F3 exit** to save the contents of the file and invoke CSQUTIL to pass the commands to the queue manager. The output from command processing is held in file *prefix*.CSQUTIL.OUTPUT. An edit session opens automatically on this file so that you can view the responses. Press **F3 exit** to exit this session and return to the main menu.

Defining objects

To define a new object, use an existing definition as the basis for it. You can do this in one of three ways:

- By selecting an object that is a member of a list displayed as a result of options selected on the initial panel. You then enter action type 2 (**Define like**) in the action field to the left of the selected object. Your new object has the attributes of the selected object, except the disposition. You can then change any attributes in your new object as you require.

- On the initial panel, select the **Define like** action type, enter the type of object that you are defining in the **Object type** field, and enter the name of a specific existing object in the **Name** field. Your new object has the same attributes as the object you named in the **Name** field, except the disposition. You can then change any attributes in your new object definition as you require.

- By selecting the **Define like** action type, specifying an object type and then leaving the **Name** field blank. You can then define your new object and it has the default attributes defined for your installation. You can then change any attributes in your new object definition as you require.

Note: You do not enter the name of the object you are defining on the initial panel, but on the **Define** panel you are presented with.

Defining a local queue

To define a local queue object from the operations and control panels, use an existing queue definition as the basis for your new definition. There are several panels to complete. When you have completed all the panels and you are satisfied that the attributes are correct, press Enter to send your definition to the queue manager, which then creates the actual queue.

Use the **Define like** action either on the initial panel or against an object entry in a list displayed as a result of options selected on the initial panel.

For example, starting from the initial panel, complete these fields:

Action	2 (Define like)
Object type	QLOCAL
Name	QUEUE.YOU.LIKE. This is the name of the queue that provides the attributes for your new queue.

Press Enter to display the **Define a Local Queue** panel. The queue name field is blank so that you can supply the name for the new queue. The description is that of the queue upon which you are basing this new definition. Overtype this field with your own description for the new queue.

The values in the other fields are those of the queue upon which you are basing this new queue, except the disposition. You can overtype these fields as you require. For example, type Y in the **Put enabled** field (if it is not already Y) if suitably authorized applications can put messages on this queue.

You get field help by moving the cursor into a field and pressing function key F1. Field help provides information about the values that can be used for each attribute.

When you have completed the first panel, press function key F8 to display the second panel.

Hints:

1. Do *not* press Enter at this stage, otherwise the queue will be created before you have a chance to complete the remaining fields. (If you do press Enter prematurely, do not worry; you can always alter your definition later on.)

2. Do not press function keys F3 or F12, or the data you typed will be lost.

Press function key F8 repeatedly to see and complete the remaining panels, including the trigger definition, event control, and backout reporting panels.

When your local queue definition is complete

When your definition is complete, press Enter to send the information to the queue manager for processing. The queue manager creates the queue according to the definition you have supplied. If you do not want the queue to be created, press function key F3 to exit and cancel the definition.

Defining other types of objects

To define other types of object, use an existing definition as the base for your new definition as explained in "Defining a local queue" on page 13.

Use the **Define like** action either on the initial panel or against an object entry in a list displayed as a result of options selected on the initial panel.

For example, starting from the initial panel, complete these fields:

Action	2 (Define like)
Object type	QALIAS, NAMELIST, PROCESS, CHANNEL, and so on.
Name	Leave blank or enter the name of an existing object of the same type.

Press Enter to display the corresponding DEFINE panels. Complete the fields as required and then press Enter again to send the information to the queue manager.

Like defining a local queue, defining another type of object generally requires several panels to be completed. Defining a namelist requires some additional work, as described in "Working with namelists" on page 15.

Working with object definitions

When an object has been defined, you can specify an action in the **Action** field, to alter, display, or manage it.

In each case, you can either:

- Select the object you want to work with from a list displayed as a result of options selected on the initial panel. For example, having entered 1 in the **Action** field to display objects, Queue in the **Object type** field, and * in the **Name** field, you are presented with a list of all queues defined in the system. You can then select from this list the queue with which you need to work.
- Start from the initial panel, where you specify the object you are working with by completing the **Object type** and **Name** fields.

Altering an object definition

To alter an object definition, specify action 3 and press Enter to see the ALTER panels. These panels are very similar to the DEFINE panels. You can alter the values you want. When your changes are complete, press Enter to send the information to the queue manager.

Displaying an object definition

If you want to see the details of an object without being able to change them, specify action 1 and press Enter to see the DISPLAY panels. Again, these panels are similar to the DEFINE panels except that you cannot change any of the fields. Change the object name to display details of another object.

Deleting an object

To delete an object, specify action 4 (Manage) and the **Delete** action is one of the actions presented on the resulting menu. Select the **Delete** action.

You are asked to confirm your request. If you press function key F3 or F12, the request is cancelled. If you press Enter, the request is confirmed and passed to the queue manager. The object you specified is then deleted.

Note: You cannot delete most types of channel object unless the channel initiator is started.

Working with namelists

When working with namelists, proceed as you would for other objects.

For the actions DEFINE LIKE or ALTER, press function key F11 to add names to the list or to change the names in the list. This involves working with the ISPF editor and all the normal ISPF edit commands are available. Enter each name in the namelist on a separate line.

When you use the ISPF editor in this way, the function key settings are the normal ISPF settings, and **not** those used by the other operations and control panels.

If you need to specify lowercase names in the list, specify CAPS(OFF) on the editor panel command line. When you do this, all the namelists that you edit in the future are in lowercase until you specify CAPS(ON).

When you have finished editing the namelist, press function key F3 to end the ISPF edit session. Then press Enter to send the changes to the queue manager.

Attention: If you do not press Enter at this stage but press function key F3 instead, you lose any updates that you have typed in.

Starting and stopping a queue manager

This chapter describes how to start and stop a queue manager. It discusses the following topics:
- "Before you start WebSphere MQ" on page 16
- "Starting a queue manager" on page 16
- "Stopping a queue manager" on page 18

Starting and stopping a queue manager is relatively straightforward. When a queue manager stops under normal conditions, its last action is to take a termination checkpoint. This checkpoint, and the logs, give the queue manager the information it needs to restart.

This section discusses the START and STOP commands, and contains a brief overview of start up after an abnormal termination has occurred.

Before you start WebSphere MQ

After you have installed WebSphere MQ, it is defined as a formal z/OS subsystem. This message appears during any initial program load (IPL) of z/OS:

```
CSQ3110I +CSQ1 CSQ3UR00 - SUBSYSTEM ssnm INITIALIZATION COMPLETE
```

where *ssnm* is the WebSphere MQ subsystem name.

From now on, you can start the queue manager for that subsystem *from any z/OS console that has been authorized to issue system control commands*; that is, a z/OS SYS command group. You must issue the START command from the authorized console, you cannot issue it through JES or TSO.

If you are using queue-sharing groups, you must start RRS first, and then DB2®, before you start the queue manager.

Starting a queue manager

You start a queue manager by issuing a START QMGR command. However, you cannot successfully use the START command unless you have appropriate authority. See the WebSphere MQ for z/OS System Setup Guide for information about WebSphere MQ security. Figure 3 shows examples of the START command. (Remember that you must prefix a WebSphere MQ command with a command prefix string (CPF).)

```
+CSQ1 START QMGR

+CSQ1 START QMGR PARM(NEWLOG)
```

Figure 3. Starting the queue manager from a z/OS console. The second example specifies a system parameter module name.

See the WebSphere MQ Script (MQSC) Command Reference manual for information about the syntax of the START QMGR command.

You cannot run the queue manager as a batch job or start it using a z/OS command START. These methods are likely to start an address space for WebSphere MQ that then ends abnormally. Nor can you start a queue manager from the CSQUTIL utility program or a similar user application.

You can, however, start a queue manager from an APF-authorized program by passing a START QMGR command to the z/OS MGCRE (SVC 34) service.

If you are using queue-sharing groups, the associated DB2 systems and RRS should be active when you start the queue manager.

Start options

When you start a queue manager, a system parameter module is loaded. You can specify the name of the system parameter module in one of two ways:

- With the PARM parameter of the +cpf START QMGR command, for example

  ```
  +cpf START QMGR PARM(CSQ1ZPRM)
  ```

- With a parameter in the startup procedure, for example, code the JCL EXEC statement as

  ```
  //MQM   EXEC PGM=CSQYASCP,PARM='ZPARM(CSQ1MSTR)'
  ```

A system parameter module provides information specified when the queue manager was customized. In Chapter 7, "User messages on startup," on page 225, the user message CSQY001I indicates the name of the system parameter module that was used, in this case, CSQ1ZPRM. For more information about this, see the WebSphere MQ for z/OS System Setup Guide.

You can also use the ENVPARM option to substitute one or more parameters in the JCL procedure for the queue manager.

For example, you can update your queue manager startup procedure, so that the DDname CSQINP2 is a variable. This means that you can change the CSQINP2 DDname without changing the startup procedure. This is very useful for implementing changes, providing backouts for operators, and so on.

Suppose your start-up procedure for queue manager CSQ1 looked like Figure 4.

```
//CSQ1MSTR PROC INP2=NORM
//MQMESA   EXEC PGM=CSQYASCP
//STEPLIB  DD   DISP=SHR,DSN=thlqual.SCSQANLE
//         DD   DISP=SHR,DSN=thlqual.SCSQAUTH
//         DD   DISP=SHR,DSN=db2qual.SDSNLOAD
//BSDS1    DD   DISP=SHR,DSN=myqual.BSDS01
//BSDS2    DD   DISP=SHR,DSN=myqual.BSDS02
//CSQP0000 DD   DISP=SHR,DSN=myqual.PSID00
//CSQP0001 DD   DISP=SHR,DSN=myqual.PSID01
//CSQP0002 DD   DISP=SHR,DSN=myqual.PSID02
//CSQP0003 DD   DISP=SHR,DSN=myqual.PSID03
//CSQINP1  DD   DISP=SHR,DSN=myqual.CSQINP(CSQ1INP1)
//CSQINP2  DD   DISP=SHR,DSN=myqual.CSQINP(CSQ1&INP2.)
//CSQOUT1  DD   SYSOUT=*
//CSQOUT2  DD   SYSOUT=*
```

Figure 4. Sample start-up procedure

If you then start the your queue manager with the command:

```
    +CSQ1 START QMGR
```

the CSQINP2 actually used is a member called CSQ1NORM.

However, suppose you are putting a new suite of programs into production so that the next time you start queue manager CSQ1, the CSQINP2 definitions are to be taken from member CSQ1NEW. To do this, you would start the queue manager with this command:

```
   +CSQ1 START QMGR ENVPARM('INP2=NEW')
```

and CSQ1NEW would be used instead of CSQ1NORM. Note that z/OS limits the `KEYWORD=value` specifications for symbolic parameters (as in `INP2=NEW`) to 255 characters.

Starting after an abnormal termination

WebSphere MQ automatically detects whether restart follows a normal shutdown or an abnormal termination.

Starting a queue manager after it ends abnormally is different from starting it after the STOP QMGR command has been issued. After STOP QMGR, the system finishes its work in an orderly way and takes a termination checkpoint before stopping. When you restart the queue manager, it uses information from the system checkpoint and recovery log to determine the system status at shutdown.

However, if the queue manager ends abnormally, it terminates without being able to finish its work or take a termination checkpoint. When you restart a queue manager after an abend, it refreshes its knowledge of its status at termination using information in the log, and notifies you of the status of various tasks. Normally, the restart process resolves all inconsistent states. But, in some cases, you must take specific steps to resolve inconsistencies.

User messages on start-up

When you start a queue manager successfully, it produces a set of start up messages similar to the ones in Chapter 7, "User messages on startup," on page 225.

Stopping a queue manager

Before stopping a queue manager, all WebSphere MQ-related write-to-operator-with-reply (WTOR) messages must receive replies, for example, getting log requests. Each command in Figure 5 terminates a running queue manager.

```
+CSQ1 STOP QMGR

+CSQ1 STOP QMGR MODE(QUIESCE)

+CSQ1 STOP QMGR MODE(FORCE)

+CSQ1 STOP QMGR MODE(RESTART)
```

Figure 5. Stopping a queue manager

The command STOP QMGR defaults to STOP QMGR MODE(QUIESCE).

In QUIESCE mode, WebSphere MQ does not allow any new connection threads to be created, but allows existing threads to continue; it terminates only when all threads have ended. Applications can request to be notified in the event of the queue manager quiescing. Therefore, use the QUIESCE mode where possible so that applications that have requested notification have the opportunity to disconnect. See the WebSphere MQ Application Programming Guide for details.

If the queue manager does not terminate in a reasonable time in response to a STOP QMGR MODE(QUIESCE) command, use the DISPLAY CONN command to determine whether any connection threads exist, and take the necessary steps to terminate the associated applications. If there are no threads, issue a STOP QMGR MODE(FORCE) command.

The STOP QMGR MODE(QUIESCE) and STOP QMGR MODE(FORCE) commands deregister WebSphere MQ from the MVS™ Automatic Restart Manager (ARM), preventing ARM from restarting the queue manager automatically. The STOP QMGR MODE(RESTART) command works in the same way as the STOP QMGR MODE(FORCE) command, except that it does not deregister WebSphere MQ from ARM. This means that the queue manager is eligible for immediate automatic restart.

If the WebSphere MQ subsystem is not registered with ARM, the STOP QMGR MODE(RESTART) command is rejected and the following message is sent to the z/OS console:

```
CSQY205I  ARM element arm-element is not registered
```

If this message is not issued, the queue manager is restarted automatically. For more information about ARM, see "Using the z/OS Automatic Restart Manager (ARM)" on page 118.

Only cancel the queue manager address space if STOP QMGR MODE(FORCE) does not terminate the queue manager.

If a queue manager is stopped by either canceling the address space or by using the command STOP QMGR MODE(FORCE), consistency is maintained with connected CICS or IMS systems. Resynchronization of resources is started when a queue manager restarts and is completed when the connection to the CICS or IMS system is established.

Note: When you stop your queue manager, you might find message IEF352I is issued. z/OS issues this message if it detects that failing to mark the address space as unusable would lead to an integrity exposure. You can ignore this message.

Stop messages

After issuing a STOP QMGR command, you get the messages CSQY009I and CSQY002I, for example:

```
CSQY009I +CSQ1 ' STOP QMGR' COMMAND ACCEPTED FROM
USER(userid), STOP MODE(FORCE)
CSQY002I +CSQ1 QUEUE MANAGER STOPPING
```

Where userid is the user ID that issued the STOP QMGR command, and the MODE parameter depends on that specified in the command.

When the STOP command has completed successfully, the following messages are displayed on the z/OS console:

```
CSQ9022I +CSQ1 CSQYASCP ' STOP QMGR' NORMAL COMPLETION
CSQ3104I +CSQ1 CSQ3EC0X - TERMINATION COMPLETE
```

If you are using ARM, and you did not specify MODE(RESTART), the following message is also displayed:

```
CSQY204I +CSQ1 ARM DEREGISTER for element arm-element type
arm-element-type successful
```

You cannot restart the queue manager until the following message has been displayed:

```
CSQ3100I +CSQ1 CSQ3EC0X - SUBSYSTEM ssnm READY FOR START COMMAND
```

Writing programs to administer WebSphere MQ

Start of General-use programming interface information

This chapter contains hints and guidance to enable you to issue WebSphere MQ commands from a WebSphere MQ application program.

It contains these sections:
- "Understanding how it all works"
- "Preparing queues for administration programs" on page 21
- "Using the command server" on page 23
- "Retrieving replies to your commands" on page 26
- "Interpreting the replies" on page 27
- "Using the DISPLAY commands" on page 28
- "Examples of commands and their replies" on page 31
- "If you do not receive a reply" on page 36
- "Passing commands using MGCRE" on page 36

Note: In this chapter, the MQI calls are described using C-language notation. For typical invocations of the calls in the COBOL, PL/I, and assembler languages, see the WebSphere MQ Application Programming Reference manual.

Understanding how it all works

In outline, the procedure for issuing commands from an application program is quite simple:

1. You build a WebSphere MQ command into a type of WebSphere MQ message called a *request message*. The command can be in MQSC or PCF format.
2. You put (**MQPUT**) this message onto a special queue called the system-command input queue. The WebSphere MQ command processor runs the command.

3. You retrieve (**MQGET**) the results of the command as *reply messages* on the reply-to queue. These messages contain the user messages that you need to determine whether your command was successful and, if it was, what the results were.

Then it is up to your application program to process the results.

Before you begin

Before you can write an application program to issue WebSphere MQ commands, you must be familiar with:

1. Issuing WebSphere MQ commands and the command syntax. See WebSphere MQ Script (MQSC) Command Reference or WebSphere MQ Programmable Command Formats and Administration Interface for more information.
2. Writing application programs that use the MQI.

   This includes:
   - Connecting to a queue manager using the **MQCONN** or **MQCONNX** call.
   - Opening a queue using **MQOPEN**.
   - Opening a dynamic queue using **MQOPEN** and specifying the name of a model queue.
   - Putting messages on a queue using **MQPUT** and **MQPUT1**.
   - Getting messages from a queue using **MQGET**.

   You need to know about messages, including:
   - The message descriptor structure.
   - What the persistence attribute of a message means.
   - The types of WebSphere MQ messages, in particular, *request messages* and the *reply messages* they generate.

   You can find all this information in the WebSphere MQ Application Programming Guide and the WebSphere MQ Application Programming Reference manual.
3. User messages.

   These messages are generated by WebSphere MQ to show the success or failure of, and the responses to, WebSphere MQ commands. Each message is identified by an ID that starts with the characters CSQ, for example, CSQN205I. For more information, see the WebSphere MQ for z/OS Messages and Codes manual.

If you want your WebSphere MQ commands to be run on a remote queue manager, see the WebSphere MQ Intercommunication manual for details of how to do this.

WebSphere MQ can also be set up to perform security checks, for example, to ensure that a user is authorized to issue a particular command for a particular resource. For more information, see the WebSphere MQ for z/OS System Setup Guide.

Preparing queues for administration programs

This section applies to commands in the MQSC format. For the equivalent in PCF, see WebSphere MQ Programmable Command Formats and Administration Interface.

Before you can issue any **MQPUT** or **MQGET** calls, you must first define, and then open, the queues you are going to use.

Defining the system-command input queue

The system-command input queue is a local queue called SYSTEM.COMMAND.INPUT. The supplied CSQINP2 initialization data set, thlqual.SCSQPROC(CSQ4INSG), contains a default definition for the system-command input queue. For compatibility with WebSphere MQ on other platforms, an alias of this queue, called SYSTEM.ADMIN.COMMAND.QUEUE is also supplied. See WebSphere MQ for z/OS System Setup Guide for more information.

Defining a reply-to queue

You must define a reply-to queue to receive reply messages from the WebSphere MQ command processor. It can be any queue whose attributes allow reply messages to be put on it. However, for normal operation, specify these attributes:

- USAGE(NORMAL)
- NOTRIGGER (unless your application uses triggering)

You should not normally use persistent messages for commands, but if you choose to do so, the reply-to queue must not be a temporary dynamic queue.

The supplied CSQINP2 initialization data set, thlqual.SCSQPROC(CSQ4INSG), contains a definition for a model queue called SYSTEM.COMMAND.REPLY.MODEL. You can use this model to create a dynamic reply-to queue.

Note: Replies generated by the command processor can be up to 15 000 bytes in length.

If you use a permanent dynamic queue as a reply-to queue, your application should allow time for all PUT and GET operations to complete before attempting to delete the queue, otherwise MQRC2055 (MQRC_Q_NOT_EMPTY) can be returned. If this occurs, retry the queue deletion after a few seconds.

Opening the system-command input queue

Before you can open the system-command input queue, your application program must be connected to your queue manager. Use the MQI call **MQCONN** or **MQCONNX** to do this.

Then use the MQI call **MQOPEN** to open the system-command input queue. To use this call:

1. Set the *Options* parameter to MQOO_OUTPUT
2. Set the MQOD object descriptor fields as follows:

ObjectType
      MQOT_Q (the object is a queue)

ObjectName
      SYSTEM.COMMAND.INPUT

ObjectQMgrName
>> If you want to send your request messages to your local queue
manager, leave this field blank. This means that your commands are
processed locally.
>>
>> If you want your WebSphere MQ commands to be processed on a
remote queue manager, put its name here. You must also have the
correct queues and links set up, as described in the WebSphere MQ
Intercommunication manual.

Opening a reply-to queue

To retrieve the replies from a WebSphere MQ command, you must open a reply-to
queue. One way of doing this is to specify the model queue,
SYSTEM.COMMAND.REPLY.MODEL in an **MQOPEN** call, to create a permanent
dynamic queue as your reply-to queue. To use this call:

1. Set the *Options* parameter to MQOO_INPUT_SHARED
2. Set the MQOD object descriptor fields as follows:

ObjectType
>> MQOT_Q (the object is a queue)

ObjectName
>> The name of your reply-to queue. If the queue name you specify is the
name of a model queue object, the queue manager creates a dynamic
queue.

ObjectQMgrName
>> To receive replies on your local queue manager, leave this field blank.

DynamicQName
>> Specify the name of the dynamic queue to be created.

Using the command server

The command server is a WebSphere MQ component that works with the
command processor component. The command server reads request messages from
the system-command input queue, verifies them, and passes the valid ones as
commands to the command processor. The command processor processes the
commands and puts any replies as reply messages on to the reply-to queue that
you specify. The first reply message contains the user message CSQN205I. See
"Interpreting the replies" on page 27 for more information. The command server
also processes channel initiator and queue-sharing group commands, wherever
they are issued from.

Identifying the queue manager that processes your commands

The queue manager that processes the commands you issue from an
administration program is the queue manager that owns the system-command
input queue that the message is put onto.

Starting the command server

Normally, the command server is started automatically when the queue manager is
started. It becomes available as soon as the message CSQ9022I 'START QMGR'
NORMAL COMPLETION is returned from the START QMGR command. The
command server is stopped when all the connected tasks have been disconnected
during the system termination phase.

You can control the command server yourself using the START CMDSERV and STOP CMDSERV commands. To prevent the command server starting automatically when WebSphere MQ is restarted, you can add a STOP CMDSERV command to your CSQINP1 or CSQINP2 initialization data sets. However, this is not recommended as it will prevent any channel initiator or queue-sharing group commands being processed.

The STOP CMDSERV command stops the command server as soon as it has finished processing the current message, or immediately if no messages are being processed.

If the command server has been stopped by a STOP CMDSERV command in the program, no other commands from the program can be processed. To restart the command server, you must issue a START CMDSERV command from the z/OS console.

If you stop and restart the command server while the queue manager is running, all the messages that are on the system-command input queue when the command server stops are processed when the command server is restarted. However, if you stop and restart the queue manager after the command server is stopped, only the persistent messages on the system-command input queue are processed when the command server is restarted. All nonpersistent messages on the system-command input queue are lost.

Sending commands to the command server

For each command, you build a message containing the command, then put it onto the system-command input queue.

Building a message that includes WebSphere MQ commands:

You can incorporate WebSphere MQ commands in an application program by building request messages that include the required commands. For each such command you:

1. Create a buffer containing a character string representing the command.
2. Issue an **MQPUT** call specifying the buffer name in the *buffer* parameter of the call.

The simplest way to do this in C is to define a buffer using 'char'. For example:

```
char message_buffer[ ] = "ALTER QLOCAL(SALES) PUT(ENABLED)";
```

When you build a command, use a null-terminated character string. Do not specify a command prefix string (CPF) at the start of a command defined in this way. This means that you do not have to alter your command scripts if you want to run them on another queue manager. However, you must take into account that a CPF is included in any response messages that are put onto the reply-to queue.

The command server folds all lowercase characters to uppercase unless they are inside single quotes.

Commands can be any length up to a maximum 32 762 characters.

Putting messages on the system-command input queue

Use the **MQPUT** call to put request messages containing commands on the system-command input queue. In this call you specify the name of the reply-to queue that you have already opened.

To use the **MQPUT** call:

1. Set these **MQPUT** parameters:

   Hconn   The connection handle returned by the **MQCONN** or **MQCONNX** call.

   Hobj   The object handle returned by the **MQOPEN** call for the system-command input queue.

   BufferLength
            The length of the formatted command.

   Buffer The name of the buffer containing the command.

2. Set these MQMD fields:

   MsgType
            MQMT_REQUEST

   Format MQFMT_STRING or MQFMT_NONE

            If you are not using the same code page as the queue manager, set *CodedCharSetId* as appropriate and set MQFMT_STRING, so that the command server can convert the message. Do not set MQFMT_ADMIN, as that causes your command to be interpreted as PCF.

   ReplyToQ
            Name of your reply-to queue.

   ReplyToQMgr
            If you want replies sent to your local queue manager, leave this field blank. If you want your WebSphere MQ commands to be sent to a remote queue manager, put its name here. You must also have the correct queues and links set up, as described in the WebSphere MQ Intercommunication manual.

3. Set any other MQMD fields, as required. You should normally use nonpersistent messages for commands.

4. Set any *PutMsgOpts* options, as required.

   If you specify MQPMO_SYNCPOINT (the default), you must follow the **MQPUT** call with a syncpoint call.

Using MQPUT1 and the system-command input queue:

If you want to put just one message on the system-command input queue, you can use the **MQPUT1** call. This call combines the functions of an **MQOPEN**, followed by an **MQPUT** of one message, followed by an **MQCLOSE**, all in one call. If you use this call, modify the parameters accordingly. See the WebSphere MQ Application Programming Guide for details.

Retrieving replies to your commands

When the command processor processes your commands, any reply messages are put onto the reply-to queue specified in the **MQPUT** call. The command server sends the reply messages with the same persistence as the command message it received.

Waiting for a reply

Use the **MQGET** call to retrieve a reply from your request message. One request message can produce several reply messages. For details, see "Interpreting the replies" on page 27.

You can specify a time interval that an **MQGET** call waits for a reply message to be generated. If you do not get a reply, use the checklist beginning in topic "If you do not receive a reply" on page 36.

To use the **MQGET** call:

1. Set these parameters:

   Hconn   The connection handle returned by the **MQCONN** or **MQCONNX** call.

   Hobj   The object handle returned by the **MQOPEN** call for the reply-to queue.

   Buffer   The name of the area to receive the reply.

   BufferLength
   > The length of the buffer to receive the reply. This must be a minimum of 80 bytes.

2. To ensure that you only get the responses from the command that you issued, you must specify the appropriate *MsgId* and *CorrelId* fields. These depend on the report options, MQMD_REPORT, you specified in the **MQPUT** call:

   MQRO_NONE
   > Binary zero, '00...00' (24 nulls).

   MQRO_NEW_MSG_ID
   > Binary zero, '00...00' (24 nulls).
   >
   > This is the default if none of these options has been specified.

   MQRO_PASS_MSG_ID
   > The *MsgId* from the **MQPUT**.

   MQRO_NONE
   > The *MsgId* from the **MQPUT** call.

   MQRO_COPY_MSG_ID_TO_CORREL_ID
   > The *MsgId* from the **MQPUT** call.
   >
   > This is the default if none of these options has been specified.

   MQRO_PASS_CORREL_ID
   > The *CorrelId* from the **MQPUT** call.

   For more details on report options, see the WebSphere MQ Application Programming Reference manual.

3. Set the following *GetMsgOpts* fields:

   Options
   > MQGMO_WAIT

If you are not using the same code page as the queue manager, set MQGMO_CONVERT, and set *CodedCharSetId* as appropriate in the MQMD.

WaitInterval
For replies from the local queue manager, try 5 seconds. Coded in milliseconds, this becomes 5 000. For replies from a remote queue manager, and channel control and status commands, try 30 seconds. Coded in milliseconds, this becomes 30 000.

Discarded messages:

If the command server finds that a request message is not valid, it discards this message and writes the message CSQN205I to the named reply-to queue. If there is no reply-to queue, the CSQN205I message is put onto the dead-letter queue. The return code in this message shows why the original request message was not valid:

00D5020F	It is not of type MQMT_REQUEST.
00D50210	It has zero length.
00D50212	It is longer than 32 762 bytes.
00D50211	It contains all blanks.
00D5483E	It needed converting, but *Format* was not MQFMT_STRING.
Other	See the WebSphere MQ for z/OS Messages and Codes manual.

The reply message descriptor

For any reply message, the following MQMD message descriptor fields are set:

MsgType	MQMT_REPLY
Feedback	MQFB_NONE
Encoding	MQENC_NATIVE
Priority	As for the MQMD in the message you issued.
Persistence	As for the MQMD in the message you issued.
CorrelId	Depends on the **MQPUT** report options.
ReplyToQ	None.

The command server sets the *Options* field of the MQPMO structure to MQPMO_NO_SYNCPOINT. This means that you can retrieve the replies as they are created, rather than as a group at the next syncpoint.

End of General-use programming interface information

Interpreting the replies

Start of Product-sensitive programming interface information

Each request message correctly processed by WebSphere MQ produces at least two reply messages. Each reply message contains a single WebSphere MQ user message.

The length of a reply depends on the command that was issued. The longest reply you can get is from a DISPLAY NAMELIST, and that can be up to 13 000 bytes long.

The first user message, CSQN205I, always contains:
- A count of the replies (in decimal), which you can use as a counter in a loop to get the rest of the replies. The count includes this first message.
- The return code from the command preprocessor.
- A reason code, which is the return code from the command processor.

This message does not contain a CPF.

For example:

```
CSQN205I   COUNT=       4, RETURN=0000000C, REASON=00000008
```

The COUNT field is 8 bytes long and is right-justified. It always starts at position 18, that is, immediately after 'COUNT='. The RETURN field is 8 bytes long in character hexadecimal and is immediately after 'RETURN=' at position 35. The REASON field is 8 bytes long in character hexadecimal and is immediately after 'REASON=' at position 52.

If the RETURN= value is 00000000 and the REASON= value is 00000004, the set of reply messages is incomplete. After retrieving the replies indicated by the CSQN205I message, issue a further **MQGET** call to wait for a further set of replies. The first message in the next set of replies is again CSQN205I, indicating how many replies there are, and whether there are still more to come.

See the WebSphere MQ for z/OS Messages and Codes manual for more details about the individual messages.

If you are using a non-English language feature, the text and layout of the replies are different from those shown here. However, the size and position of the count and return codes in message CSQN205I are the same.

Using the DISPLAY commands

To obtain information about WebSphere MQ, use the WebSphere MQ MQSC DISPLAY commands. Use these commands rather than **MQINQ** if you want:
- Information about objects on a remote queue manager. (**MQINQ** only returns information from the local queue manager.)
- Reply messages ready-formatted for printing. (**MQINQ** returns information that is not formatted.)
- Other information that **MQINQ** does not provide.

The format of the replies from these commands:
- DISPLAY ARCHIVE
- DISPLAY CMDSERV
- DISPLAY CHINIT
- DISPLAY GROUP
- DISPLAY LOG
- DISPLAY MAXSMSGS

- DISPLAY SECURITY
- DISPLAY SYSTEM
- DISPLAY THREAD
- DISPLAY TRACE
- DISPLAY USAGE

is the same, regardless of whether you issue the command from an application program or from a z/OS console, with one reply for each line you see on the z/OS console. However, if you issue DISPLAY commands listed below for WebSphere MQ objects or object status, the format is different when they are issued from an application program:

- DISPLAY AUTHINFO
- DISPLAY CFSTATUS
- DISPLAY CFSTRUCT
- DISPLAY CHANNEL
- DISPLAY CHSTATUS
- DISPLAY CLUSQMGR
- DISPLAY CONN
- DISPLAY NAMELIST
- DISPLAY PROCESS
- DISPLAY QMGR
- DISPLAY QUEUE
- DISPLAY QSTATUS
- DISPLAY STGCLASS

Whereas on the z/OS console you get one line for each attribute and value, for an application program there is only one message for each object.

The format of these reply messages is:

```
msg_no +CSQ1 attr_name(value) attr_name attr_name(value)
```

where:
- `msg_no` is an 8 character message number
- `+CSQ1` is the command prefix string
- `attr_name` is the attribute or keyword name
- `value` is the attribute value

The format of the attributes and their values follow these rules:
1. Not all attributes have associated values.
2. Each attribute or attribute and value pair is separated by one or more blanks.
3. Attributes are not always returned in the same order.
4. The attribute values returned are fixed length and surrounded by parentheses.

   Integer values are ten characters long, right justified, and padded with blanks.

   Character values are left justified and padded with blanks. Their lengths are as follows:

   a. Character string lengths are the same as those given in the WebSphere MQ Application Programming Reference manual.

b. Attributes that return a keyword, for example, DEFSOPT returns EXCL or SHARED, are 10 characters long, left justified, and padded with blanks, with the exception of the TYPE attribute for queues, which is 8 characters long.

c. Some attribute keywords can take negated values, for example, NOTRIGGER, NOSHARE, and NOHARDENBO. The attribute keywords that can have negated values take their length from the negated value. For example, the negated equivalent of SHARE is NOSHARE; it has a length of 7. These attributes are left justified and padded with blanks.

5. The number of attributes returned depends on what attributes are requested by the command.

6. Some attributes return a list of values, each fixed length, separated by commas, as follows:

a. The NAMES attribute returns a list of names. Use the NAMCOUNT attribute to discover the number of names in the list. If there are no names in the list, the NAMES attribute is returned as NAMES().

b. The MSGEXIT, SENDEXIT and RCVEXIT attributes of a channel return multiple values in a list, each of 128 characters. There can be any number of entries in the list from zero to 8.

c. The MSGDATA, SENDDATA and RCVDATA attributes of a channel return multiple values in a list, each 32 characters long.

d. The COMPHDR and COMPMSG channel attributes return a list of keywords, each 10 characters long.

e. The CONNOPTS and OPENOPTS keywords return a list of option values, each 30 characters long.

f. Many of the monitoring attributes return a pair of integers, each 10 characters long.

7. Attributes that normally require quotes around the string because they contain embedded blanks, lowercase characters, or special characters, are returned without the quotes.

8. When you want to use the reply to a DISPLAY command as input to another command, put single quotes (' ') around each attribute. For example, if you define this queue:

```
+CSQ1 DEFINE QLOCAL(SALES) DESCR('Sales enquiries queue')
```

You can display it using the command:

```
+CSQ1 DISPLAY QUEUE(SALES) DESCR
```

The DESCR attribute is displayed as:

```
DESCR(Sales enquiries queue)
```

To use this description in another command you must add the quotes as follows:

```
   DESCR('Sales enquiries queue')
```

If the attribute itself contains any quotes, you must double them.

Examples of commands and their replies

Here are some examples of commands that could be built into WebSphere MQ messages, and the user messages that are the replies. Unless otherwise stated, each line of the reply is a separate message.

End of product-sensitive programming interface information

Messages from a DEFINE command

The following command:

```
 DEFINE QLOCAL(Q1)
```

produces these messages:

```
CSQN205I   COUNT=       2, RETURN=00000000, REASON=00000000
CSQ9022I +CSQ1 CSQMMSGP ' DEFINE QLOCAL' NORMAL COMPLETION
```

These reply messages are produced on normal completion.

Messages from a DELETE command

The following command:

```
 DELETE QLOCAL(Q2)
```

produces these messages:

```
CSQN205I   COUNT=       4, RETURN=0000000C, REASON=00000008
CSQM125I +CSQ1 CSQMUQLC QLOCAL (Q2) QSGDISP(QMGR) WAS NOT FOUND
CSQM090E +CSQ1 CSQMUQLC FAILURE REASON CODE X'00D44002'
CSQ9023E +CSQ1 CSQMUQLC ' DELETE QLOCAL' ABNORMAL COMPLETION
```

These messages indicate that a local queue called Q2 does not exist.

Messages from DISPLAY commands

The following examples show the replies from some DISPLAY commands.

Finding out the name of the dead-letter queue:

If you want to find out the name of the dead-letter queue for a queue manager, issue this command from an application program:

```
DISPLAY QMGR DEADQ
```

The following three user messages are returned, from which you can extract the required name:

```
CSQN205I   COUNT=       3, RETURN=00000000, REASON=00000000
CSQM409I +CSQ1 QMNAME(CSQ1) DEADQ(SYSTEM.DEAD.QUEUE                )
CSQ9022I +CSQ1 CSQMDRTS ' DISPLAY QMGR' NORMAL COMPLETION
```

Messages from the DISPLAY QUEUE command:

The following examples show how the results from a command depend on the attributes specified in that command.

Example 1:

You define a local queue using the command:

```
DEFINE QLOCAL(Q1) DESCR('A sample queue') GET(ENABLED) SHARE
```

If you issue the following command from an application program:

```
DISPLAY QUEUE(Q1) SHARE GET DESCR
```

these three user messages are returned:

```
CSQN205I   COUNT=       3, RETURN=00000000, REASON=00000000
CSQM401I +CSQ1 QUEUE(Q1                                          ) TYPE(
QLOCAL  ) QSGDISP(QMGR       )
         DESCR(A sample queue
       ) SHARE   GET(ENABLED   )
CSQ9022I +CSQ1 CSQMDMSG ' DISPLAY QUEUE' NORMAL COMPLETION
```

Note: The second message, CSQM401I, is shown here occupying four lines.

Example 2:

Two queues have names beginning with the letter A:
- A1 is a local queue with its PUT attribute set to DISABLED.
- A2 is a remote queue with its PUT attribute set to ENABLED.

If you issue the following command from an application program:

```
DISPLAY QUEUE(A*) PUT
```

these four user messages are returned:

```
CSQN205I   COUNT=      4, RETURN=00000000, REASON=00000000
CSQM401I +CSQ1 QUEUE(A1                                             ) TYPE(
QLOCAL  ) QSGDISP(QMGR     )
         PUT(DISABLED  )
CSQM406I +CSQ1 QUEUE(A2                                             ) TYPE(
QREMOTE ) PUT(ENABLED   )
CSQ9022I +CSQ1 CSQMDMSG ' DISPLAY QUEUE' NORMAL COMPLETION
```

Note: The second and third messages, CSQM401I and CSQM406I, are shown here occupying three and two lines respectively.

Messages from the DISPLAY NAMELIST command:

You define a namelist using the command:

```
DEFINE NAMELIST(N1) NAMES(Q1,SAMPLE_QUEUE)
```

If you issue the following command from an application program:

```
DISPLAY NAMELIST(N1) NAMES NAMCOUNT
```

the following three user messages are returned:

```
CSQN205I   COUNT=      3, RETURN=00000000, REASON=00000000
CSQM407I +CSQ1 NAMELIST(N1                                         ) QS
GDISP(QMGR     ) NAMCOUNT(       2) NAMES(Q1
 ,SAMPLE_QUEUE                               )
CSQ9022I +CSQ1 CSQMDMSG ' DISPLAY NAMELIST' NORMAL COMPLETION
```

Note: The second message, CSQM407I, is shown here occupying three lines.

Messages from commands with CMDSCOPE

The following examples show the replies from commands that have been entered with the CMDSCOPE attribute.

Messages from the ALTER PROCESS command:

The following command:

```
ALT PRO(V4) CMDSCOPE(*)
```

produces the following messages:

```
CSQN205I   COUNT=       2, RETURN=00000000, REASON=00000004
CSQN137I !MQ25 'ALT PRO' command accepted for CMDSCOPE(*), sent to 2
CSQN205I   COUNT=       5, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'ALT PRO' command responses from MQ26
CSQM125I !MQ26 CSQMMSGP PROCESS(V4) QSGDISP(QMGR) WAS NOT FOUND
CSQM090E !MQ26 CSQMMSGP FAILURE REASON CODE X'00D44002'
CSQ9023E !MQ26 CSQMMSGP ' ALT PRO' ABNORMAL COMPLETION
CSQN205I   COUNT=       3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'ALT PRO' command responses from MQ25
CSQ9022I !MQ25 CSQMMSGP ' ALT PRO' NORMAL COMPLETION
CSQN205I   COUNT=       2, RETURN=0000000C, REASON=00000008
CSQN123E !MQ25 'ALT PRO' command for CMDSCOPE(*) abnormal completion
```

These messages tell you that the command was entered on queue manager MQ25 and sent to two queue managers (MQ25 and MQ26). The command was successful on MQ25 but the process definition did not exist on MQ26, so the command failed on that queue manager.

Messages from the DISPLAY PROCESS command:

The following command:

```
DIS PRO(V*) CMDSCOPE(*)
```

produces the following messages:

```
CSQN205I   COUNT=       2, RETURN=00000000, REASON=00000004
CSQN137I !MQ25 'DIS PRO' command accepted for CMDSCOPE(*), sent to 2
CSQN205I   COUNT=       5, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DIS PRO' command responses from MQ26
CSQM408I !MQ26  PROCESS(V2)   QSGDISP(COPY)
CSQM408I !MQ26  PROCESS(V3)   QSGDISP(QMGR)
CSQ9022I !MQ26 CSQMDRTS ' DIS PROCESS' NORMAL COMPLETION
CSQN205I   COUNT=       7, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DIS PRO' command responses from MQ25
CSQM408I !MQ25  PROCESS(V2)   QSGDISP(COPY)
CSQM408I !MQ25  PROCESS(V2)   QSGDISP(GROUP)
CSQM408I !MQ25  PROCESS(V3)   QSGDISP(QMGR)
CSQM408I !MQ25  PROCESS(V4)   QSGDISP(QMGR)
CSQ9022I !MQ25 CSQMDRTS ' DIS PROCESS' NORMAL COMPLETION
CSQN205I   COUNT=       2, RETURN=00000000, REASON=00000000
CSQN122I !MQ25 'DIS PRO' command for CMDSCOPE(*) normal completion
```

These mesasges tell you that the command was entered on queue manager MQ25 and sent to two queue managers (MQ25 and MQ26). Information is displayed about all the processes on each queue manager with names starting with the letter V.

Messages from the DISPLAY CHSTATUS command:

The following command:

```
DIS  CHS(VT) CMDSCOPE(*)
```

produces the following messages:

```
CSQN205I   COUNT=      2, RETURN=00000000, REASON=00000004
CSQN137I !MQ25 'DIS CHS' command accepted for CMDSCOPE(*), sent to 2
CSQN205I   COUNT=      4, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DIS CHS' command responses from MQ25
CSQM422I !MQ25 CHSTATUS(VT) CHLDISP(PRIVATE) CONNAME( ) CURRENT STATUS(STOPPED)
CSQ9022I !MQ25 CSQXDRTS ' DIS CHS' NORMAL COMPLETION
CSQN205I   COUNT=      4, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DIS CHS' command responses from MQ26
CSQM422I !MQ26 CHSTATUS(VT) CHLDISP(PRIVATE) CONNAME( ) CURRENT STATUS(STOPPED)
CSQ9022I !MQ26 CSQXDRTS ' DIS CHS' NORMAL COMPLETION
CSQN205I   COUNT=      2, RETURN=00000000, REASON=00000000
CSQN122I !MQ25 'DIS CHS' command for CMDSCOPE(*) normal completion
```

These messages tell you that the command was entered on queue manager MQ25 and sent to two queue managers (MQ25 and MQ26). Information is displayed about channel status on each queue manager.

Messages from the STOP CHANNEL command:

The following command:

```
STOP CHL(VT) CMDSCOPE(*)
```

produces these messages:

```
CSQN205I   COUNT=      2, RETURN=00000000, REASON=00000004
CSQN137I !MQ25 'STOP CHL' command accepted for CMDSCOPE(*), sent to 2
CSQN205I   COUNT=      3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'STOP CHL' command responses from MQ25
CSQM134I !MQ25 CSQMTCHL  STOP CHL(VT) COMMAND ACCEPTED
SQN205I    COUNT=      3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'STOP CHL' command responses from MQ26
CSQM134I !MQ26 CSQMTCHL  STOP CHL(VT) COMMAND ACCEPTED
CSQN205I   COUNT=      3, RETURN=00000000, REASON=00000000
CSQN121I !MQ25 'STOP CHL' command responses from MQ26
CSQ9022I !MQ26 CSQXCRPS ' STOP CHL' NORMAL COMPLETION
CSQN205I   COUNT=      3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'STOP CHL' command responses from MQ25
CSQ9022I !MQ25 CSQXCRPS ' STOP CHL' NORMAL COMPLETION
CSQN205I   COUNT=      2, RETURN=00000000, REASON=00000000
CSQN122I !MQ25 'STOP CHL' command for CMDSCOPE(*) normal completion
```

These messages tell you that the command was entered on queue manager MQ25 and sent to two queue managers (MQ25 and MQ26). Channel VT was stopped on each queue manager.

Messages from commands that generate commands with CMDSCOPE

The following command:

```
DEF PRO(V2) QSGDISP(GROUP)
```

produces these messages:

```
CSQN205I   COUNT=      3, RETURN=00000000, REASON=00000004
CSQM122I !MQ25 CSQMMSGP ' DEF PRO' COMPLETED FOR QSGDISP(GROUP)
CSQN138I !MQ25 'DEFINE PRO' command generated for CMDSCOPE(*), sent to 2
CSQN205I   COUNT=      3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DEFINE PRO' command responses from MQ25
CSQ9022I !MQ25 CSQMMSGP ' DEFINE PROCESS' NORMAL COMPLETION
CSQN205I   COUNT=      3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DEFINE PRO' command responses from MQ26
CSQ9022I !MQ26 CSQMMSGP ' DEFINE PROCESS' NORMAL COMPLETION
CSQN205I   COUNT=      2, RETURN=00000000, REASON=00000000
CSQN122I !MQ25 'DEFINE PRO' command for CMDSCOPE(*) normal completion
```

These messages tell you that the command was entered on queue manager MQ25. When the object was created on the shared repository, another command was generated and sent to all the active queue managers in the queue-sharing group (MQ25 and MQ26).

If you do not receive a reply

If you do not receive a reply to your request message, work through this checklist:
- Is the command server running?
- Is the *WaitInterval* long enough?
- Are the system-command input and reply-to queues correctly defined?
- Were the **MQOPEN** calls to these queues successful?
- Are both the system-command input and reply-to queues enabled for **MQPUT** and **MQGET** calls?
- Have you considered increasing the MAXDEPTH and MAXMSGL attributes of your queues?
- Are you are using the *CorrelId* and *MsgId* fields correctly?
- Is the queue manager still running?
- Was the command built correctly?
- Are all your remote links defined and operating correctly?
- Were the **MQPUT** calls correctly defined?
- Has the reply-to queue been defined as a temporary dynamic queue instead of a permanent dynamic queue? (If the request message is persistent, you must use a permanent dynamic queue for the reply.)

When the command server generates replies but cannot write them to the reply-to queue that you specify, it writes them to the dead-letter queue.

Passing commands using MGCRE

If you have the correct authorization, you can pass WebSphere MQ commands from your program to multiple queue managers by the MGCRE (SVC 34) z/OS service. The value of the CPF identifies the particular queue manager to which the command is directed. For information about CPFs, see the WebSphere MQ for z/OS System Setup Guide.

If you use MGCRE, you can use a Command and Response Token (CART) to get the direct responses to the command.

Chapter 2. WebSphere MQ and CICS

Operating the CICS adapter

Use the CICS adapter control functions to initiate and manage connections between WebSphere MQ and CICS.

This chapter is applicable only for users of CICS TS V3.1 and earlier. Users of CICS TS V3.2 and later should refer to the section *CICS Integration with WebSphere MQ* in the CICS information center. Equivalent information to this chapter is in the topic *Operating the CICS-MQ Adapter.*

This chapter describes these tasks:
- "Invoking the adapter's control functions"
- "Accessing the CICS adapter control panels" on page 41
- "Starting a connection" on page 42
- "Stopping a connection" on page 45
- "Modifying a connection" on page 47
- "Displaying details of connections and CICS tasks" on page 49
- "Starting an instance of the task initiator CKTI" on page 50
- "Stopping an instance of CKTI" on page 52
- "Displaying the current instances of CKTI" on page 54
- "Displaying CICS task information" on page 55
- "Purging tasks that are using the CICS adapter" on page 56
- "Shutting down a connection between WebSphere MQ and the CICS adapter" on page 57

Before you can use the CICS adapter for messaging, you must start the queue manager.

Invoking the adapter's control functions

You can invoke the control functions of the CICS adapter in three different ways:
1. From the CICS adapter control panels.
2. From the CICS command line.
3. From an application program.

From the CICS adapter control panels

You can use the CICS adapter control panels to monitor and control connections between WebSphere MQ and CICS.

From the initial panel, you first select an item from the menu bar at the top of the panel, and then select an action from one of the pull-down menus. In the displayed panel or secondary window, you can then type new values in the fields, as required.

From the CICS command line

You can take a "fast-path" approach and bypass the CICS adapter control panels, by specifying command line parameters on the CKQC transaction. The syntax of these command parameters, and examples of them, are given for each task described later in this chapter.

Note: You can also issue these commands from the console using z/OS commands. These commands take the following form:

```
MODIFY CICS-job-name CKQC command-line-command
```

From CICS application programs

You can use the EXEC CICS LINK command to invoke most adapter control functions from CICS application programs. The syntax of the EXEC CICS LINK commands you need, and examples, are given for each task described later in this chapter.

Command syntax in application programs:

Some commands issued in this way must be padded with trailing spaces to make the length of the command 10 characters. When an argument follows the command, an extra space character must be added as a separator (see Figure 6). The commands affected by this restriction and the number of trailing spaces required for each command are as follows:

Command	Number of trailing spaces (not including the separator)
START	5
MODIFY	4
STARTCKTI	1
STOPCKTI	2

With all other commands the padding is optional.

```
EXEC CICS LINK PROGRAM('CSQCRST ')
          INPUTMSG('CKQC MODIFY     Y')
                    ↑           ↑
                    ............
                    1          12
```

Figure 6. Padding adapter commands. The MODIFY command must be padded with 4 trailing spaces plus another space as a separator. Starting at the 'M' in MODIFY, the argument 'Y' is the twelfth character.

Note: This restriction applies only to commands issued from an application program; it does not apply to commands issued from the command line.

Passing parameters from a CICS transaction:

Use the following rules to determine how to pass the parameters:
- The CICS transaction must be running on an attached terminal. If it is not, all WebSphere MQ commands are ignored.

- If a CICS application program on an attached terminal is connected to WebSphere MQ, you must use the INPUTMSG option with EXEC CICS LINK to pass parameters, except at PLTPI time.
- If you connect to WebSphere MQ at PLTPI time, you must use the COMMAREA option to pass parameters. If you use the INPUTMSG option, the command is ignored.

    However, the adapter STOP commands:

```
CKQC STOP
CKQC STOP FORCE
```

cannot be run at PLTPI time, regardless of whether you use the INPUTMSG option or the COMMAREA option.

EXEC CICS LINK interface messages:

If you invoke the adapter operation functions START and STOP from an application program using EXEC CICS LINK, the resultant messages are written to both the system console and a transient data queue (TDQ) named CKQQ. When control is returned to the application program from the LINK, the application program can read back the messages by repeating EXEC CICS READQ TD QUEUE(CKQQ) until the queue is empty. The following restrictions apply:

- The TDQ queue name is CKQQ and cannot be changed. A sample TDQ definition is provided (in CSQ4B100), which defines CKQQ as an intra-partition TDQ.
- The queue is not cleared before it is written to.
- The messages are not time-stamped.
- If you have more than one application writing to the TDQ, the messages are not serialized. It is the responsibility of the invoking programs to serialize themselves.
- The same set of messages also appear on the system console.
- The server subtask messages are not written to CKQQ.

Accessing the CICS adapter control panels

To access the adapter control panels, use the CICS transaction CKQC:

1. Type CKQC and press Enter.

    The CICS adapter control initial panel, shown in Figure 7 on page 42, is displayed.
2. In the menu bar at the top of the screen, use the TAB key to move between the three options **Connection**, **CKTI**, and **Task**.
3. Press Enter to select your choice.
4. Select the required option from one of the pull-down menus by typing the number of your choice and then pressing Enter to confirm or function key F12 to cancel.

Press function key F1 to get help on any panel or window.

```
  Connection       CKTI       Task
-----------------------------------------------------------------------
CKQCM0              CICS Adapter Control -- Initial panel

Select menu bar item using Tab key. Then press Enter.




 



IBM WebSphere MQ for z/OS



(C) Copyright IBM Corporation 1993, 2005. All rights reserved.


F1=Help  F3=Exit
```

Figure 7. The CICS adapter control initial panel

> **Note:** You can access the adapter control panels *without* starting the queue manager. You can also start a connection but it will not be active until the queue manager is started.

Starting a connection

You can start a connection from:
- The CICS adapter control panels
- The CICS command line
- A CICS application program
- A PLTPI program
- The CICS MQCONN SIT parameter

Starting a connection from the CICS adapter control panels

To start a connection from the CICS adapter control initial panel:

1. Select **Connection** from the menu bar.
2. Select the **Start** action from the pull-down menu. See Figure 8 on page 43.
3. Modify the connection values displayed in the **Start a Connection** secondary parameter window. Alternatively, use the defaults derived from the INITPARM settings, if defined.
4. Press Enter to confirm.

Messages indicating the success or failure of the attempt to start the connection are displayed on the CICS adapter messages panel, CKQCM1.

```
   Connection        CKTI        Task
+------------------+--------------------------------------------------------
| Select an action. |CS Adapter Control -- Initial panel
|                   |
| 1 1. Start...     |sing Tab key. Then press Enter.
|   2. Stop...      |
|   3. Modify...    |   +----------------------------------------------------+
|   4. Display      |   |                Start a Connection                  |
|                   |   |                                                    |
+------------------+   | Type parameters. Then press Enter.                 |
| F1=Help F12=Cancel |  |                                                    |
+------------------+   | 1. Queue Manager Name (SN) . . . QMGR              |
                       | 2. Initiation Queue Name (IQ) . . . . . . . . . . |
                       | CICS.INITIATION.QUEUE1                              |
                       | 3. Trace Number (TN) . . . . .    123              |
                       |                                                    |
                       +----------------------------------------------------+
IBM WebSphere MQ for z/O| F1=Help  F12=Cancel                               |
                       +----------------------------------------------------+

(C) Copyright IBM Corporation 1993, 2005 All rights reserved.


F1=Help  F3=Exit
```

Figure 8. Starting a connection

Starting a connection from the CICS command line

The example shown in Figure 9 starts a connection, using the default connection values set at system initialization.

```
CKQC START
```

Figure 9. Starting a connection from the command line

The command shown in Figure 10 starts a connection, using the explicitly defined connection parameter values. The parameters are positional; you must enter every field to its maximum length if you want to override the default.

```
CKQC START Y|N <subsystem ID> <trace number> <initiation queue name>
```

Figure 10. Starting a connection from the command line specifying parameters

Where:

Y|N   Specify either:

> **Y**   Use the default values, that is, substitute default values for any blank arguments.

> **N**   Do not use the default values.

<subsystem ID>
> Name of the queue manager to connect to.

<trace number>
> The trace number. It must be in the range zero through 199.

<initiation queue name>
> The name of the default initiation queue.

Specifying lowercase queue names:

By default, CICS folds lowercase input, for both keywords and parameters, to uppercase. Therefore, by default, these commands are equivalent:

```
CKQC START Y CSQ1 199 CICS01.INITQ
ckqc start y csq1 199 cics01.initq
```

Figure 11. Specifying lowercase queue names

If you want to use lowercase queue names, you must:

1. Specify UCTRAN(TRANID) on the TYPETERM definition of terminals that start adapter control functions.
2. Specify UCTRAN(NO) on the transaction profile used by all "CKxx" transactions.

Thereafter, the adapter translates all lowercase arguments, *except queue names*, to uppercase.

For details of TYPETERM and PROFILE definitions, see the *CICS Resource Definition Guide*.

Starting a connection from a CICS application program

You can start a connection by linking the adapter connect program, CSQCQCON, from a CICS application program. Your program, which can be written in C, COBOL, PL/I, or assembler language, must pass a parameter list that specifies the connection values to be used. The parameter list is:

CKQC
> 4-character transaction ID—must be `'CKQC'`.

DISPMODE
> 1-byte field—must contain a blank.

CONNREQ
> 10-character field—must contain `'START     '`.

DELIM1
> 1-byte delimiter field—must contain a blank.

INITP 1-character field that specifies whether this connection is to use the default parameters set by INITPARM. The possible values are:

> **Y** Use the default values, that is, substitute default values for any blank arguments.
>
> **N** Do not use the default values. If you specify 'N', you must supply all the new connection values, to override the INITPARM settings, in the CONNSSN, CONNTN, and CONNIQ fields.
>
> ' ' Equivalent to 'Y'.

DELIM2
> 1-byte delimiter field—must contain a blank.

CONNSSN
> 4-character field used to specify the z/OS subsystem name of the target queue manager.
>
> This must be the name of a queue manager, not a queue-sharing group.

DELIM3

1-byte delimiter field—must contain a blank.

CONNTN

3-character trace number. If supplied, it must be in the range zero through 199.

DELIM4

1-byte delimiter field—must contain a blank.

CONNIQ

48-character field that specifies the name of the default initiation queue.

Figure 12 shows the LINK command that your CICS program must issue.

```
EXEC CICS LINK PROGRAM('CSQCQCON')
          INPUTMSG(CONNPL) INPUTMSGLEN(length of CONNPL)
```

Figure 12. Linking to the adapter connect program, CSQCQCON, from a CICS program. In this example, the name of the parameter list is CONNPL.

Output messages from CSQCQCON are displayed on the system console.

Stopping a connection

You can stop a connection from:
- The CICS adapter control panels
- The CICS command line
- A CICS application program

Stopping a connection from the CICS adapter control panels

From the initial panel:
1. Select **Connection** from the menu bar.
2. Select the **Stop** action from the pull-down menu.
3. Use the **Stop Connection** secondary parameter window to select the type of shutdown that you require. Methods of shutting down the CICS adapter are summarized in Table 2 on page 57.

```
   Connection        CKTI       Task
+-------------------+------------------------------------------------------
| Select an action. |CS Adapter Control -- Initial panel
|                   |
| 2 1. Start...     |sing Tab key. Then press Enter
|   2. Stop...      |
|   3. Modify...    |   +--------------------+
|   4. Display      |   |   Stop Connection  |
|                   |   |                    |
+-------------------+   | Select stop type.  |
| F1=Help F12=Cancel |  | Then press Enter   |
+-------------------+   |                    |
|                   |   | 1 1. Quiesce       |
|                   |   |   2. Force         |
|                   |   |                    |
|                   |   +--------------------+
|                   |   | F1=Help F12=Cancel |
IBM WebSphere MQ for z/O+--------------------+


(C) Copyright IBM Corporation 1993, 2005 All rights reserved.


F1=Help  F3=Exit
```
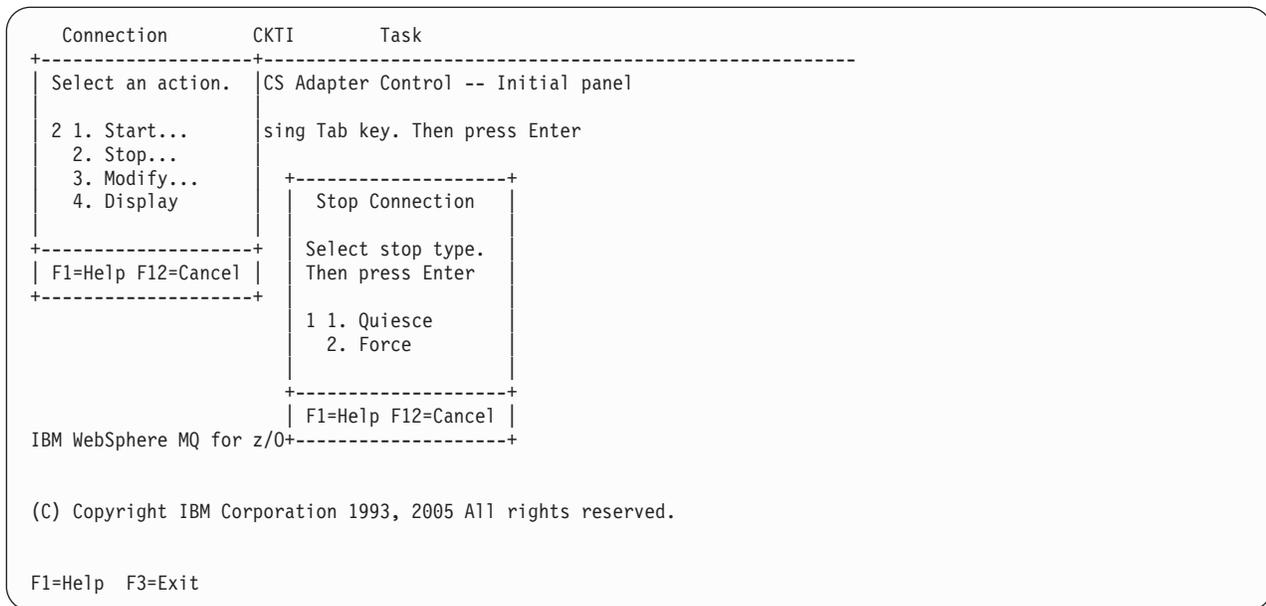
Figure 13. Stopping a connection from the CKQC initial panel

The messages associated with stopping a connection are displayed on the system console.

Stopping a connection from the CICS command line

The command shown in Figure 14 initiates a *quiesced* shutdown. The connection shuts down only after the last task has completed its work.

```
CKQC STOP
```

Figure 14. Stopping a connection from the command line: a quiesced shutdown

The command shown in Figure 15 initiates a *forced* shutdown. The connection shuts down immediately, regardless of the state of any in-flight tasks.

```
CKQC STOP FORCE
```

Figure 15. Stopping a connection from the command line: a forced shutdown

Stopping a connection from a CICS application program

To stop a connection from a CICS program, the program must link to the adapter shutdown program, CSQCDSC. Figures Figure 16 and Figure 17 on page 47 show examples of LINK commands initiating quiesced and forced shutdowns. When you do an EXEC CICS LINK to CSQCDSC, the program requires a terminal associated task.

```
EXEC CICS LINK PROGRAM('CSQCDSC ')
          INPUTMSG('CKQC STOP')
```

Figure 16. Stopping a connection from a CICS application program: a quiesced shutdown. The QUIESCE parameter is optional.

```
EXEC CICS LINK PROGRAM('CSQCDSC ')
          INPUTMSG('CKQC STOP FORCE')
```

Figure 17. Stopping a connection from a CICS application program: a forced shutdown

Output messages from CSQCDSC are displayed on the system console.

Modifying a connection

You can modify a connection to reset the connection statistics, enable or disable the API-crossing exit, or change the adapter's trace number. You can do this from:

- The CICS adapter control panels
- The CICS command line
- A CICS application program

Modifying a connection from the CICS adapter control panels

From the initial panel:

1. Select **Connection** from the menu bar.
2. Select the **Modify** action from the pull-down menu.
3. Use the **Modification Options** secondary parameter window to specify the option you require.

   To change the trace number:

   - Enter 4 in the options selection field
   - Enter a number, in the range zero through 199, in the trace number field. Do not change the 4 in the options selection field before you press Enter. If you do, the trace number is not changed.
4. Press Enter to confirm your choice.
5. Repeat steps 1 through 4, as required.

```
    Connection      CKTI        Task
   +-------------------+------------------------------------------------------
   | Select an action. |CS Adapter Control -- Initial panel
   |                   |
   | 3 1. Start...     |sing Tab key. Then press Enter.
   |   2. Stop...      |
   |   3. Modify...    |   +-------------------------------+
   |   4. Display      |   |        Modification Options   |
   |                   |   |                               |
   +-------------------+   |  Select modify option. Then   |
   | F1=Help F12=Cancel|   |  press Enter.                 |
   +-------------------+   |                               |
                          |  4 1. Reset statistics        |
                          |    2. Enable API Exit         |
                          |    3. Disable API Exit         |
                          |    4. Change Trace Number 123 |
                          +-------------------------------+
   IBM WebSphere MQ for z/O| F1=Help  F12=Cancel           |
                          +-------------------------------+

   (C) Copyright IBM Corporation 1993, 2005 All rights reserved.


   F1=Help  F3=Exit
```

Figure 18. Modifying a connection

Modifying a connection from the CICS command line

You can use the CKQC MODIFY command to modify a connection, as shown in Figure 19:

```
CKQC MODIFY Y|N E|D <trace-number>
```

Figure 19. Format of command to modify connection parameters from the command line

where:

Y|N   Specify one of:

    Y   Reset connection statistics.

    N   Do not reset connection statistics.

    This parameter is required.

E|D   Specify one of:

    E   Enable the API-crossing exit.

    D   Disable the API-crossing exit.

    This parameter is optional, the default is D.

<trace number>
    Specify a valid trace number in the range zero through 199. This parameter is optional. If it is not specified, the trace number is not changed.

The command shown in Figure 20 resets the connection statistics only. The command shown in Figure 21 disables the API-crossing exit and changes the trace number to 121.

```
CKQC MODIFY Y
```

Figure 20. Resetting connection statistics from the command line

```
CKQC MODIFY N D 121
```

Figure 21. Changing the adapter's trace number and disabling the API-crossing exit from the command line

Modifying a connection from a CICS application program

To modify a connection from a CICS program, the program must link to the adapter reset program, CSQCRST.

Figure 22 shows the format of the LINK command. It has the same effect as the command line requests described in "Modifying a connection from the CICS command line."

```
EXEC CICS LINK PROGRAM('CSQCRST ')
          INPUTMSG('CKQC MODIFY     Y E <trace-number>')
```

Figure 22. Format of the MODIFY command issued from a CICS adapter application program

The command shown in Figure 23 resets the connection statistics only.

```
EXEC CICS LINK PROGRAM('CSQCRST ')
          INPUTMSG('CKQC MODIFY    Y')
```

Figure 23. Resetting connection statistics from a CICS program

The command shown in Figure 24 disables the API-crossing exit and changes the trace number to 121.

```
EXEC CICS LINK PROGRAM('CSQCRST ')
          INPUTMSG('CKQC MODIFY    N D 121')
```

Figure 24. Linking to the adapter reset program, CSQCRST, from a CICS program

Note: The MODIFY command must be padded to 10 characters, see "Command syntax in application programs" on page 40.

Displaying details of connections and CICS tasks

You can use the CICS adapter control panels to display details of the current connection. The equivalent functionality is not available from the CICS command line or from a CICS application program. However, you can obtain some status information using the CKQC DISPLAY command, see "Displaying connection status and in-flight tasks" on page 56.

Displaying details of a connection from the CICS adapter control panels

From the initial panel:
1. Select **Connection** from the menu bar.
2. Select the **Display** action from the pull-down menu.

Figure 25 on page 50 shows the details provided:

```
CKQCM2                        Display Connection panel

Read connection information. Then press F12 to cancel.

  CICS Applid =  VICIC14   Connection Status = Connected       QMgr name = VCA
  Trace Num   = 124        Tracing         = On                API Exit = Off
  Initiation Queue Name = VICIC14.INITIATION.QUEUE
------------------------------ STATISTICS -------------------------------
Number of in-flight tasks =  1           Total API calls =          43912
Number of running CKTI    =  1
        APIs and flows analysis              Syncpoint          Recovery
----------------------------------------- ------------------- --------------
Run OK      43874  MQINQ         6806  Tasks          26  Indoubt       0
Futile          0  MQSET            0  Backout         0  UnResol       0
MQOPEN       6833  ------ Flows ------  Commit         10  Commit        0
MQCLOSE      6823  Calls        43952  S-Phase        10  Backout       0
MQGET       10032  SyncComp     43922  2-Phase         0
 GETWAIT     3399  SuspReqd         0  -------------Task Use-------------
MQPUT       13399  Msg Wait         7  Initial  8   Started  8   Busy  0
MQPUT1          5  Switched     43940




F1=Help  F12=Cancel  Enter=Refresh
```

Figure 25. The display connection panel

The display is organized into three areas:

- Top: parameters used for the connection and current status.
- Middle: connection statistics. These are totals for the current connection, since statistics were last reset.
- Bottom: statistics produced by the adapter.

For an explanation of specific fields on this screen, view the online help panels by pressing function key F1.

Starting an instance of the task initiator CKTI

CKTI is the WebSphere MQ-supplied task initiator (or *trigger monitor* in WebSphere MQ terminology), used in a CICS environment to start a transaction when the trigger conditions on any of its associated queues are met.

You can start a CKTI instance from:

- The CICS adapter control panels
- The CICS command line
- A CICS application program
- Emulated terminals, automatically

Starting CKTI from the CICS adapter control panels

From the initial panel:

1. Select **CKTI** from the menu bar.
2. Select the **Start** action from the pull-down menu.
3. In the **Start Task Initiator** secondary window, use the **Initiation Queue Name** field to specify the name of the initiation queue to be serviced by this CKTI instance. If you leave this field blank, the default initiation queue is used, if defined.

```
   Connection        CKTI          Task
----------------- +-------------------+-------------------------------------
CKQCM0            | Select an action. |-- Initial panel
                  |                   |
Select menu bar it| 1 1. Start...     |press Enter.
                  |    2. Stop...     |
                  |    3. Display     |
                  |                   |
          +--- +-------------------------------------------------+
          | F1 |               Start Task Initiator              |
          +--- |                                                 |
               | Type Initiation Queue Name. Then press Enter.   |
               |                                                 |
               | Initiation Queue Name (IQ) . . . . . . . . . .  |
               |   CICS01.INITIATION.QUEUE2                       |
               |                                                 |
          +-------------------------------------------------+
IBM WebSphere MQ for z/O| F1=Help  F12=Cancel                      |
          +-------------------------------------------------+

(C) Copyright IBM Corporation 1993, 2005 All rights reserved.


F1=Help  F3=Exit
```

Figure 26. Starting an instance of CKTI

Starting CKTI from the CICS command line

The command shown in Figure 27 starts an instance of CKTI to serve the default initiation queue, if defined.

```
CKQC STARTCKTI
```

Figure 27. Starting an instance of CKTI, for the default initiation queue

The command shown in Figure 28 starts an instance of CKTI to serve a specified initiation queue.

```
CKQC STARTCKTI CICS01.INITIATION.QUEUE2
```

Figure 28. Starting an instance of CKTI, for a specified initiation queue

Starting CKTI from a CICS application program

To start an instance of CKTI from a CICS program, the program must link to the adapter task initiation program, CSQCSSQ. Figures Figure 29 and Figure 30 on page 52 show example LINK commands. When you do an EXEC CICS LINK to CSQCSSQ, the program requires a terminal associated task.

```
EXEC CICS LINK PROGRAM('CSQCSSQ ')
         INPUTMSG('CKQC STARTCKTI ')
```

Figure 29. Linking to the adapter task-initiator program CSQCSSQ from CICS. This starts a CKTI that uses the default initiation queue.

```
EXEC CICS LINK PROGRAM('CSQCSSQ ')
          INPUTMSG('CKQC STARTCKTI  CICS01.INITIATION.QUEUE2')
```

Figure 30. Linking to the adapter task-initiator program CSQCSSQ from CICS. This starts a CKTI that uses a named initiation queue.

Output messages from CSQCSSQ are displayed on the system console.

Note: The STARTCKTI command must be padded to 10 characters; see "Command syntax in application programs" on page 40.

Starting CKTI automatically

To automate the starting of CKTIs under a specific user ID, you can use an automation product, for example, NetView®. You can use this to sign on to a CICS console and issue the STARTCKTI command.

You can also use preset security sequential terminals, which have been defined to emulate a CRLP terminal, with the sequential terminal input containing the CKQC STARTCKTI command.

However, when the CICS adapter alert monitor reconnects CICS to WebSphere MQ, for example, after a queue manager restart, only the CKTI specified at the initial WebSphere MQ connection is restarted. You must automate starting any extra CKTIs yourself.

Stopping an instance of CKTI

You can stop an instance of CKTI by using:
*   The CICS adapter control panels
*   The CICS command line
*   A CICS application program

Stopping an instance of CKTI from the CICS adapter control panels

From the initial panel:
1. Select **CKTI** from the menu bar.
2. Select the **Stop** action from the pull-down menu.
3. Use the **Stop Task Initiator** secondary window to specify the name of the initiation queue serviced by this instance of CKTI. If you leave the name blank, the default initiation queue is used, if defined.

```
   Connection        CKTI        Task
------------------ +-------------------+------------------------------------
CKQCM0             | Select an action. |-- Initial panel
                   |                   |
Select menu bar it | 2 1. Start...     |press Enter.
                   |   2. Stop...      |
                   |   3. Display      |
                   |                   |
         +--- +-------------------------------------------------+
         | F1 |              Stop Task Initiator                |
         +--- |                                                 |
              |   Type Initiation Queue Name. Then press Enter. |
              |                                                 |
              |   Initiation Queue Name (IQ) . . . . . . . . . .|
              |   CICS01.INITIATION.QUEUE2                       |
              |                                                 |
         +-------------------------------------------------+    |
IBM WebSphere MQ for z/O| F1=Help  F12=Cancel             |    |
         +-------------------------------------------------+----+

(C) Copyright IBM Corporation 1993, 2005 All rights reserved.


F1=Help  F3=Exit
```
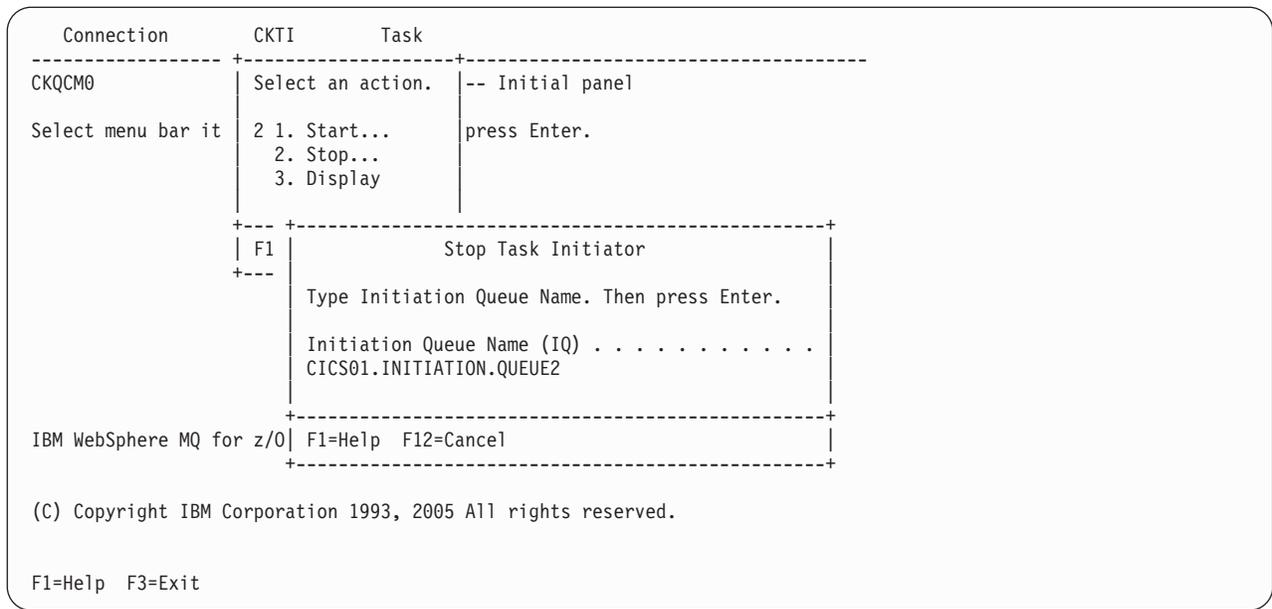
Figure 31. Stopping an instance of the task initiator CKTI

Stopping an instance of CKTI from the command line

The command shown in Figure 32 stops an instance of CKTI that is serving the default initiation queue, if there is one.

```
CKQC STOPCKTI
```

Figure 32. Stopping an instance of CKTI from the command line, for the default initiation queue

The command shown in Figure 33 stops the instance of CKTI that is serving a specified initiation queue.

```
CKQC STOPCKTI CICS01.INITIATION.QUEUE2
```

Figure 33. Stopping an instance of CKTI from the command line, for a specified initiation queue

Stopping an instance of CKTI from an application program

You can stop an instance of CKTI by linking to the adapter task-initiator program, CSQCSSQ. Figures Figure 34 and Figure 35 on page 54 show alternative LINK commands to stop an instance of CKTI from a CICS program. The first command stops the CKTI that is serving the default initiation queue; the second stops the CKTI serving a specified initiation queue.

```
EXEC CICS LINK PROGRAM('CSQCSSQ ')
          INPUTMSG('CKQC STOPCKTI  ')
```

Figure 34. Stopping an instance of CKTI from a program—for the default initiation queue from CICS

```
EXEC CICS LINK PROGRAM('CSQCSSQ ')
          INPUTMSG('CKQC STOPCKTI   CICS01.INITIATION.QUEUE2')
```

Figure 35. Stopping an instance of CKTI from a program—for a specified initiation queue from CICS

Note: The STOPCKTI command must be padded to 10 characters; see "Command syntax in application programs" on page 40.

Displaying the current instances of CKTI

You can use the CICS adapter control panels to display details of the current instances of CKTI. The equivalent functionality is not available from the CICS command line or from a CICS application program.

Displaying the current instances of CKTI from the CICS adapter control panels

From the initial panel:

1. Select **CKTI** from the menu bar
2. Select the **Display** action from the pull-down menu

Figure 36 shows the details provided for each instance of CKTI:

- CICS task number
- Task status
- Thread status
- Number of API calls it has issued
- Most recent API call it has issued
- Name of the initiation queue it is serving

Press function key F1 to display help information about each field in this panel.

```
CKQCM4                     Display CKTI panel

Read CKTI status information. Then press F12 to cancel.

CKTI   1 to   1 of   1

 Task Num    Task Status    Thread Status    Num of APIs    Last API
----------  -------------  ---------------  -------------  ----------
 0000123       Normal         Msg Wait               2       MQGET
   Initiation Queue Name: CICS01.INITIATION.QUEUE1







 F1=Help  F7=Backward  F8=Forward  F12=Cancel  Enter=Refresh
```

Figure 36. The CKQC Display CKTI panel

Displaying CICS task information

You can use the CICS adapter control panels to display information about CICS tasks using MQI calls. The equivalent functionality is not available from the CICS command line or from a CICS application program. However, you can obtain some status information using the CKQC DISPLAY command, see "Displaying connection status and in-flight tasks" on page 56.

Displaying CICS tasks from the CICS adapter control panels

You can display information about the CICS tasks that are currently using MQI calls. From the initial panel:

1. Select **Task** from the menu bar.
2. Select an action from the pull-down menu.

   Select option 1, **List all tasks** to obtain information about all tasks that are currently active. To limit the scope of the display, select option 2, **List from task**, to specify the starting number of the first task to be displayed.
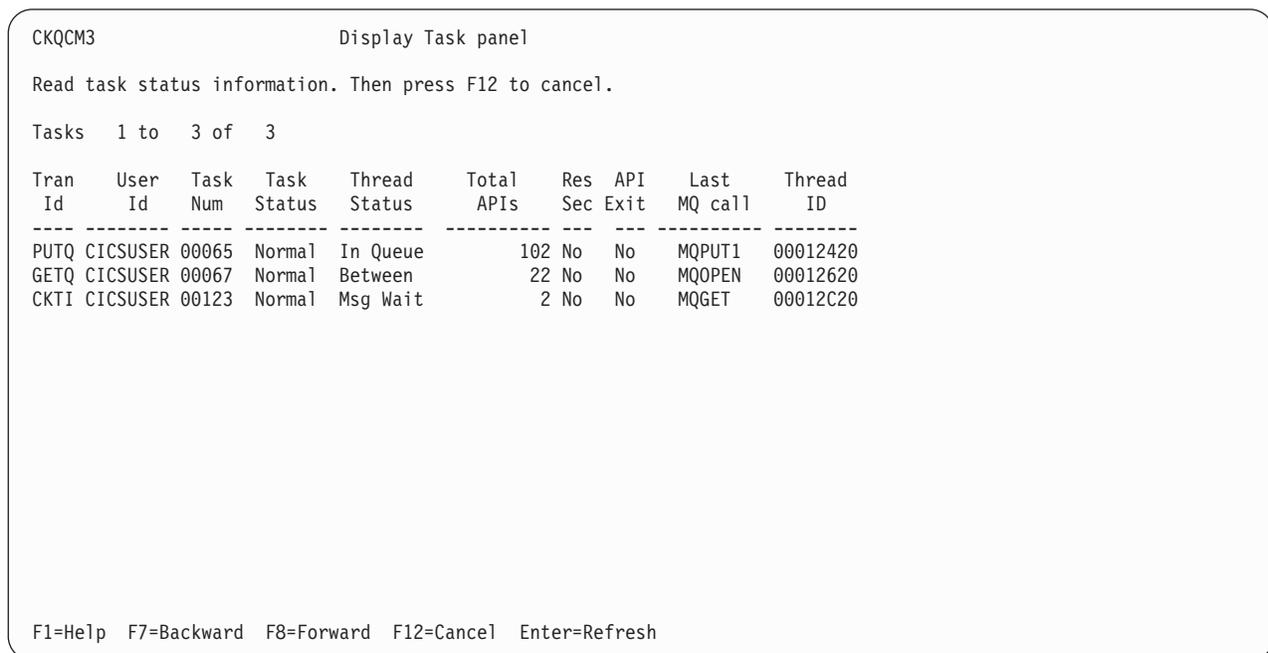
```
CKQCM3                      Display Task panel

Read task status information. Then press F12 to cancel.

Tasks   1 to   3 of   3

Tran    User    Task  Task    Thread    Total    Res API   Last     Thread
 Id      Id     Num   Status  Status     APIs    Sec Exit  MQ call    ID
---- -------- ----- -------- -------- ---------- --- --- ---------- --------
PUTQ CICSUSER 00065  Normal  In Queue     102 No   No    MQPUT1   00012420
GETQ CICSUSER 00067  Normal  Between       22 No   No    MQOPEN   00012620
CKTI CICSUSER 00123  Normal  Msg Wait       2 No   No    MQGET    00012C20




















 F1=Help  F7=Backward  F8=Forward  F12=Cancel  Enter=Refresh
```

Figure 37. The CKQC Display Task panel

Figure 37 shows the details provided for each CICS task:
- Transaction ID (name)
- User ID
- CICS task number
- Task status
- Thread status
- Total number of API calls issued by this task
- Whether resource security checking is active for this task
- Whether this task is currently in the API-crossing exit
- Most recent API call issued by this task
- Thread ID used by WebSphere MQ

Displaying connection status and in-flight tasks

You can use the CKQC DISPLAY command to display limited information about the current connection and CICS tasks from either the CICS command line or from a CICS application program. The information from this command is returned in a message CSQC453I, as shown in Figure 38. This message contains:

- The name of the queue manager
- The status of the connection
- The number of in-flight tasks that are still using the connection

```
CSQC453I VICY06 CSQCDSPL Status of connection to JAC2 is Connected. 2
tasks are in-flight
```

Figure 38. Message showing the status of a connection

To obtain more detailed information, use the CICS adapter control panels. See "Displaying details of a connection from the CICS adapter control panels" on page 49 and "Displaying CICS tasks from the CICS adapter control panels" on page 55.

From the CICS command line:

You can use the CKQC DISPLAY command, shown in Figure 39, from the CICS command line.

```
CKQC DISPLAY
```

Figure 39. Displaying the status of a connection

Figure 38 shows a typical response to this command. The response messages are sent to your CICS terminal.

From a CICS application program:

Figure 40 shows the LINK command for displaying the status of a connection from a CICS application program.

```
EXEC CICS LINK PROGRAM('CSQCDSPL') INPUTMSG('CKQC DISPLAY')
```

Figure 40. Linking to the adapter program CSQCDSPL from a CICS program

Figure 38 shows a typical output from this command. The response messages are sent to the CKQQ queue (the transient data queue).

The COMMAREA option can be used instead of INPUTMSG, but only when the program is run at PLT time.

Purging tasks that are using the CICS adapter

You can use the CICS CEMT transaction to purge user tasks that are using the CICS adapter. Tasks that are waiting on the adapter respond only to CEMT SET TASK FORCEPURGE commands—CEMT SET TASK PURGE commands are ignored. The way the adapter handles a FORCEPURGE command depends on the kind of wait state that the task is in:

- If a task is waiting for a message to arrive, for example, the application has issued an **MQGET WAIT** call, the task is stopped with code AEXY immediately.
- If a task is waiting for an MQI request to be completed by WebSphere MQ, message CSQC413I is displayed on the system console.

  The adapter waits for the request to complete, and then checks whether it is suitable to end the task:

  – If the task is in a critical state, the CICS adapter lets the task continue and ignores the attempt to purge it. This is done to preserve data and system integrity. Message CSQC415I is displayed.

    A task is in a critical state is when, for example, it is in the process of completing phase 2 of a two-phase commit sequence.

  – If the task is not in a critical state, the adapter ends it with code AEXY. Message CSQC414I is displayed.

For information about CEMT commands, see the *CICS-Supplied Transactions* manual.

Shutting down a connection between WebSphere MQ and the CICS adapter

You can shut down a connection between WebSphere MQ and the CICS adapter by using the CKQC transaction or an application program. There are two types of shutdown:

- Forced
- Quiesced

Other forms of connection shutdown result from a termination of CICS or the queue manager. Table 2 summarizes how the adapter handles different forms of connection shutdown.

Table 2. Shutting down a CICS adapter connection

Method of shutdown	How this is handled by the adapter
CKQC STOP (*A quiesced shutdown*)	Mark the status of the adapter as *Quiescing*. Allow both active and waiting tasks to complete. Allow syncpoint. Do not allow calls from a new task. The last task initiates disconnection from WebSphere MQ.
CKQC STOP FORCE	Mark the status of the adapter as *StoppingForce*. Disconnect from WebSphere MQ. Resume waiting tasks. Fail any in-flight or following MQI calls.
CICS warm shutdown	Issue message CSQC411I. Initiate a quiesced shutdown of the connection; see CKQC STOP, above.
CICS immediate shutdown	Issue message CSQC410I. Any in-flight tasks using WebSphere MQ are backed out.
CICS abend	Issue message CSQC412I.
WebSphere MQ quiesced	Initiate a quiesced shutdown of the connection; see CKQC STOP, above.
WebSphere MQ abend or forced shutdown	Initiate a forced shutdown of connection; see CKQC STOP FORCE, above.

Table 2. Shutting down a CICS adapter connection (continued)

Method of shutdown	How this is handled by the adapter
Notes:	
1. If the connection is not active (for example, quiesced) when CICS or the queue manager shuts down, no action is taken and no messages are issued.	
2. "Waiting tasks" includes instances of CKTI, which you must stop before shutdown completes.	

Orderly shutdown

An orderly shutdown of the connection lets each CICS transaction terminate before thread subtasks are detached. When you use this method, there should be no in-doubt units of work when you reconnect CICS. An orderly termination occurs in each of the following situations:

- The CICS terminal operator issues a CKQC STOP command. CICS and the queue manager remain active. The command can be issued from the command line, from a terminal using the CKQC panels, or from a program, see topic "Stopping a connection from a CICS application program" on page 46.
- The CICS terminal operator issues the CICS command:

```
CEMT PERFORM SHUTDOWN
```

For information about the CEMT PERFORM SHUTDOWN command, see the *CICS-Supplied Transactions* manual.

- The queue manager is quiesced by the command:

```
+CSQ1 STOP QMGR MODE(QUIESCE)
```

This stops the queue manager, allows the currently identified tasks to continue normal processing, and does not allow new tasks to identify themselves to the queue manager. CICS remains active.

Forced shutdown

A forced shutdown of the connection can abnormally end CICS transactions connected to the queue manager. Therefore, there might be in-doubt units of work when the system is reconnected. A forced shutdown occurs in each of these situations:

- The CICS terminal operator issues the CKQC STOP FORCE command. The command can be issued from the command line, from a terminal using the CKQC panels, or from a program, (see topic "Stopping a connection from a CICS application program" on page 46).
- The CICS terminal operator issues the CICS immediate termination command:

```
CEMT PERFORM SHUTDOWN IMMEDIATE
```

The queue manager remains active.

For information about this command, see the *CICS-Supplied Transactions* manual.

• The WebSphere MQ forced termination command is issued:

```
+CSQ1 STOP QMGR MODE(FORCE) or +CSQ1 STOP QMGR MODE(RESTART)
```

> CICS remains active.
• A WebSphere MQ abend occurs. CICS remains active.
• CICS abend occurs. The queue manager remains active.

Operating the CICS bridge

This chapter describes how to operate the WebSphere MQ CICS bridge. It discusses the following topics:
• "Starting the CICS bridge"
• "Shutting down the CICS bridge" on page 60

For information on how to write programs that will interact with existing CICS transactions and programs using the WebSphere MQ CICS bridge, see WebSphere MQ Application Programming Guide. For information on how to set up the queues and other definitions needed by the bridge see WebSphere MQ for z/OS System Setup Guide. For further details of the MQCIH and other MQ headers see WebSphere MQ Application Programming Guide. For details of the CICS 3270 bridge see 'Bridging to 3270 transactions' in the *CICS External Interfaces Guide*.

Starting the CICS bridge

To start the bridge, run the CICS bridge transaction. The default transaction is CKBR but you might have defined a different one; it has six optional parameters:
• Q=*qqq*, where *qqq* is the name of the queue holding requests. If you do not specify a queue name, the default is SYSTEM.CICS.BRIDGE.QUEUE.

  Remember that names of objects within WebSphere MQ are case-sensitive.
• AUTH=*LOCAL*|*IDENTIFY*|*VERIFY_UOW*|*VERIFY_ALL*. The default is LOCAL. All options are described in WebSphere MQ for z/OS System Setup Guide.
• WAIT=*nnn*, where *nnn* is the number of seconds that you want the bridge task to wait for subsequent requests before timing out when processing a unit of work that runs many user programs.

  The default wait time is unlimited.

  You are recommended to specify a wait time. If you do not specify a wait time, the CICS bridge might inhibit CICS or queue manager shut down.
• MSG=*CSMT*|*LOG*|*BOTH*, where the option determines whether messages generated by the CICS bridge will be sent to the CICS job log, the CICS master terminal, or both. The default is BOTH.
• PASSTKTA=*applid*, where *applid* specifies the application id to be used for validating the passticket. The default is the CICS region application ID. See WebSphere MQ for z/OS System Setup Guide for details of passtickets.
• ROUTEMEM=*Y*|*N*, where the option determines whether mark expired messages are to be routed to the DLQ. If you do not specify the parameter the default is N.

Start the CKBR task running by using one of the following methods:

- Input a single line from a terminal (3270 or other). The format is:

```
CKBR Q=<queue name>,AUTH=<auth option>,WAIT=nnn,MSG=<msg option>,PASSTKTA=<applid>,ROUTEMEM=<routemem option>
```

For example:

```
CKBR Q=MyQueue,AUTH=IDENTIFY,WAIT=30,MSG=LOG,PASSTKTA=APP1,ROUTEMEM=Y
```

The terminal will be unlocked so it can be used for other work.
- Issue an EXEC CICS START for the CKBR transaction with the parameters as data.

  You can have a program which runs as part of CICS PLTPI processing which issues this command, and specify the userid which the bridge monitor transaction is to run under.
- Issue an EXEC CICS LINK to the program CSQCBR00 with the parameters as data in the commarea.

  CSQCBR00 is a long running task and this program will only return when the bridge stops.
- Use TRIGGER TRIGTYPE(FIRST) on the bridge request queue to start a process specifying APPLICID(CKBR), with any parameters in USERDATA. However, you can not specify the Q=*qqq* parameter in USERDATA.

If you are running multiple bridge monitors sharing a queue, you can start one of the monitors by putting a message onto the bridge request queue. However for private local queues only one trigger message will be produced, so the bridge monitor will only start on one CICS region. You should therefore consider alternative ways of starting the bridge monitors, for example using a program in the CICS startup PLT processing to start the transaction with the required parameters or using automation products to start the transaction.

The level of security you want to use influences how you start the monitor task. See the WebSphere MQ for z/OS System Setup Guide for information on the security options available to you.

Shutting down the CICS bridge

There are various ways in which you can shut down the CICS bridge:
- By altering the attributes of the request queue by setting GET(DISABLED). If you do this, remember to reset GET(ENABLED) after the bridge has shut down
- By shutting down CICS
- By shutting down the queue manager

Chapter 3. WebSphere MQ and IMS

Operating the IMS adapter

This chapter describes how to operate the IMS adapter, which connects WebSphere MQ to IMS systems.

Note: The IMS adapter does not incorporate any operations and control panels.

This chapter contains the following sections:
- "Controlling IMS connections"
- "Connecting from the IMS control region"
- "Displaying in-doubt units of recovery" on page 64
- "Controlling IMS dependent region connections" on page 65
- "Disconnecting from IMS" on page 68
- "Controlling the IMS trigger monitor" on page 68

Controlling IMS connections

IMS provides the following operator commands to control and monitor the connection to WebSphere MQ:

/CHANGE SUBSYS
Deletes an in-doubt unit of recovery from IMS.

/DISPLAY OASN SUBSYS
Displays outstanding recovery elements.

/DISPLAY SUBSYS
Displays connection status and thread activity.

/START SUBSYS
Connects the IMS control region to a queue manager.

/STOP SUBSYS
Disconnects IMS from a queue manager.

/TRACE
Controls the IMS trace.

For more information about these commands, see the *IMS/ESA Operator's Reference* manual for the level of IMS that you are using.

IMS command responses are sent to the terminal from which the command was issued. Authorization to issue IMS commands is based on IMS security.

Connecting from the IMS control region

IMS makes one connection from its control region to each queue manager that uses IMS. IMS must be enabled to make the connection in one of these ways:
- Automatically during either:
  - A cold-start initialization.

– A warm start of IMS, if the WebSphere MQ connection was active when IMS was shut down.

• In response to the IMS command:

```
/START SUBSYS sysid
```

where *sysid* is the queue manager name.

The command can be issued regardless of whether the queue manager is active.

The connection is not actually made until the first MQ API call to the queue manager is made. Until that time, the IMS command /DIS SUBSYS will show the status as 'NOT CONN'.

The order in which you start IMS and the queue manager is not significant.

IMS cannot re-enable the connection to the queue manager automatically if the queue manager is stopped with a STOP QMGR command, the IMS command /STOP SUBSYS, or an abnormal end. Therefore, you must make the connection by using the IMS command /START SUBSYS.

Initializing the adapter and connecting to the queue manager

The adapter is a set of modules loaded into the IMS control and dependent regions, using the IMS external Subsystem Attach Facility.

This procedure initializes the adapter and connects to the queue manager:

1. Read the subsystem member (SSM) from IMS.PROCLIB. The SSM chosen is an IMS EXEC parameter. There is one entry in the member for each queue manager to which IMS can connect. Each entry contains control information about a WebSphere MQ adapter.
2. Load the IMS adapter.

   Note: IMS loads one copy of the adapter modules for each WebSphere MQ instance that is defined in the SSM member.
3. Attach the external subsystem task for WebSphere MQ.
4. Run the adapter with the CTL EXEC parameter (IMSID) as the connection name.

The process is the same whether the connection is part of initialization or a result of the IMS command /START SUBSYS.

If the queue manager is active when IMS tries to make the connection, the following messages are sent:

• to the z/OS console:

```
DFS3613I ESS TCB INITIALIZATION COMPLETE
```

• to the IMS master terminal:

```
CSQQ000I IMS/TM imsid connected to queue manager ssnm
```

When IMS tries to make the connection and *the queue manager is not active*, the following messages are sent to the IMS master terminal each time an application makes an MQI call:

```
CSQQ001I IMS/TM imsid not connected to queue manager ssnm.
         Notify message accepted
DFS3607I MQM1     SUBSYSTEM  ID  EXIT FAILURE, FC = 0286, RC = 08,
JOBNAME = IMSEMPR1
```

If you get DFS3607I messages when you start the connection to IMS or on system startup, this indicates that the queue manager is not available. To prevent a large numbers of messages being generated, you must do one of the following:

1. Start the relevant queue manager.
2. Issue the IMS command:

```
/STOP SUBSYS
```

so that IMS does not expect to connect to the queue manager.

If you do neither, a DFS3607I message and the associated CSQQ001I message are issued each time a job is scheduled in the region and each time a connection request to the queue manager is made by an application.

Thread attachment

In an MPP or IFP region, IMS makes a thread connection when the first application program is scheduled into that region, even if that application program does not make a WebSphere MQ call. In a BMP region, the thread connection is made when the application makes its first WebSphere MQ call (**MQCONN** or **MQCONNX**). This thread is retained for the duration of the region or until the connection is stopped.

For both the message driven and non-message driven regions, the recovery thread cross-reference identifier, *Thread-xref*, associated with the thread is:

```
PSTid + PSBname
```

where:

PSTid   Partition specification table region identifier

PSBname
         Program specification block name

You can use connection IDs as unique identifiers in WebSphere MQ commands, in which case WebSphere MQ automatically inserts these IDs into any operator message that it generates.

Displaying in-doubt units of recovery

The operational steps used to list and recover in-doubt units of recovery are discussed here for relatively simple cases only.

If the queue manager ends abnormally while connected to IMS, IMS might commit or back out work without WebSphere MQ being aware of it. When the queue manager restarts, that work is termed *in doubt*. A decision must be made about the status of the work.

To display a list of in-doubt units of recovery, issue the command:

```
+CSQ1 DISPLAY CONN(*) WHERE(UOWSTATE EQ UNRESOLVED)
```

WebSphere MQ responds with a message like the following:

```
CSQM201I +CSQ1 CSQMDRTC  DIS CONN DETAILS
CONN(BC0F6125F5A30001)
EXTCONN(C3E2D8C3C3E2D8F140404040404040404040)
TYPE(CONN)
CONNOPTS(
 MQCNO_STANDARD_BINDING
 )
UOWLOGDA(2004-11-02)
UOWLOGTI(12.27.58)
UOWSTDA(2004-11-02)
UOWSTTI(12.27.58)
UOWSTATE(UNRESOLVED)
NID(CSQ1CHIN.BC0F5F1C86FC0766)
EXTURID(000000000000001F000000007472616E5F6964547565204E6F762020...)
QMURID(000000026232)
URTYPE(XA)
USERID( )
APPLTAG(CSQ1CHIN)
ASID(0000)
APPLTYPE(CHINIT)
CHANNEL( )
CONNAME( )
 END CONN DETAILS
```

For an explanation of the attributes in this message see the description of the DISPLAY CONN command in WebSphere MQ Script (MQSC) Command Reference.

Recovering in-doubt units of recovery

To recover in-doubt units of recovery, issue this command:

```
+CSQ1 RESOLVE INDOUBT(connection-name) ACTION(COMMIT|BACKOUT)
           NID(net-node.number)
```

where:

connection-name
> The IMS system ID.

ACTION
Indicates whether to commit (COMMIT) or back out (BACKOUT) this unit of recovery.

net-node.number
The associated net-node.number.

When you have issued the RESOLVE INDOUBT command, one of the following messages is displayed:

```
CSQV414I +CSQ1 THREAD network-id COMMIT SCHEDULED

CSQV415I +CSQ1 THREAD network-id BACKOUT SCHEDULED
```

Resolving residual recovery entries

At given times, IMS builds a list of residual recovery entries (RREs). RREs are units of recovery about which WebSphere MQ might be in doubt. They arise in several situations:

- If the queue manager is not active, IMS has RREs that cannot be resolved until the queue manager is active. These RREs are not a problem.
- If the queue manager is active and connected to IMS, and if IMS backs out the work that WebSphere MQ has committed, the IMS adapter issues message CSQQ010E. If the data in the two systems must be consistent, there is a problem. Resolving this problem is discussed in "Recovering IMS units of recovery manually" on page 128.
- If the queue manager is active and connected to IMS, there might still be RREs even though no messages have informed you of this problem. After the WebSphere MQ connection to IMS has been established, you can issue the following IMS command to find out if there is a problem:

```
/DISPLAY OASN SUBSYS sysid
```

To purge the RRE, issue one of the following IMS commands:

```
/CHANGE SUBSYS sysid RESET
/CHANGE SUBSYS sysid RESET OASN nnnn
```

where *nnnn* is the originating application sequence number listed in response to your +CSQ1 DISPLAY command. This is the schedule number of the program instance, giving its place in the sequence of invocations of that program since the last IMS cold start. IMS cannot have two in-doubt units of recovery with the same schedule number.

These commands reset the status of IMS; they do not result in any communication with WebSphere MQ.

Controlling IMS dependent region connections

Controlling IMS dependent region connections involves the following activities:

- Connecting from dependent regions

- Region error options
- Monitoring the activity on connections
- Disconnecting from dependent regions

Connecting from dependent regions

The IMS adapter used in the control region is also loaded into dependent regions. A connection is made from each dependent region to WebSphere MQ. This connection is used to coordinate the commitment of WebSphere MQ and IMS work. To initialize and make the connection, IMS does the following:

1. Reads the subsystem member (SSM) from IMS.PROCLIB.

   A subsystem member can be specified on the dependent region EXEC parameter. If it is not specified, the control region SSM is used. If the region is never likely to connect to WebSphere MQ, to avoid loading the adapter, specify a member with no entries.

2. Loads the WebSphere MQ adapter.

   For a batch message program, the load is not done until the application issues its first messaging command. At that time, IMS tries to make the connection.

   For a message-processing program region or IMS fast-path region, the attempt is made when the region is initialized.

Region error options

If the queue manager is not active, or if resources are not available when the first messaging command is sent from application programs, the action taken depends on the error option specified on the SSM entry. The options are:

R   The appropriate return code is sent to the application.

Q   The application ends abnormally with abend code U3051. The input message is re-queued.

A   The application ends abnormally with abend code U3047. The input message is discarded.

Monitoring the activity on connections

A thread is established from a dependent region when an application makes its first successful WebSphere MQ request. You can display information about connections and the applications currently using them by issuing the following command from WebSphere MQ:

```
+CSQ1 DISPLAY CONN(*)
```

The command produces a message like the following:

```
CONN(BC45A794C4290001)
 EXTCONN(C3E2D8C3C3E2D8F14040404040404040)
 TYPE(CONN)
 CONNOPTS(
  MQCNO_STANDARD_BINDING
  )
 UOWLOGDA(2004-12-15)
 UOWLOGTI(16.39.43)
 UOWSTDA(2004-12-15)
 UOWSTTI(16.39.43)
 UOWSTATE(ACTIVE)
 NID( )
 EXTURID(
  0000052900000000
  )
 QMURID(00000354B76E)
 URTYPE(IMS)
 USERID(STCPI)
 APPLTAG(IM8F)
 ASID(0049)
 APPLTYPE(IMS)
 PSTID(0004)
 PSBNAME(GP01MPP)
```

For the control region, *thread-xref* is the special value CONTROL. For dependent regions, it is the PSTid concatenated with the PSBname. *auth-id* is either the user field from the job card, or the ID from the z/OS started procedures table.

For an explanation of the displayed list, see the description of message CSQV402I in the WebSphere MQ for z/OS Messages and Codes manual.

IMS provides a display command to monitor the connection to WebSphere MQ. It shows which program is active on each dependent region connection, the LTERM user name, and the control region connection status. The command is:

```
/DISPLAY SUBSYS name
```

The status of the connection between IMS and WebSphere MQ is shown as one of:

```
 CONNECTED
 NOT CONNECTED
 CONNECT IN PROGRESS
 STOPPED
 STOP IN PROGRESS
 INVALID SUBSYSTEM NAME=name
 SUBSYSTEM name NOT DEFINED BUT RECOVERY OUTSTANDING
```

The thread status from each dependent region is one of the following:

```
 CONN
 CONN, ACTIVE (includes LTERM of user)
```

Disconnecting from dependent regions

To change values in the SSM member of IMS.PROCLIB, you disconnect a
dependent region. To do this, you must:

1. Issue the IMS command:

```
/STOP REGION
```

2. Update the SSM member.
3. Issue the IMS command:

```
/START REGION
```

Disconnecting from IMS

The connection is ended when either IMS or the queue manager terminates.
Alternatively, the IMS master terminal operator can explicitly break the connection
by issuing the following IMS command:

```
/STOP SUBSYS sysid
```

The command sends the following message to the terminal that issued it, usually
the master terminal operator (MTO):

```
DFS058I STOP COMMAND IN PROGRESS
```

The IMS command:

```
/START SUBSYS sysid
```

is required to reestablish the connection.

Note: The IMS command /STOP SUBSYS will not be completed if an IMS trigger
monitor is running.

Controlling the IMS trigger monitor

The IMS trigger monitor (the CSQQTRMN transaction) is described in the
WebSphere MQ for z/OS Concepts and Planning Guide.

Starting CSQQTRMN

1. Start a batch oriented BMP running the program CSQQTRMN for each
   initiation queue you want to monitor.
2. Modify your batch JCL (described in the WebSphere MQ for z/OS System
   Setup Guide) to add a DDname of CSQQUT1 that points to a data set
   containing the following information:

```
QMGRNAME=q_manager_name        Comment: queue manager name
INITQUEUENAME=init_q_name      Comment: initiation queue name
LTERM=lterm                    Comment: LTERM to remove error messages
CONSOLEMESSAGES=YES            Comment: Send error messages to console
```

where:

q_manager_name	The name of the queue manager (if this is blank, the default nominated in CSQQDEFV is assumed)
init_q_name	The name of the initiation queue to be monitored
lterm	The IMS LTERM name for the destination of error messages (if this is blank, the default value is MASTER).
CONSOLEMESSAGES=YES	Requests that messages sent to the nominated IMS LTERM are also sent to the z/OS console. If this parameter is omitted or misspelled, the default is NOT to send messages to the console.

3. Add a DD name of CSQQUT2 if you want a printed report of the processing of CSQQUT1 input.

Note:

1. The data set CSQQUT1 is defined with LRECL=80. Other DCB information is taken from the data set. The DCB for data set CSQQUT2 is RECFM=VBA and LRECL=125.
2. You can put only one keyword on each record. The keyword value is delimited by the first blank following the keyword; this means that you can include comments. An asterisk in column 1 means that the whole input record is a comment.
3. If you misspell either of the QMGRNAME or LTERM keywords, CSQQTRMN uses the default for that keyword.
4. Ensure that the subsystem is started in IMS (by the /START SUBSYS command) before submitting the trigger monitor BMP job. If it is not started, your trigger monitor job terminates with abend code U3042.

Stopping CSQQTRMN

Once started, CSQQTRMN runs until either the connection between WebSphere MQ and IMS is broken due to one of the following events:

- the queue manager ending
- IMS ending

or a z/OS STOP jobname command is entered.

Controlling the IMS bridge

This chapter describes how to control the IMS bridge. It discusses the following topics:

- "Starting and stopping the IMS bridge" on page 70
- "Controlling IMS connections" on page 70
- "Controlling bridge queues" on page 71
- "Resynchronizing the IMS bridge" on page 72
- "Deleting messages from IMS" on page 73

- "Deleting Tpipes" on page 73

There are no WebSphere MQ commands to control the WebSphere MQ-IMS bridge. However, you can stop messages being delivered to IMS in the following ways:

- For non-shared queues, by using the ALTER QLOCAL(xxx) GET(DISABLED) command for all bridge queues.
- For clustered queues, by using the SUSPEND QMGR CLUSTER(xxx) command. This is only effective when another queue manager is also hosting the clustered bridge queue.
- For clustered queues, by using the SUSPEND QMGR FACILITY(IMSBRIDGE) command. No further messages are sent to IMS, but the responses for any outstanding transactions are received from IMS.

  To start sending messages to IMS again, issue the RESUME QMGR FACILITY(IMSBRIDGE) command.

See WebSphere MQ Script (MQSC) Command Reference for details of these commands.

Starting and stopping the IMS bridge

Start the WebSphere MQ bridge by starting OTMA. Either use the IMS command:

```
/START OTMA
```

or start it automatically by specifying OTMA=YES in the IMS system parameters. If OTMA is already started, the bridge starts automatically when queue manager startup has completed. A WebSphere MQ event message is produced when OTMA is started.

Use the IMS command:

```
/STOP OTMA
```

to stop OTMA communication. When this command is issued, a WebSphere MQ event message is produced.

Controlling IMS connections

IMS provides these operator commands to control and monitor the connection to WebSphere MQ:

/DEQUEUE TMEMBER *tmember* **TPIPE** *tpipe*
Removes messages from a Tpipe. Specify PURGE to remove all messages or PURGE1 to remove the first message only.

/DISPLAY OTMA
Displays summary information about the OTMA server and clients, and client status.

/DISPLAY TMEMBER *name*
Displays information about an OTMA client.

/DISPLAY TRACE TMEMBER *name*
Displays information about what is being traced.

/SECURE OTMA
Sets security options.

/START OTMA
Enables communications through OTMA.

/START TMEMBER *tmember* **TPIPE** *tpipe*
Starts the named Tpipe.

/STOP OTMA
Stops communications through OTMA.

/STOP TMEMBER *tmember* **TPIPE** *tpipe*
Stops the named Tpipe.

/TRACE
Controls the IMS trace.

For more information about these commands, see the *IMS/ESA Operator's Reference* manual for the level of IMS that you are using.

IMS command responses are sent to the terminal from which the command was issued. Authorization to issue IMS commands is based on IMS security.

Controlling bridge queues

To stop communicating with the queue manager with XCF member name *tmember* through the bridge, issue the following IMS command:

```
/STOP TMEMBER tmember TPIPE ALL
```

To resume communication, issue the following IMS command:

```
/START TMEMBER tmember TPIPE ALL
```

The Tpipes for a queue can be displayed using the MQ DISPLAY QUEUE command.

To stop communication with the queue manager on a single Tpipe, issue the following IMS command:

```
/STOP TMEMBER tmember TPIPE tpipe
```

One or two Tpipes are created for each active bridge queue, so issuing this command stops communication with the WebSphere MQ queue. To resume communication, use the following IMS command :

```
/START TMEMBER tmember TPIPE tpipe
```

Alternatively, you can alter the attributes of the WebSphere MQ queue to make it get inhibited.

Resynchronizing the IMS bridge

The IMS bridge is automatically restarted whenever the queue manager, IMS, or OTMA are restarted.

The first task undertaken by the IMS bridge is to resynchronize with IMS. This involves WebSphere MQ and IMS checking sequence numbers on every synchronized Tpipe. A synchronized Tpipe is used when persistent messages are sent to IMS from a WebSphere MQ-IMS bridge queue using commit mode zero (commit-then-send).

If the bridge cannot resynchronize with IMS, the IMS sense code is returned in message CSQ2023E and the connection to OTMA is stopped. If the bridge cannot resynchronize with an individual IMS Tpipe, the IMS sense code is returned in message CSQ2025E and the Tpipe is stopped. If a Tpipe has been cold started, the recoverable sequence numbers are automatically reset to 1.

If the bridge discovers mismatched sequence numbers when resynchronizing with a Tpipe, message CSQ2020E is issued. Use the WebSphere MQ command RESET TPIPE to initiate resynchronization with the IMS Tpipe. You need to provide the XCF group and member name, and the name of the Tpipe; this information is provided by the message.

You can also specify:
- A new recoverable sequence number to be set in the Tpipe for messages sent by WebSphere MQ, and to be set as the partner's receive sequence number. If you do not specify this, the partner's receive sequence number is set to the current WebSphere MQ send sequence number.
- A new recoverable sequence number to be set in the Tpipe for messages received by WebSphere MQ, and to be set as the partner's send sequence number. If you do not specify this, the partner's send sequence number is set to the current WebSphere MQ receive sequence number.

If there is an unresolved unit of recovery associated with the Tpipe, this is also notified in the message. Use the WebSphere MQ command RESET TPIPE to specify whether to commit the unit of recovery, or back it out. If you commit the unit of recovery, the batch of messages has already been sent to IMS, and is deleted from the bridge queue. If you back the unit of recovery out, the messages are returned to the bridge queue, to be subsequently sent to IMS.

Commit mode 1 (send-then-commit) Tpipes are not synchronized.

Considerations for Commit mode 1 transactions

In IMS, commit mode 1 (CM1) transactions send their output replies before syncpoint.

A CM1 transaction might not be able to send its reply, for example because:
- The Tpipe on which the reply is to be sent is stopped
- OTMA is stopped
- The OTMA client (that is, the queue manager) has gone away
- The reply-to queue and dead-letter queue are unavailable

For all the above reasons, the IMS application sending the message pseudo-abends with code U0119. The IMS transaction and program are not stopped in this case.

These reasons often prevent messages being sent into IMS, as well as replies being delivered from IMS. A U0119 abend can occur if:

- The Tpipe, OTMA, or the queue manager is stopped while the message is in IMS
- IMS replies on a different Tpipe to the incoming message, and that Tpipe is stopped
- IMS replies to a different OTMA client, and that client is unavailable.

Whenever a U0119 abend occurs, both the incoming message to IMS and the reply messages to WebSphere MQ are lost. If the output of a CM0 transaction cannot be delivered for any of the above reasons, it is queued on the Tpipe within IMS.

Deleting messages from IMS

A message that is destined for WebSphere MQ through the IMS bridge can be deleted if the Tmember/Tpipe is stopped. To delete one message for the queue manager with XCF member name *tmember*, issue the following IMS command:

```
/DEQUEUE TMEMBER tmember TPIPE tpipe PURGE1
```

To delete all the messages on the Tpipe, issue the following IMS command:

```
/DEQUEUE TMEMBER tmember TPIPE tpipe PURGE
```

Deleting Tpipes

You cannot delete IMS Tpipes yourself. They are deleted by IMS at the following times:

- Synchronized Tpipes are deleted when IMS is cold started.
- Non-synchronized Tpipes are deleted when IMS is restarted.

Chapter 4. Managing WebSphere MQ resources

Managing the logs

This chapter describes the tasks involved in managing the WebSphere MQ logs. It contains these sections:

- "Archiving logs with the ARCHIVE LOG command"
- "Controlling archiving and logging" on page 77
- "Printing log records" on page 78
- "Recovering logs" on page 78
- "Discarding archive log data sets" on page 78

Archiving logs with the ARCHIVE LOG command

An authorized operator can archive the current WebSphere MQ active log data sets whenever required using the ARCHIVE LOG command.

When you issue the ARCHIVE LOG command, WebSphere MQ truncates the current active log data sets, then runs an asynchronous off-load, and updates the BSDS with a record of the off-load.

The ARCHIVE LOG command has a MODE(QUIESCE) option. With this option, WebSphere MQ jobs and users are quiesced after a commit point, and the resulting point of consistency is captured in the current active log before it is off-loaded.

Consider using the MODE(QUIESCE) option when planning a backup strategy for off site recovery. It creates a system-wide point of consistency, which minimizes the number of data inconsistencies when the archive log is used with the most current backup page set copy during recovery. For example:

```
ARCHIVE LOG MODE(QUIESCE)
```

If you issue the ARCHIVE LOG command without specifying a TIME parameter, the quiesce time period defaults to the value of the QUIESCE parameter of the CSQ6ARVP macro. If the time required for the ARCHIVE LOG MODE(QUIESCE) to complete is less than the time specified, the command completes successfully; otherwise, the command fails when the time period expires. You can specify the time period explicitly by using the TIME option, for example:

```
ARCHIVE LOG MODE(QUIESCE) TIME(60)
```

This command specifies a quiesce period of up to 60 seconds before ARCHIVE LOG processing occurs.

Attention: Using the TIME option when time is critical can significantly disrupt WebSphere MQ availability for all jobs and users that use WebSphere MQ resources.

By default, the command is processed asynchronously from the time you submit the command. (To process the command synchronously with other WebSphere MQ commands use the WAIT(YES) option with QUIESCE, but be aware that the z/OS console is locked from WebSphere MQ command input for the entire QUIESCE period.)

During the quiesce period:
- Jobs and users on the queue manager are allowed to go through commit processing, but are suspended if they try to update any WebSphere MQ resource after the commit.
- Jobs and users that only read data can be affected, since they might be waiting for locks held by jobs or users that were suspended.
- New tasks can start, but they cannot update data.

The output from the DISPLAY LOG command uses the message CSQV400I to indicate that a quiesce is in effect. For example:

```
CSQJ322I +CSQ1 DISPLAY LOG report ...
Parameter   Initial value          SET value
-----------  ----------------------  ----------------------
INBUFF      60
OUTBUFF     4000
MAXRTU      2
MAXARCH     2
TWOACTV     YES
TWOARCH     YES
TWOBSDS     YES
OFFLOAD     YES
WRTHRSH     20
DEALLCT     0
End of LOG report
CSQJ370I +CSQ1 LOG status report ...
Copy %Full DSName
  1    68  VICY.CSQ1.LOGCOPY1.DS01
  2    68  VICY.CSQ1.LOGCOPY2.DS01
Restarted at 2005-02-24 09:49:30 using RBA=00000891B000
Latest RBA=00000891CCF8
Offload task is AVAILABLE
Full logs to offload - 0 of 4
CSQV400I +CSQ1 ARCHIVE LOG QUIESCE CURRENTLY ACTIVE
CSQ9022I +CSQ1 CSQJC001 ' DISPLAY LOG' NORMAL COMPLETION
```

When all updates are quiesced, the quiesce history record in the BSDS is updated with the date and time that the active log data sets were truncated, and with the last-written RBA in the current active log data sets. WebSphere MQ truncates the current active log data sets, switches to the next available active log data sets, and issues message CSQJ311E stating that off-load started.

If updates cannot be quiesced before the quiesce period expires, WebSphere MQ issues message CSQJ317I, and ARCHIVE LOG processing terminates. The current active log data sets are not truncated, nor switched to the next available log data sets, and off-load is not started.

Whether the quiesce was successful or not, all suspended users and jobs are then resumed, and WebSphere MQ issues message CSQJ312I, stating that the quiesce is ended and update activity is resumed.

If ARCHIVE LOG is issued when the current active log is the last available active log data set, the command is not processed, and WebSphere MQ issues the following message:

```
CSQJ319I - csect-name CURRENT ACTIVE LOG DATA SET IS THE LAST
           AVAILABLE ACTIVE LOG DATA SET.  ARCHIVE LOG PROCESSING
           WILL BE TERMINATED
```

If ARCHIVE LOG is issued when another ARCHIVE LOG command is already in progress, the new command is not processed, and WebSphere MQ issues the following message:

```
CSQJ318I - ARCHIVE LOG COMMAND ALREADY IN PROGRESS
```

For information about the syntax of the ARCHIVE LOG command, see the WebSphere MQ Script (MQSC) Command Reference manual. For information about the messages issued during archiving, see the WebSphere MQ for z/OS Messages and Codes manual.

Restarting the log archive process after a failure

If there is a problem during the log archive process (for example, a problem with allocation or tape mounts), the archiving of the active log might be suspended. You can cancel the archive process and restart it by using the ARCHIVE LOG CANCEL OFFLOAD command. This command cancels any off-load processing currently in progress, and restarts the archive process. It starts with the oldest log data set that has not been archived, and proceeds through all active log data sets that need off-loading. Any log archive operations that have been suspended are restarted.

Use this command only if you are sure that the current log archive task is no longer functioning, or if you want to restart a previous attempt that failed. This is because the command might cause an abnormal termination of the off-load task, which might result in a dump.

Controlling archiving and logging

Many aspects of archiving and logging are controlled by parameters set using the CSQ6LOGP, CSQ6ARVP and CSQ6SYSP macros of the system parameter module when the queue manager is customized. See the WebSphere MQ for z/OS System Setup Guide for details of these macros.

Some of these parameters can be changed while a queue manager is running using the WebSphere MQ MQSC SET LOG, SET SYSTEM and SET ARCHIVE commands. They are shown in Table 3:

Table 3. Archiving and logging parameters that can be changed while a queue manager is running

SET command	Parameters
LOG	WRTHRSH, MAXARCH, DEALLCT, MAXRTU
ARCHIVE	All
SYSTEM	LOGLOAD

You can display the settings of all the parameters using the MQSC DISPLAY LOG, DISPLAY ARCHIVE and DISPLAY SYSTEM commands. These commands also show status information about archiving and logging.

These commands are described in WebSphere MQ Script (MQSC) Command Reference.

Printing log records

You can extract and print log records using the CSQ1LOGP utility. For instructions, see "The log print utility (CSQ1LOGP)" on page 201.

Recovering logs

Normally, you do not need to back up and restore the WebSphere MQ logs, especially if you are using dual logging. However, in rare circumstances, such as an I/O error on a log, you might need to recover the logs. Use Access Method Services to delete and redefine the data set, and then copy the corresponding dual log into it.

Discarding archive log data sets

You must keep enough log data to be able to perform unit of work recovery, page set media recovery if a page set is lot, or CF structure media recovery if a CF structure is lost. Do not discard archive log data sets that might be required for recovery; if you discard these archive log data sets you might not be able to perform required recovery operations.

If you have confirmed that your archive log data sets can be discarded, you can do this in either of the following ways:
- Automatic archive deletion
- Manual archive deletion

Automatic archive log data set deletion

You can use a DASD or tape management system to delete archive log data sets automatically. The retention period for WebSphere MQ archive log data sets is specified by the retention period field ARCRETN in the CSQ6ARVP installation macro (see the WebSphere MQ for z/OS System Setup Guide for more information). This value is passed to the management system in the JCL parameter RETPD.

The default for the retention period specifies that archive logs are to be kept for 9999 days (the maximum possible). **You can change the retention period but you must ensure that you can accommodate the number of backup cycles that you have planned for**.

WebSphere MQ uses the retention period value as the value for the JCL parameter RETPD when archive log data sets are created.

The retention period set by MVS/DFP's storage management subsystem (SMS) can be overridden by this WebSphere MQ parameter. Typically, the retention period is set to the smaller value specified by either WebSphere MQ or SMS. The storage administrator and WebSphere MQ administrator must agree on a retention period value that is appropriate for WebSphere MQ.

Note: WebSphere MQ does not have an automated method to delete information about archive log data sets from the BSDS, because some tape management systems provide external manual overrides of retention periods. Therefore, information about an archive log data set can still be in the BSDS long after the data-set retention period has expired and the data set has been scratched by the tape management system. Conversely, the maximum number of archive log data sets might have been exceeded and the data from the BSDS might have been dropped before the data set has reached its expiration date.

If archive log data sets are deleted automatically, remember that the operation does not update the list of archive logs in the BSDS. You can update the BSDS with the change log inventory utility, as described in "Changing the BSDS" on page 83. The update is not essential. Recording old archive logs wastes space in the BSDS, but does no other harm.

Manually deleting archive log data sets

You must keep all the log records as far back as the lowest RBA identified in messages CSQI024I and CSQI025I. This RBA is obtained using the DISPLAY USAGE command that you issued when creating a point of recovery using "Method 1: Full backup" on page 95. **Read "Creating a point of recovery for non-shared resources" on page 94 before discarding any logs**.

Locate and discard archive log data sets:

Having established the minimum log RBA required for recovery, you can find archive log data sets that contain only earlier log records by performing the following procedure:

1. Use the print log map utility to print the contents of the BSDS. For an example of the output, see "The print log map utility (CSQJU004)" on page 200.
2. Find the sections of the output titled "ARCHIVE LOG COPY n DATA SETS". If you use dual logging, there are two sections. The columns labeled STARTRBA and ENDRBA show the range of RBAs contained in each volume. Find the volumes whose ranges include the minimum RBA you found with messages CSQI024I and CSQI025I. These are the earliest volumes you need to keep. If you are using dual-logging, there are two such volumes.

    If no volumes have an appropriate range, one of the foloowing cases applies:

    - The minimum RBA has not yet been archived, and you can discard all archive log volumes.

    - The list of archive log volumes in the BSDS wrapped around when the number of volumes exceeded the number allowed by the MAXARCH parameter of the CSQ6LOGP macro. If the BSDS does not register an archive log volume, that volume cannot be used for recovery. Therefore, consider adding information about existing volumes to the BSDS. For instructions, see "Changes for archive logs" on page 84.

        Also consider increasing the value of MAXARCH. For information, see the WebSphere MQ for z/OS System Setup Guide.

3. Delete any archive log data set or volume whose ENDRBA value is less than the STARTRBA value of the earliest volume you want to keep. If you are using dual logging, delete both such copies.

    Because BSDS entries wrap around, the first few entries in the BSDS archive log section might be more recent than the entries at the bottom. Look at the

combination of date and time and compare their ages. Do not assume that you can discard all entries *above* the entry for the archive log containing the minimum LOGRBA.

Delete the data sets. If the archives are on tape, erase the tapes. If they are on DASD, run a z/OS utility to delete each data set. Then, if you want the BSDS to list only existing archive volumes, use the change log inventory utility (CSQJU003) to delete entries for the discarded volumes. See "Changes for archive logs" on page 84 for an example.

The impact of log shunting

When a unit of work is considered to be long, a representation of each log record is written further down the log. This is known as *log shunting*. It is described more fully in WebSphere MQ for z/OS Concepts and Planning Guide.

The queue manager uses these shunted log records instead of the originals after a failure, to ensure unit of work integrity. There are two benefits to this:

- the quantity of log data which must be retained for unit of work coordination is reduced
- less log data must be traversed at queue manager restart time, so the queue manager is restarted more quickly

Shunted log records do not contain sufficient information for media recovery operations.

Data held in the log is used for two distinct purposes; media recovery and unit of work coordination. In the event of a media failure affecting either a CF structure or page set, the queue manager can recover the media to the point of failure by restoring a prior copy and updating this using data contained in the log. Persistent activity performed in a unit of work is recorded on the log so that in the event of a failure, it can either be backed out or locks can be recovered on changed resources. The quantity of log data you need to retain to enable queue manager recovery is impacted by these two elements.

For media recovery, you must retain sufficient log data to be able to perform media recovery from at least the most recent media copy and to be able to back out. (Your site may stipulate the ability to recover from older backups.) For unit of work integrity, you must retain the log data for your oldest inflight or indoubt units of work.

To assist you with managing the system, the queue manager detects old units of work at each log archive and reports them in messages CSQJ160 and CSQJ161. An internal task reads unit of work log information for these old units of work and rewrites it in a more succinct form to the current position in the log. Message CSQR026 indicates when this has happened. The MQSC command DISPLAY USAGE TYPE(DATASET) will also assist you to manage the retention of log data. The command reports 3 pieces of recovery information which relate to the description above:

1. how much of the log must be retained for unit of work recovery
2. how much of the log must be retained for media recovery of page sets
3. for a queue manager in a queue-sharing group, how much of the log must be retained for media recovery of CF structures

For each of these, an attempt is made to map the oldest log data required into a data set. As new units of work start and stop, we would expect (1) above to move to a more recent position in the log. If it is not moving, the long running UOW

messages will warn you that there is an issue. (2) relates to page set media recovery if the queue manager were to be shut down now and restarted. It does not know about when you last backed up your page sets, or which backup you might have to use if there was a page set failure. It will normally move to a more recent position in the log during checkpoint processing as changes held in the buffer pools are written to the page sets. In (3), the queue manager does know about CF structure backups taken either on this queue manager or on other queue managers in the queue sharing group. However, CF structure recovery requires a merge of log data from all queue managers in the queue-sharing group which have interacted with the CF structure since the last backup. This means that the log data is identified by a log record sequence number, (or LRSN), which is timestamp based and so applicable across the entire queue-sharing group rather than an RBA which would be different on different queue managers in the queue-sharing group. It will normally move to a more recent position in the log as BACKUP CFSTRUCT commands are performed on either this or other queue managers in the queue-sharing group.

Managing the BSDS

This chapter describes the tasks involved in managing the bootstrap data set. It contains these sections:

- "Finding out what the BSDS contains"
- "Changing the BSDS" on page 83
- "Recovering the BSDS" on page 85

Finding out what the BSDS contains

The print log map utility (CSQJU004) is a batch utility that lists the information stored in the BSDS. For instructions on running it, see "The print log map utility (CSQJU004)" on page 200.

Time stamps in the BSDS

The output of the print log map utility shows the time stamps, which are used to record the date and time of various system events, that are stored in the BSDS.

The following time stamps are included in the header section of the report:

SYSTEM TIMESTAMP

Reflects the date and time the BSDS was last updated. The BSDS time stamp can be updated when:

- The queue manager starts.
- The write threshold is reached during log write activities. Depending on the number of output buffers you have specified and the system activity rate, the BSDS might be updated several times a second, or might not be updated for several seconds, minutes, or even hours. For details of the write threshold, see the WRTHRSH parameter of the CSQ6LOGP macro in the WebSphere MQ for z/OS System Setup Guide.
- WebSphere MQ drops into a single BSDS mode from its normal dual BSDS mode due to an error. This can occur when a request to get, insert, point to, update, or delete a BSDS record is unsuccessful. When this error occurs, WebSphere MQ updates the time stamp in the remaining BSDS to force a time stamp mismatch with the disabled BSDS.

UTILITY TIMESTAMP
> The date and time the contents of the BSDS were altered by the change log inventory utility (CSQJU003).

The following time stamps are included in the active and archive log data sets portion of the report:

Active log date
> The date the active log entry was created in the BSDS, that is, when the CSQJU003 NEWLOG was done.

Active log time
> The time the active log entry was created in the BSDS, that is, when the CSQJU003 NEWLOG was done.

Archive log date
> The date the archive log entry was created in the BSDS, that is, when the CSQJU003 NEWLOG was done or the archive itself was done.

Archive log time
> The time the archive log entry was created in the BSDS, that is, when the CSQJU003 NEWLOG was done or the archive itself was done.

Active log data set status

The BSDS records the status of an active log data set as one of the following:

NEW The data set has been defined but never used by WebSphere MQ, or the log was truncated to a point before the data set was first used. In either case, the data set starting and ending RBA values are reset to zero.

REUSABLE
> Either the data set has been defined but never used by WebSphere MQ, or the data set has been off-loaded. In the print log map output, the start RBA value for the last REUSABLE data set is equal to the start RBA value of the last archive log data set.

NOT REUSABLE
> The data set contains records that have not been off-loaded.

STOPPED
> The off-load processor encountered an error while reading a record, and that record could not be obtained from the other copy of the active log.

TRUNCATED
> Either:
>
> • An I/O error occurred, and WebSphere MQ has stopped writing to this data set. The active log data set is off-loaded, beginning with the starting RBA and continuing up to the last valid record segment in the truncated active log data set. The RBA of the last valid record segment is lower than the ending RBA of the active log data set. Logging is switched to the next available active log data set, and continues uninterrupted.
>
> or
>
> • An ARCHIVE LOG function has been called, which has truncated the active log.

The status appears in the output from the print log map utility.

Changing the BSDS

You do not have to take special steps to keep the BSDS updated with records of logging events because WebSphere MQ does that automatically. However, you might want to change the BSDS if you do any of the following:

- Add more active log data sets.
- Copy active log data sets to newly allocated data sets, for example, when providing larger active log allocations.
- Move log data sets to other devices.
- Recover a damaged BSDS.
- Discard outdated archive log data sets.

You can change the BSDS by running the change log inventory utility (CSQJU003). Only run this utility when the queue manager is inactive, or you might get inconsistent results. The action of the utility is controlled by statements in the SYSIN data set. This section shows several examples. For complete instructions, see "The change log inventory utility (CSQJU003)" on page 191.

You can copy an active log data set only when the queue manager is inactive because WebSphere MQ allocates the active log data sets as exclusive (DISP=OLD) at queue manager startup.

Changes for active logs

You can add to, delete from, and record entries in the BSDS for active logs using the change log utility. Examples only are shown here; replace the data set names shown with the ones you want to use. For more details of the utility, see "The change log inventory utility (CSQJU003)" on page 191.

Adding record entries to the BSDS:

If an active log has been flagged as "stopped", it is not reused for logging; however, it continues to be used for reading. Use the access method services to define new active log data sets, then use the change log inventory utility to register the new data sets in the BSDS. For example, use:

```
NEWLOG DSNAME=MQM111.LOGCOPY1.DS10,COPY1
NEWLOG DSNAME=MQM111.LOGCOPY2.DS10,COPY2
```

If you are copying the contents of an old active log data set to the new one, you can also give the RBA range and the starting and ending time stamps on the NEWLOG function.

Deleting information about the active log data set from the BSDS:

To delete information about an active log data set from the BSDS, you could use:

```
DELETE DSNAME=MQM111.LOGCOPY1.DS99
DELETE DSNAME=MQM111.LOGCOPY2.DS99
```

Recording information about the log data set in the BSDS:

To record information about an existing active log data set in the BSDS, use:

```
NEWLOG DSNAME=MQM111.LOGCOPY1.DS10,COPY2,STARTIME=19930212205198,
           ENDTIME=19930412205200,STARTRBA=6400,ENDRBA=94FF
```

You might need to insert a record containing this type of information in the BSDS because:

- The entry for the data set has been deleted, but is needed again.
- You are copying the contents of one active log data set to another data set.
- You are recovering the BSDS from a backup copy.

Enlarging the active log:

This procedure must only be used when the queue manager is inactive:

1. Stop the queue manager. This step is required because WebSphere MQ allocates all active log data sets for its exclusive use when it is active.
2. Use Access Method Services ALTER with the NEWNAME option to rename your active log data sets.
3. Use Access Method Services DEFINE to define larger active log data sets.

    By reusing the old data set names, you do not have to run the change log inventory utility to establish new names in the BSDSs. The old data set names and the correct RBA ranges are already in the BSDSs.

4. Use Access Method Services REPRO to copy the old (renamed) data sets into their respective new data sets.
5. Start the queue manager.

If all your log data sets are the same size, your system will be operationally more consistent and efficient. If the log data sets are not the same size, it is more difficult to track your system's logs, and so space can be wasted.

Changes for archive logs

You can add to, delete from, and change the password of, entries in the BSDS for archive logs. Examples only are shown here; replace the data set names shown with the ones you want to use. For more details of the utility, see "The change log inventory utility (CSQJU003)" on page 191.

Adding an archive log:

When the recovery of an object depends on reading an existing archive log data set, the BSDS must contain information about that data set so that WebSphere MQ can find it. To register information about an existing archive log data set in the BSDS, use:

```
NEWLOG DSNAME=CSQARC1.ARCHLOG1.E00021.T2205197.A0000015,COPY1VOL=CSQV04,
           UNIT=TAPE,STARTRBA=3A190000,ENDRBA=3A1F0FFF,CATALOG=NO
```

Deleting an archive log:

To delete an entire archive log data set on one or more volumes, use:

```
DELETE DSNAME=CSQARC1.ARCHLOG1.E00021.T2205197.A0000015,COPY1VOL=CSQV04
```

Changing the password of an archive log:

If you change the password of an existing archive log data set, you must also change the information in the BSDS.

1. List the BSDS, using the print log map utility.
2. Delete the entry for the archive log data set with the changed password, using the DELETE function of the CSQJU003 utility (see topic "The change log inventory utility (CSQJU003)" on page 191).
3. Name the same data set as a new archive log data set. Use the NEWLOG function of the CSQJU003 utility (see topic "The change log inventory utility (CSQJU003)" on page 191), and give the new password, the starting and ending RBAs, and the volume serial numbers (which can be found in the print log map utility output, see page "The print log map utility (CSQJU004)" on page 200).

To change the password for new archive log data sets, use:

```
ARCHIVE PASSWORD=password
```

To stop placing passwords on new archive log data sets, use:

```
ARCHIVE NOPASSWD
```

Note: Only use the ARCHIVE utility function if you do not have an external security manager.

Recovering the BSDS

If WebSphere MQ is operating in dual BSDS mode and one BSDS becomes damaged, forcing WebSphere MQ into single BSDS mode, WebSphere MQ continues to operate without a problem (until the next restart). To return the environment to dual BSDS mode:

1. Use Access Method Services to rename or delete the damaged BSDS and to define a new BSDS with the same name as the damaged BSDS. Example control statements can be found in job CSQ4BREC in thlqual.SCSQPROC.
2. Issue the WebSphere MQ command RECOVER BSDS to make a copy of the valid BSDS in the newly allocated data set and to reinstate dual BSDS mode.

If WebSphere MQ is operating in single BSDS mode and the BSDS is damaged, or if WebSphere MQ is operating in dual BSDS mode and both BSDSs are damaged, the queue manager stops and does not restart until the BSDS data sets are repaired. In this case:

1. Locate the BSDS associated with the most recent archive log data set. The data set name of the most recent archive log appears on the job log in the last occurrence of message CSQJ003I, which indicates that off-loading has been completed successfully. In preparation for the rest of this procedure, it is a good practice to keep a log of all successful archives noted by that message:

- If archive logs are on DASD, the BSDS is allocated on any available DASD. The BSDS name is like the corresponding archive log data set name; change only the first letter of the last qualifier, from A to B, as in the example below:

  Archive log name
  > CSQ.ARCHLOG1.*A0000001*

  BSDS copy name
  > CSQ.ARCHLOG1.*B0000001*

- If archive logs are on tape, the BSDS is the first data set of the first archive log volume. The BSDS is not repeated on later volumes.

2. If the most recent archive log data set has no copy of the BSDS (for example, because an error occurred when off-loading it), locate an earlier copy of the BSDS from an earlier off-load.

3. Rename *damaged* BSDSs using the Access Method Services ALTER command with the NEWNAME option. If you want to delete a damaged BSDS, use the Access Method Services DELETE command. For each damaged BSDS, use Access Method Services to define a new BSDS as a replacement data set. Job CSQ4BREC in thlqual.SCSQPROC contains Access Method Services control statements to define a new BSDS.

4. Use the Access Method Services REPRO command to copy the BSDS from the archive log to one of the replacement BSDSs you defined in step 3. Do not copy any data to the second replacement BSDS, you do that in step 5 on page 87.

   a. Print the contents of the replacement BSDS.

   Use the print log map utility (CSQJU004) to print the contents of the replacement BSDS. This enables you to review the contents of the replacement BSDS before continuing your recovery work.

   b. Update the archive log data set inventory in the replacement BSDS.

   Examine the output from the print log map utility and check that the replacement BSDS does not contain a record of the archive log from which the BSDS was copied. If the replacement BSDS is an old copy, its inventory might not contain all archive log data sets that were created more recently. The BSDS inventory of the archive log data sets must be updated to reflect the current subsystem inventory.

   Use the change log inventory utility (CSQJU003) NEWLOG statement to update the replacement BSDS, adding a record of the archive log from which the BSDS was copied. If the archive log data set is password-protected, use the PASSWORD option of the NEWLOG function. Also, if the archive log data set is cataloged, ensure that the CATALOG option of the NEWLOG function is properly set to CATALOG=YES. Use the NEWLOG statement to add any additional archive log data sets that were created later than the BSDS copy.

   c. Update passwords in the replacement BSDS.

   The BSDS contains passwords for the archive log data sets and for the active log data sets. To ensure that the passwords in the replacement BSDS reflect the current passwords used by your installation, use the change log inventory ARCHIVE utility function with the PASSWORD option.

   d. Update the active log data set inventory in the replacement BSDS.

   In unusual circumstances, your installation might have added, deleted, or renamed active log data sets since the BSDS was copied. In this case, the replacement BSDS does not reflect the actual number or names of the active log data sets your installation currently has in use.

If you need to delete an active log data set from the replacement BSDS log inventory, use the change log inventory utility DELETE function.

If you need to add an active log data set to the replacement BSDS log inventory, use the change log inventory utility NEWLOG function. Ensure that the RBA range is specified correctly on the NEWLOG function. If the active log data set is password-protected, use the PASSWORD option.

If you need to rename an active log data set in the replacement BSDS log inventory, use the change log inventory utility DELETE function, followed by the NEWLOG function. Ensure that the RBA range is specified correctly on the NEWLOG function. If the active log data set is password-protected, use the PASSWORD option.

e. Update the active log RBA ranges in the replacement BSDS.

Later, when the queue manager restarts, it compares the RBAs of the active log data sets listed in the BSDS with the RBAs found in the actual active log data sets. If the RBAs do not agree, the queue manager does not restart. The problem is magnified when a particularly old copy of the BSDS is used. To solve this problem, use the change log inventory utility (CSQJU003) to adjust the RBAs found in the BSDS using the RBAs in the actual active log data sets. You do this by:

- Using the print log records utility (CSQ1LOGP) to print a summary report of the active log data set. This shows the starting and ending RBAs.

- Comparing the actual RBA ranges with the RBA ranges you have just printed, when the RBAs of all active log data sets are known.

  If the RBA ranges are equal for all active log data sets, you can proceed to the next recovery step without any additional work.

  If the RBA ranges are not equal, adjust the values in the BSDS to reflect the actual values. For each active log data set that needs to have the RBA range adjusted, use the change log inventory utility DELETE function to delete the active log data set from the inventory in the replacement BSDS. Then use the NEWLOG function to redefine the active log data set to the BSDS. If the active log data sets are password-protected, use the PASSWORD option of the NEWLOG function.

f. If only two active log data sets are specified for each copy of the active log, WebSphere MQ can have difficulty during queue manager restart. The problem can arise when one of the active log data sets is full and has not been off-loaded, while the second active log data set is close to filling. In this case, add a new active log data set for each copy of the active log and define each new active log data set in the replacement BSDS log inventory.

Use the Access Method Services DEFINE command to define a new active log data set for each copy of the active log and use the change log inventory utility NEWLOG function to define the new active log data sets in the replacement BSDS. You do not need to specify the RBA ranges on the NEWLOG statement. However, if the active log data sets are password-protected, use the PASSWORD option of the NEWLOG function. Example control statements to accomplish this task can be found in job CSQ4LREC in thlqual.SCSQPROC.

5. Copy the updated BSDS to the second new BSDS data set. The BSDSs are now identical.

Use the print log map utility (CSQJU004) to print the contents of the second replacement BSDS at this point.

6. See "Active log problems" on page 133 for information about what to do if you have lost your current active log data set.

7. Restart the queue manager using the newly constructed BSDS. WebSphere MQ determines the current RBA and what active logs need to be archived.

Managing page sets

This chapter describes how to add, copy, and generally manage the page sets associated with a queue manager. It contains these sections:

- "How to add a page set to a queue manager"
- "What to do when one of your page sets becomes full" on page 89
- "How to balance loads on page sets" on page 89
- "How to expand a page set" on page 91
- "How to reduce a page set" on page 93
- "How to reintroduce a page set" on page 93
- "How to back up and recover page sets" on page 94
- "How to delete page sets" on page 98
- "How to back up and restore queues using CSQUTIL" on page 99

See the WebSphere MQ for z/OS Concepts and Planning Guide for a description of page sets, storage classes, buffers, and buffer pools, and some of the performance considerations that apply.

How to add a page set to a queue manager

This description assumes that you have a queue manager that is already running. You might need to add a page set if, for example, your queue manager has to cope with new applications using new queues.

To add a new page set, use the following procedure:

1. Define and format the new page set. You can use the sample JCL in thlqual.SCSQPROC(CSQ4PAGE) as a basis. For more information, see "Formatting page sets (FORMAT)" on page 162.

   Take care not to format any page sets that are in use, unless this is what you intend. If so, use the FORCE option of the FORMAT utility function.

2. Use the DEFINE PSID command with the DSN option to associate the page set with a buffer pool.

3. Add the appropriate storage class definitions for your page set by issuing DEFINE STGCLASS commands.

4. To ensure the page set remains available when you restart your queue manager:

   a. Add the new page set to the started task procedure for your queue manager.

   b. Add a definition for the new page set to your CSQINP1 initialization data set.

For details of the DEFINE PSID and DEFINE STGCLASS commands, see WebSphere MQ Script (MQSC) Command Reference

What to do when one of your page sets becomes full

You can find out about the utilization of page sets by using the WebSphere MQ command DISPLAY USAGE. For example, the command:

```
DISPLAY USAGE PSID(03)
```

displays the current state of the page set 03. This tells you how many free pages this page set has.

If you have defined secondary extents for your page sets, they are dynamically expanded each time they fill up. Eventually, all secondary extents are used, or no further disk space is available. If this happens, an application receives the return code MQRC_STORAGE_MEDIUM_FULL.

If an application receives a return code of MQRC_STORAGE_MEDIUM_FULL from an MQI call, this is a clear indication that there is not enough space left on the page set. If the problem persists or is likely to reoccur, you must do something to solve it.

You can approach this problem in a number of ways:
- Balance the load between page sets by moving queues from one page set to another.
- Expand the page set. See "How to expand a page set" on page 91 for instructions.
- Redefine the page set so that it can expand beyond 4GB to a maximum size of 64GB. See "Defining a page set to be larger than 4GB" on page 91 for instructions.

How to balance loads on page sets

Load balancing on page sets means moving the messages associated with one or more queues from one page set to another, less used, page set. Use this technique if it is not practical to expand the page set.

To identify which queues are using a page set, use the appropriate WebSphere MQ commands. For example, to find out which queues are mapped to page set 02, first, find out which storage classes map to page set 02, by using the command:

```
DISPLAY STGCLASS(*) PSID(02)
```

Then use the following command to find out which queues use which storage class:

```
DISPLAY QUEUE(*) TYPE(QLOCAL) STGCLASS
```

Moving a non-shared queue

To move queues and their messages from one page set to another, use the MQSC MOVE QLOCAL command (described in WebSphere MQ Script (MQSC)

Command Reference). When you have identified the queue or queues that you want to move to a new page set, follow this procedure for each of these queues:

1. Ensure that the queue you want to move is not in use by any applications (that is, IPPROCS and OPPROCS values from the DISPLAY QSTATUS command are zero) and that it has no uncommitted messages (the UNCOM value from the DISPLAY QSTATUS command is NO).

   Note: The only way to ensure that this state continues is to change your security settings temporarily. If you cannot do this, later stages in this procedure might fail if applications start to use the queue despite precautionary steps such as setting PUT(DISABLED). However, messages can never be lost by this procedure.

2. Prevent applications from putting messages on the queue being moved by altering the queue definition to disable **MQPUT**s. Change the queue definition to PUT(DISABLED).

3. Define a temporary queue with the same attributes as the queue that is being moved, using the command:

```
DEFINE QL(TEMP_QUEUE) LIKE(QUEUE_TO_MOVE) PUT(ENABLED) GET(ENABLED)
```

   Note: If this temporary queue already exists from a previous run, delete it before doing the define.

4. Move the messages to the temporary queue using the following command:

```
MOVE QLOCAL(QUEUE_TO_MOVE) TOQLOCAL(TEMP_QUEUE)
```

5. Delete the queue you are moving, using the command:

```
DELETE QLOCAL(QUEUE_TO_MOVE)
```

6. Define a new storage class that maps to the required page set, for example:

```
DEFINE STGCLASS(NEW) PSID(nn)
```

   Add the new storage class definition to the CSQINP2 data sets ready for the next queue manager restart.

7. Redefine the queue that you are moving, by changing the storage class attribute:

```
DEFINE QL(QUEUE_TO_MOVE) LIKE(TEMP_QUEUE) STGCLASS(NEW)
```

   When the queue is redefined, it is based on the temporary queue created in step 3.

8. Move the messages back to the new queue, using the command:

```
MOVE QLOCAL(TEMP) TOQLOCAL(QUEUE_TO_MOVE)
```

9. The queue created in step 3 on page 90 is no longer required. Use the following command to delete it:

```
DELETE QL(TEMP_QUEUE)
```

10. If the queue being moved was defined in the CSQINP2 data sets, change the STGCLASS attribute of the appropriate DEFINE QLOCAL command in the CSQINP2 data sets. Add the REPLACE keyword so that the existing queue definition is replaced.

Figure 41 shows an extract from a load balancing job.

```
//UTILITY  EXEC  PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB  DD    DSN=thlqual.SCSQANLE,DISP=SHR
//         DD    DSN=thlqual.SCSQAUTH,DISP=SHR
//SYSPRINT DD    SYSOUT=*
//SYSIN    DD    *
COMMAND DDNAME(MOVEQ)
/*
//MOVEQ  DD    *
ALTER QL(QUEUE_TO_MOVE) PUT(DISABLED)
DELETE QL(TEMP_QUEUE) PURGE
DEFINE QL(TEMP_QUEUE) LIKE(QUEUE_TO_MOVE) PUT(ENABLED) GET(ENABLED)
MOVE QLOCAL(QUEUE_TO_MOVE) TOQLOCAL(TEMP_QUEUE)
DELETE QL(QUEUE_TO_MOVE)
DEFINE STGCLASS(NEW) PSID(2)
DEFINE QL(QUEUE_TO_MOVE) LIKE(TEMP_QUEUE) STGCLASS(NEW)
MOVE QLOCAL(TEMP_QUEUE) TOQLOCAL(QUEUE_TO_MOVE)
DELETE QL(TEMP_QUEUE)
/*
```

Figure 41. Extract from a load balancing job for a page set

How to expand a page set

A page set can be defined to be automatically expanded as it becomes full by specifying EXPAND(SYSTEM) or EXPAND(USER). If your page set was defined with EXPAND(NONE), you can expand it in either of two ways:

- Alter its definition to allow automatic expansion. See "Altering a page set to allow automatic expansion" on page 92.
- Create a new, larger page set and copy the messages from the old page set to the new one. See "Moving messages to a new, larger page set" on page 92.

Defining a page set to be larger than 4GB

WebSphere MQ can use a page set up to 64GB in size, provided the data set is defined with 'extended addressability' to VSAM. Extended addressability is an attribute which is conferred by an SMS data class. In the examples below, the management class 'EXTENDED' is defined to SMS with 'Extended addressability'. If your existing page set is not currently defined as having extended addressability, use the following method to migrate to an extended addressability format data set.

1. Stop the queue manager.

2. Use Access Method Services to rename the existing page set.
3. Define a destination page set, the same size as the existing page set, but with DATACLASS(EXTENDED).
4. Use Access Method Services REPRO to copy the contents of the existing page set to the destination page set.
5. Restart the queue manager.
6. Alter the page set to use system expansion, to allow it to continue growing beyond its current allocation.

The following JCL shows example Access Method Services commands:

```
//S1       EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
  ALTER 'VICY.CSQ1.PAGE01' -
        NEWNAME('VICY.CSQ1.PAGE01.OLD')
  ALTER 'VICY.CSQ1.PAGE01.DATA' -
        NEWNAME('VICY.CSQ1.PAGE01.DATA.OLD')
  DEFINE CLUSTER (NAME('VICY.CSQ1.PAGE01') -
        MODEL('VICY.CSQ1.PAGE01.OLD') -
        DATACLASS(EXTENDED))
  REPRO  INDATASET('VICY.CSQ1.PAGE01.OLD') -
        OUTDATASET('VICY.CSQ1.PAGE01')
/*
```

Altering a page set to allow automatic expansion

Use the ALTER PSID command with the EXPAND(USER) or EXPAND(SYSTEM) options. See WebSphere MQ Script (MQSC) Command Reference for details of this command and its parameters and WebSphere MQ for z/OS Concepts and Planning Guide for general information on expanding page sets.

Moving messages to a new, larger page set

Note

This technique involves stopping and restarting the queue manager. This deletes any nonpersistent messages that are not on shared queues at restart time. If you have nonpersistent messages that you do not want to be deleted, use load balancing instead (see "How to balance loads on page sets" on page 89).

In this description, the page set that you want to expand is referred to as the *source* page set; the new, larger page set is referred to as the *destination* page set.

Follow these steps:
1. Stop the queue manager.
2. Define the destination page set, ensuring that it is larger than the source page set, with a larger secondary extent value.
3. Use the FORMAT function of CSQUTIL to format the destination page set. See "Formatting page sets (FORMAT)" on page 162 for more details.
4. Use the COPYPAGE function of CSQUTIL to copy all the messages from the source page set to the destination page set. See "Expanding a page set (COPYPAGE)" on page 167 for more details.
5. Restart the queue manager using the destination page set by doing one of the following:

- Change the queue manager started task procedure to reference the destination page set. See the WebSphere MQ for z/OS System Setup Guide for more details.
- Use Access Method Services to delete the source page set and then rename the destination page set, giving it the same name as that of the source page set.

Attention:

Before you delete any WebSphere MQ page set, be sure that you have made the required backup copies.

How to reduce a page set

If you have a large page set that is mostly empty (as shown by the DISPLAY USAGE command), you might want to reduce its size. The procedure to do this involves using the COPY, FORMAT, and LOAD functions of CSQUTIL (see "WebSphere MQ utility program (CSQUTIL)" on page 159). This procedure does not work for page set zero, as it is not practical to reduce the size of this page set; the only way to do so is by reinitializing your queue manager (see "Reinitializing a queue manager" on page 116). The initial steps of the procedure are to try and remove all users from the system so that all UOWs are complete and the page sets are consistent.

1. Prevent all users, other than the WebSphere MQ administrator, from using the queue manager. This can be done by changing the access security settings, for example.
2. Use the STOP QMGR command with the QUIESCE or FORCE attribute to stop the queue manager.
3. Run the SCOPY function of CSQUTIL with the PSID option, to copy all message data from the large page set and save them in a sequential data set.
4. Define a new smaller page set data set to replace the large page set. Run the FORMAT TYPE(NEW) function of CSQUTIL against the page set that you want to reduce.
5. Restart the queue manager using the page sets created in step 4, the other existing page sets, BSDS, and log data sets.
6. Run the LOAD function of CSQUTIL to load back all the messages saved during step 3.
7. Allow all users access to the queue manager.
8. Delete the old large page set.

How to reintroduce a page set

In certain scenarios it is useful to be able to bring an old page set online again to the queue manager. Unless specific action is taken, when the old page set is brought online the queue manager will recognize that the page set recovery RBA stored in the page set itself and in the checkpoint records is old, and will therefore automatically start media recovery of the page set to bring it up to date.

Such media recovery can only be performed at queue manager restart, and is likely to take a considerable length of time, especially if archive logs held on tape must be read. However, normally in this circumstance, the page set has been offline for the intervening period and so the log contains no information pertinent to the page set recovery.

The following three choices are available:

Allow full media recovery to be performed.
1. Stop the queue manager.
2. Ensure definitions are available for the page set in both the started task procedure for the queue manager and in the CSQINP1 initialization data set.
3. Restart the queue manager.

Allow any messages on the page set to be destroyed.
This choice is useful where a page set has been offline for a long time (some months, for example) and it has now been decided to reuse it for a different purpose.
1. Format the page set using the FORMAT function of CSQUTIL with the TYPE(NEW) option.
2. Add definitions for the page set to both the started task procedure for the queue manager and the CSQINP1 initialization data set.
3. Restart the queue manager.

Using the TYPE(NEW) option for formatting clears the current contents of the page set and tells the queue manager to ignore any historical information in the checkpoint about the page set.

Bring the page set online avoiding the media recovery process.
Use this technique only if you are sure that the page set has been offline since a clean shutdown of the queue manager. This choice is most appropriate where the page set has been offline for a short period, typically due to operational issues such as a backup running while the queue manager is being started.
1. Format the page set using the FORMAT function of CSQUTIL with the TYPE(REPLACE) option.
2. Either add the page set back into the queue manager dynamically using the DEFINE PSID command with the DSN option or allow it to be added at a queue manager restart.

Using the TYPE(REPLACE) option for formatting checks that the page set was cleanly closed by the queue manager, and marks it so that media recovery will not be performed. No other changes are made to the contents of the page set.

How to back up and recover page sets

This section describes:
- "Creating a point of recovery for non-shared resources"
- "Recovering page sets" on page 97

For information on how to create a point of recovery for shared resources, see "Recovering shared queues" on page 102.

Creating a point of recovery for non-shared resources

WebSphere MQ can recover objects and non-shared persistent messages to their current state if both:
1. Copies of page sets from an earlier point exist.
2. All the WebSphere MQ logs are available to perform recovery from that point.

These represent a point of recovery for non-shared resources.

Both objects and messages are held on page sets. Multiple objects and messages from different queues can exist on the same page set. For recovery purposes, objects and messages cannot be backed up in isolation, so a page set must be backed up as a whole to ensure the proper recovery of the data.

The WebSphere MQ recovery log contains a record of all persistent messages and changes made to objects. If WebSphere MQ fails (for example, due to an I/O error on a page set), you can recover the page set by restoring the backup copy and restarting the queue manager. WebSphere MQ applies the log changes to the page set from the point of the backup copy.

There are two ways of creating a point of recovery:

Full backup
> Stop the queue manager, thereby forcing all updates on to the page sets.
>
> This allows you to restart from the point of recovery, using only the backed up page set data sets and the logs from that point on.

Fuzzy backup
> Take *fuzzy* backup copies of the page sets without stopping the queue manager.
>
> If you use this method, and your associated logs subsequently become damaged or lost, you cannot to use the fuzzy page set backup copies to recover. This is because the fuzzy page set backup copies contain an inconsistent view of the state of the queue manager and are dependent on the logs being available. If the logs are not available, you need to return to the last set of backup page set copies taken while the subsystem was inactive (Method 1 below) and accept the loss of data from that time.

Method 1: Full backup:

This method involves shutting the queue manager down. This forces all updates on to the page sets so that the page sets are in a consistent state.

1. Stop all the WebSphere MQ applications that are using the queue manager (allowing them to complete first). This can be done by changing the access security or queue settings, for example.

2. When all activity has completed, display and resolve any in-doubt units of recovery. (Use the commands DISPLAY CONN and RESOLVE INDOUBT, as described in the WebSphere MQ Script (MQSC) Command Reference manual.)

   This brings the page sets to a consistent state; if you do not do this, your page sets might not be consistent, and you are effectively doing a fuzzy backup.

3. Issue the ARCHIVE LOG command to ensure that the latest log data is written out to the log data sets.

4. Issue the STOP QMGR MODE(QUIESCE) command. Record the lowest RBA value in the CSQI024I or CSQI025I messages (see WebSphere MQ for z/OS Messages and Codes for more information). You should keep the log data sets starting from the one indicated by the RBA value up to the current log data set.

5. Take backup copies of all the queue manager page sets (see "Backing up page sets" on page 96).

Method 2: Fuzzy backup:

This method does not involve shutting the queue manager down. Therefore, updates might be in virtual storage buffers during the backup process. This means that the page sets are not in a consistent state, and can only be used for recovery in conjunction with the logs.

1. Issue the DISPLAY USAGE TYPE(ALL) command, and record the RBA value in the CSQI024I or CSQI025I messages (see WebSphere MQ for z/OS Messages and Codes for more information).
2. Take backup copies of the page sets (see "Backing up page sets").
3. Issue the ARCHIVE LOG command, to ensure that the latest log data is written out to the log data sets. To restart from the point of recovery, you must keep the log data sets starting from the log data set indicated by the RBA value up to the current log data set.

Backing up page sets:

You can take a backup of your page sets in two ways:
- "Using Access Method Services"
- "Using volume dump and restore" on page 97.

To recover a page set, WebSphere MQ needs to know how far back in the log to go. WebSphere MQ maintains a log RBA number in page zero of each page set, called the *recovery log sequence number* (LSN). This number is the starting RBA in the log from which WebSphere MQ can recover the page set. When you back up a page set, this number is also copied.

If the copy is later used to recover the page set, WebSphere MQ must have access to all the log records from this RBA value to the current RBA. That means you must keep enough of the log records to enable WebSphere MQ to recover from the oldest backup copy of a page set you intend to keep.

Using Access Method Services:

You can use Access Method Services REPRO function (or any equivalent) to make copies of your page sets. You can do this whether or not the queue manager is running. If you want to do it while the queue manager is running, you must DEFINE the page sets with SHAREOPTIONS(2,3).

If you copy the page set while the queue manager is running you *must* use a copy utility that copies page zero of the page set first – if you do not do this you could corrupt the data in your page set. Access Method Services REPRO meets this requirement.

If the process of dynamically expanding a page set is interrupted, for example by power to the system being lost, you can still use Access Method Services REPRO to take a backup of a page set.

If you perform an Access Method Services IDCAMS `LISTCAT ENT('page set data set name') ALLOC`, you will see that the `HI-ALLOC-RBA` is higher than the `HI-USED-RBA`. Access Method Services REPRO just copies the page set up to the high used RBA, and does not give an error.

The next time this page set fills up it is extended again, if possible, and the pages between the high used RBA and the highest allocated RBA are used, along with another new extent.

For more information on the REPRO statement, see the *DFSMS/MVS Access Method Services for VSAM* or the *DFSMS/MVS Access Method Services for the Integrated Catalog Facility* manual.

Using volume dump and restore:

Volume dump services dumps all the data sets on the volume. Likewise, restore volume services can (depending on the type of service) restore all the data sets.

Backing up your object definitions:

You should also back up copies of your object definitions. To do this, use the MAKEDEF feature of the CSQUTIL COMMAND function (described in "Issuing commands to WebSphere MQ (COMMAND)" on page 171).

Back up your object definitions whenever you take a backup copy of your queue manager, and keep the most current version.

Recovering page sets

If the queue manager has terminated due to a failure, the queue manager can normally be restarted with all recovery being performed during restart. However, such recovery is not possible if any of your page sets or log data sets are not available. The extent to which you can now recover depends on the availability of backup copies of page sets and log data sets.

To restart from a point of recovery you must have:
- A backup copy of the page set that is to be recovered.
- If you used the "fuzzy" backup process described in "Method 2: Fuzzy backup" on page 95, the log data set that included the recorded RBA value, the log data set that was made by the ARCHIVE LOG command, and all the log data sets between these.
- If you used full backup, but you do not have the log data sets following that made by the ARCHIVE LOG command, run the FORMAT TYPE(REPLACE) function of the CSQUTIL utility against all you page sets.

To recover a page set to its current state, you must also have all the log data sets and records since the ARCHIVE LOG command.

There are two methods for recovering a page set. To use either method, the queue manager must be stopped.

Simple recovery:

This is the simpler method, and is appropriate for most recovery situations.
1. Delete and redefine the page set you want to restore from backup.
2. Use Access Method Services REPRO to copy the backup copy of the page set into the new page set. Define your new page set with a secondary extent value so that it can be expanded dynamically.

   Alternatively, you can rename your backup copy to the original name, or change the CSQP00xx DD statement in your queue manager procedure to point to your backup page set. However, if you then lose or corrupt the page set, you will no longer have a backup copy to restore from.
3. Restart the queue manager.

4. When the queue manager has restarted successfully, you can restart your applications
5. Reinstate your normal backup procedures for the restored page.

Advanced recovery:

This method provides performance advantages if you have a large page set to recover, or if there has been a lot of activity on the page set since the last backup copy was taken. However, it requires more manual intervention than the simple method, which might increase the risk of error and the time taken to perform the recovery.

1. Delete and redefine the page set you want to restore from backup.
2. Use Access Method Services REPRO to copy the backup copy of the page set into the new page set. Define your new page set with a secondary extent value so that it can be expanded dynamically.

   Alternatively, you can rename your backup copy to the original name, or change the CSQP00xx DD statement in your queue manager procedure to point to your backup page set. However, if you then lose or corrupt the page set, you will no longer have a backup copy to restore from.
3. Change the CSQINP1 definitions for your queue manager to make the buffer pool associated with the page set being recovered as large as possible. By making the buffer pool this large, you might be able to keep all the changed pages resident in the buffer pool and reduce the amount of I/O to the page set.
4. Restart the queue manager.
5. When the queue manager has restarted successfully, stop it (using quiesce) and then restart it using the normal buffer pool definition for that page set. After this second restart completes successfully, you can restart your applications
6. Reinstate your normal backup procedures for the restored page.

What happens when the queue manager is restarted:

When the queue manager is restarted, it applies all changes made to the page set that are registered in the log, beginning at the restart point for the page set. WebSphere MQ can recover multiple page sets in this way. The page set is dynamically expanded, if required, during media recovery.

During restart, WebSphere MQ determines the log RBA to start from by taking the lowest value from the following:
- Recovery LSN from the checkpoint log record for each page set.
- Recovery LSN from page zero in each page set.
- The RBA of the oldest incomplete unit of recovery in the system at the time the backup was taken.

All object definitions are stored on page set zero. Messages can be stored on any available page set.

Note: The queue manager cannot restart if page set zero is not available.

How to delete page sets

You delete a page set by using the DELETE PSID command. See WebSphere MQ Script (MQSC) Command Reference for details of this command.

You cannot delete a page set that is still referenced by any storage class. Use DISPLAY STGCLASS to find out which storage classes reference a page set.

The data set is deallocated from WebSphere MQ but is not deleted. It remains available for future use, or can be deleted using z/OS facilities.

Remove the page set from the started task procedure for your queue manager.

Remove the definition of the page set from your CSQINP1 initialization data set.

How to back up and restore queues using CSQUTIL

You can use the CSQUTIL utility functions for backing up and restoring queues. To back up a queue, use the COPY or SCOPY function to copy the messages from a queue onto a data set. To restore the queue, use the complementary function LOAD. For more information, see "WebSphere MQ utility program (CSQUTIL)" on page 159.

Managing buffer pools

This chapter describes how to alter and delete buffer pools. It contains these sections:
- "How to change the number of buffers in a buffer pool"
- "How to delete a buffer pool" on page 100

Buffer pools are defined during queue manager initialization, using DEFINE BUFFPOOL commands issued from the initialization input data set CSQINP1. Their sizes may be altered in response to business requirements while the queue manager is running, using the processes detailed below. The queue manager records the current buffer pool sizes in checkpoint log records. These are automatically restored on subsequent queue manager restart. Use the DISPLAY USAGE command to display the current buffer pools, their sizes, and page set to buffer pool mapping.

You can also define buffer pools dynamically using the DEFINE PSID command with the DSN option, see WebSphere MQ Script (MQSC) Command Reference.

If you change buffer pools dynamically, you should also update their definitions in the initialization data set CSQINP1.

See the WebSphere MQ for z/OS Concepts and Planning Guide for a description of page sets, storage classes, buffers, and buffer pools, and some of the performance considerations that apply.

How to change the number of buffers in a buffer pool

If a buffer pool is too small, as may be shown by message CSQP020E on the console, you can allocate more buffers to it using the ALTER BUFFPOOL command as follows:
1. Determine how much space is available for new buffers by looking at the CSQY220I messages in the log. The available space is reported in MB. As a buffer has a size of 4 KB, each MB of available space allows you to allocate 256 buffers. Do not allocate all the free space to buffers, as some is required for other tasks.

2. If the reported free space is inadequate, release some buffers from another buffer pool using the command

```
ALTER BUFFPOOL(buf-pool-id) BUFFERS(integer)
```

where *buf-pool-id* is the buffer pool from which you want to reclaim space and *integer* is the new number of buffers to be allocated to this buffer pool, which must be smaller than the original number of buffers allocated to it.

3. Add buffers to the buffer pool you want to expand using the command

```
ALTER BUFFPOOL(buf-pool-id) BUFFERS(integer)
```

where *buf-pool-id* is the buffer pool to be expanded and *integer* is the new number of buffers to be allocated to this buffer pool, which must be larger than the original number of buffers allocated to it.

How to delete a buffer pool

When a buffer pool is no longer used by any page sets, delete it to release the virtual storage allocated to it.

You delete a buffer pool using the DELETE BUFFPOOL command. See WebSphere MQ Script (MQSC) Command Reference for details. The command will fail if any page sets are using this buffer pool.

See "How to delete page sets" on page 98 for information on how to delete page sets.

Managing queue-sharing groups and shared queues

This chapter discusses the following topics:
- "Managing queue-sharing groups"
- "Managing shared queues" on page 102
- "Managing group objects" on page 107
- "Managing the Coupling Facility" on page 107

Managing queue-sharing groups

This section describes the following tasks:
- "Adding a queue-sharing group to the DB2 tables"
- "Adding a queue manager to a queue-sharing group" on page 101
- "Removing a queue manager from a queue-sharing group" on page 101
- "Removing a queue-sharing group from the DB2 tables" on page 101

Adding a queue-sharing group to the DB2 tables

To add a queue-sharing group to the DB2 tables, use the ADD QSG function of the queue-sharing group utility (CSQ5PQSG). This program is described in "The queue-sharing group utility (CSQ5PQSG)" on page 210. A sample is provided in thlqual.SCSQPROC(CSQ45AQS).

Adding a queue manager to a queue-sharing group

To add a queue manager to a queue-sharing group, use the ADD QMGR function of the queue-sharing group utility (CSQ5PQSG). This program is described in "The queue-sharing group utility (CSQ5PQSG)" on page 210. A sample is provided in thlqual.SCSQPROC(CSQ45AQM).

The queue-sharing group must exist before you can add queue managers to it.

A queue manager can only be a member of one queue-sharing group.

Note: To add a Version 6.0 queue manager to an existing queue-sharing group containing Version 5.3 or 5.3.1 queue managers, you must first apply the WebSphere MQ for z/OS Version 5 Release 3 Coexistence PTF, as described in the WebSphere MQ for z/OS System Setup Guide.

Removing a queue manager from a queue-sharing group

You can only remove a queue manager from a queue-sharing group if the queue manager's logs are not needed by another process. The logs are needed if they contain:
* the latest backup of one of the Coupling Facility (CF) application structures used by the queue-sharing group
* data needed by a future restore process, that is, the queue manager has used a recoverable structure since the time described by the last backup exclusion interval value.

If either or both of these points apply, the queue manager cannot be removed. To determine which queue managers' logs are needed for a future restore process, use the MQSC DISPLAY CFSTATUS command (see WebSphere MQ Script (MQSC) Command Reference for details of this command).

If the queue manager logs are not needed, take the following steps to remove the queue manager from the queue-sharing group:
1. Resolve any indoubt units of work involving this queue manager.
2. Shut the queue manager down cleanly using STOP QMGR MODE(QUIESCE).
3. Wait for an interval at least equivalent to the value of the EXCLINT parameter you will specify in the BACKUP CFSTRUCT command in the next step.
4. On another queue manager, run a CF structure backup for each recoverable CF structure by using the MQSC BACKUP CFSTRUCT command and specifying an EXCLINT value as required in the previous step.
5. Use the REMOVE QMGR function of the CSQ5PQSG utility to remove the queue manager from the queue-sharing group. This program is described in "The queue-sharing group utility (CSQ5PQSG)" on page 210. A sample is provided in thlqual.SCSQPROC(CSQ45RQM).
6. Before restarting the queue manager, reset the QSGDATA system parameter to its default value. See the WebSphere MQ for z/OS System Setup Guide for information on how to tailor your system parameters.

Removing a queue-sharing group from the DB2 tables

To remove a queue-sharing group from the DB2 tables, use the REMOVE QSG function of the queue-sharing group utility (CSQ5PQSG). This program is described in "The queue-sharing group utility (CSQ5PQSG)" on page 210. A sample is provided in thlqual.SCSQPROC(CSQ45RQS).

You can only remove a queue-sharing group from the common DB2 data-sharing group tables after you have removed all the queue managers from the queue-sharing group (as described in "Removing a queue manager from a queue-sharing group" on page 101).

When the queue-sharing group record is deleted from the queue-sharing group administration table, all objects and administrative information relating to that queue-sharing group are deleted from other WebSphere MQ DB2 tables. This includes shared queue and group object information.

Validating the consistency of DB2 definitions

Problems for shared queues within a queue-sharing group may occur if the DB2 object definitions have, for any reason, become inconsistent.

To validate the consistency of the DB2 object definitions for queue managers, CF structures, and shared queues, use the VERIFY QSG function of the queue-sharing group utility (CSQ5PQSG). This program is described in "The queue-sharing group utility (CSQ5PQSG)" on page 210.

Managing shared queues

This section describes the following tasks:
- "Recovering shared queues"
- "Moving shared queues" on page 103
- "Migrating non-shared queues to shared queues" on page 106

Recovering shared queues

WebSphere MQ can recover persistent messages on shared queues if all of:
- Backups of the CF structures containing the messages have been performed.
- All the logs for all queue managers in the queue-sharing group are available, to perform recovery from the point the backups are taken.
- DB2 is available and the structure backup table is more recent that the most recent CF structure backup.

The messages on a shared queue are stored in a Coupling Facility (CF) structure. Persistent messages can be put onto shared queues, and like persistent messages on non-shared queues, they are copied to the queue manager log. The MQSC BACKUP CFSTRUCT and RECOVER CFSTRUCT commands are provided to allow the recovery of a CF structure in the unlikely event of a Coupling Facility failure. In such circumstances, any nonpersistent messages stored in the affected structure are lost, but persistent messages can be recovered. Any further application activity using the structure is prevented until the structure has been recovered.

To enable recovery, you must back up your Coupling Facility list structures frequently using the MQSC BACKUP CFSTRUCT command . The messages in the CF structure are written onto the active log data set of the queue manager making the backup. It writes a record of the backup to DB2: the name of the CF structure being backed up, the name of the queue manager doing the backup, the RBA range for this backup on that queue manager's log, and the backup time. Note that you should back up CF list structures even if you are not actively using shared queues, for example, if you have set up a queue-sharing group intending to make use of it in the future.

You recover a CF structure by issuing an MQSC RECOVER CFSTRUCT command to the queue manager that will do the recovery; you can use any queue manager in the queue-sharing group. You can specify a single CF structure to be recovered, or you can recover several CF structures simultaneously.

As noted previously, it is important that you back up your CF list structures frequently, otherwise recovering a CF structure can take a very long time. Moreover, the recovery process cannot be cancelled.

The definition of a shared queue is kept in a DB2 database and can therefore be recovered if necessary using standard DB2 database procedures.

For more information about the BACKUP CFSTRUCT and RECOVER CFSTRUCT commands, see WebSphere MQ Script (MQSC) Command Reference.

For more information about planning your backup and recovery strategy for queue-sharing groups, see WebSphere MQ for z/OS Concepts and Planning Guide.

Moving shared queues

This section describes how to perform load balancing by moving a shared queue from one Coupling Facility structure to another. It also describes how to move a non-shared queue to a shared queue, and how to move a shared queue to a non-shared queue.

When you move a queue, you need to define a temporary queue as part of the procedure. This is because every queue must have a unique name, so you cannot have two queues of the same name, even if the queues have different queue dispositions. WebSphere MQ tolerates having two queues with the same name (as in step 2 on page 106), but you cannot use the queues.

Moving a queue from one Coupling Facility structure to another:

To move queues and their messages from one CF structure to another, use the MQSC MOVE QLOCAL command (described in WebSphere MQ Script (MQSC) Command Reference). When you have identified the queue or queues that you want to move to a new CF structure, use the following procedure to move each queue:

1. Ensure that the queue you want to move is not in use by any applications, that is, the queue attributes IPPROCS and OPPROCS are zero on all queue managers in the queue-sharing group.
2. Prevent applications from putting messages on the queue being moved by altering the queue definition to disable **MQPUT**s. Change the queue definition to PUT(DISABLED).
3. Define a temporary queue with the same attributes as the queue that is being moved using the following command:

```
DEFINE QL(TEMP_QUEUE) LIKE(QUEUE_TO_MOVE) PUT(ENABLED) GET(ENABLED) QSGDISP(QMGR)
```

> **Note:** If this temporary queue already exists from a previous run, delete it before doing the define.

4. Move the messages to the temporary queue using the following command:

```
MOVE QLOCAL(QUEUE_TO_MOVE) TOQLOCAL(TEMP_QUEUE)
```

5. Delete the queue you are moving, using the command:

```
DELETE QLOCAL(QUEUE_TO_MOVE)
```

6. Redefine the queue that is being moved, changing the CFSTRUCT attribute, using the following command:

```
DEFINE QL(QUEUE_TO_MOVE) LIKE(TEMP_QUEUE) CFSTRUCT(NEW) QSGDISP(SHARED)
```

When the queue is redefined, it is based on the temporary queue created in step 3 on page 103.

7. Move the messages back to the new queue using the command:

```
MOVE QLOCAL(TEMP) TOQLOCAL(QUEUE_TO_MOVE)
```

8. The queue created in step 3 on page 103 is no longer required. Use the following command to delete it:

```
DELETE QL(TEMP_QUEUE)
```

9. If the queue being moved was defined in the CSQINP2 data sets, change the CFSTRUCT attribute of the appropriate DEFINE QLOCAL command in the CSQINP2 data sets. Add the REPLACE keyword so that the existing queue definition is replaced.

Figure 42 shows a sample job for moving a queue from one CF structure to another.

```
//UTILITY  EXEC  PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB  DD    DSN=thlqual.SCSQANLE,DISP=SHR
//         DD    DSN=thlqual.SCSQAUTH,DISP=SHR
//SYSPRINT DD    SYSOUT=*
//SYSIN    DD    *
COMMAND DDNAME(MOVEQ)
/*
//MOVEQ   DD    *
ALTER QL(QUEUE_TO_MOVE) PUT(DISABLED)
DELETE QL(TEMP_QUEUE) PURGE
DEFINE QL(TEMP_QUEUE) LIKE(QUEUE_TO_MOVE) PUT(ENABLED) GET(ENABLED) QSGDISP(QMGR)
MOVE QLOCAL(QUEUE_TO_MOVE) TOQLOCAL(TEMP_QUEUE)
DELETE QL(QUEUE_TO_MOVE)
DEFINE QL(QUEUE_TO_MOVE) LIKE(TEMP_QUEUE) CFSTRUCT(NEW) QSGDISP(SHARED)
MOVE QLOCAL(TEMP_QUEUE) TOQLOCAL(QUEUE_TO_MOVE)
DELETE QL(TEMP_QUEUE)
/*
```

Figure 42. Sample job for moving a queue from one CF structure to another

Moving a non-shared queue to a shared queue:

The procedure for moving a non-shared queue to a shared queue is very similar to the procedure for moving a queue from one CF structure to another (see "Moving a queue from one Coupling Facility structure to another" on page 103). Figure 43 gives a sample job to do this.

Note: Remember that messages on shared queues are subject to certain restrictions on the maximum message size, message persistence, and queue index type, so you might not be able to move some non-shared queues to a shared queue.

```
//UTILITY  EXEC  PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB  DD    DSN=thlqual.SCSQANLE,DISP=SHR
//         DD    DSN=thlqual.SCSQAUTH,DISP=SHR
//SYSPRINT DD    SYSOUT=*
//SYSIN    DD    *
COMMAND DDNAME(MOVEQ)
/*
//MOVEQ  DD    *
ALTER QL(QUEUE_TO_MOVE) PUT(DISABLED)
DELETE QL(TEMP_QUEUE) PURGE
DEFINE QL(TEMP_QUEUE) LIKE(QUEUE_TO_MOVE) PUT(ENABLED) GET(ENABLED)
MOVE QLOCAL(QUEUE_TO_MOVE) TOQLOCAL(TEMP_QUEUE)
DELETE QL(QUEUE_TO_MOVE)
DEFINE QL(QUEUE_TO_MOVE) LIKE(TEMP_QUEUE) CFSTRUCT(NEW) QSGDISP(SHARED)
MOVE QLOCAL(TEMP_QUEUE) TOQLOCAL(QUEUE_TO_MOVE)
DELETE QL(TEMP_QUEUE)
/*
```

Figure 43. Sample job for moving a non-shared queue to a shared queue

Moving a shared queue to a non-shared queue:

The procedure for moving a shared queue to a non-shared queue is very similar to the procedure for moving a queue from one CF structure to another (see "Moving a queue from one Coupling Facility structure to another" on page 103). Figure 44 gives a sample job to do this.

```
//UTILITY  EXEC  PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB  DD    DSN=thlqual.SCSQANLE,DISP=SHR
//         DD    DSN=thlqual.SCSQAUTH,DISP=SHR
//SYSPRINT DD    SYSOUT=*
//SYSIN    DD    *
COMMAND DDNAME(MOVEQ)
/*
//MOVEQ  DD    *
ALTER QL(QUEUE_TO_MOVE) PUT(DISABLED)
DELETE QL(TEMP_QUEUE) PURGE
DEFINE QL(TEMP_QUEUE) LIKE(QUEUE_TO_MOVE) PUT(ENABLED) GET(ENABLED) QSGDISP(QMGR)
MOVE QLOCAL(QUEUE_TO_MOVE) TOQLOCAL(TEMP_QUEUE)
DELETE QL(QUEUE_TO_MOVE)
DEFINE QL(QUEUE_TO_MOVE) LIKE(TEMP_QUEUE) STGCLASS(NEW) QSGDISP(QMGR)
MOVE QLOCAL(TEMP_QUEUE) TOQLOCAL(QUEUE_TO_MOVE)
DELETE QL(TEMP_QUEUE)
/*
```

Figure 44. Sample job for moving a shared queue to a non-shared queue

Migrating non-shared queues to shared queues

There are two stages to migrating non-shared queues to shared queues:
- migrating the first queue manager
- migrating the other queue managers

Migrating the first (or only) queue manager in the queue-sharing group:

Figure 43 on page 105 shows an example job for moving a non-shared queue to a shared queue. Do this for each queue that needs migrating.

Note:

1. Messages on shared queues are subject to certain restrictions on the maximum message size, message persistence, and queue index type, so you might not be able to move some non-shared queues to a shared queue.

2. You must use the correct index type for shared queues. If you migrate a transmission queue to be a shared queue, the index type must be MSGID.

If the queue is empty, or you do not need to keep the messages that are on it, migrating the queue is simpler. Figure 45 shows an example job to use in these circumstances.

```
//UTILITY  EXEC  PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB  DD    DSN=thlqual.SCSQANLE,DISP=SHR
//         DD    DSN=thlqual.SCSQAUTH,DISP=SHR
//SYSPRINT DD    SYSOUT=*
//SYSIN    DD    *
COMMAND DDNAME(MOVEQ)
/*
//MOVEQ  DD    *
DELETE QL(TEMP_QUEUE) PURGE
DEFINE QL(TEMP_QUEUE) LIKE(QUEUE_TO_MOVE) PUT(ENABLED) GET(ENABLED)
DELETE QL(QUEUE_TO_MOVE)
DEFINE QL(QUEUE_TO_MOVE) LIKE(TEMP_QUEUE) CFSTRUCT(NEW) QSGDISP(SHARED)
DELETE QL(TEMP_QUEUE)
/*
```

Figure 45. Sample job for moving a non-shared queue without messages to a shared queue

Migrating any other queue managers in the queue-sharing group:

1. For each queue that does not have the same name as an existing shared queue, move the queue as described in Figure 43 on page 105 or Figure 45.

2. For queues that have the same name as an existing shared queue, move the messages to the shared queue using the commands shown in Figure 46.

```
MOVE QLOCAL(QUEUE_TO_MOVE) QSGDISP(QMGR) TOQLOCAL(QUEUE_TO_MOVE)
DELETE QLOCAL(QUEUE_TO_MOVE) QSGDISP(QMGR)
```

Figure 46. Moving messages from a non-shared queue to an existing shared queue

Suspending a connection to DB2

If you want to apply maintenance or service to the DB2 tables or plans related to shared queues without stopping your queue manager, you must temporarily disconnect from DB2, using the MQSC command SUSPEND QMGR FACILITY(DB2). Reconnect to DB2 using the MQSC command RESUME QMGR

FACILITY(DB2). There are restrictions on the use of these commands. See
WebSphere MQ Script (MQSC) Command Reference for details.

Managing group objects

WebSphere MQ automatically copies the definition of a group object to page set
zero of each queue manager that uses it. You can alter the copy of the definition
temporarily, and WebSphere MQ allows you to refresh the page set copies from the
repository copy. WebSphere MQ always tries to refresh the page set copies from
the repository copy on start up (for channel objects, this is done when the channel
initiator restarts). This ensures that the page set copies reflect the version on the
repository, including any changes that were made when the queue manager was
inactive.

There are circumstances under which the refresh is not performed, for example:
- If a copy of the queue is open, a refresh that would change the usage of the
  queue fails.
- If a copy of a queue has messages on it, a refresh that would delete that queue
  fails.

In these circumstances, the refresh is not performed on that copy, but is performed
on the copies on all other queue managers. Check for and correct any problems
with copy objects after adding, changing, or deleting a group object, and at queue
manager or channel initiator restart.

Managing the Coupling Facility

This section describes the following tasks:
- "Adding a Coupling Facility structure"
- "Removing a Coupling Facility structure"

Adding a Coupling Facility structure

There are no WebSphere MQ actions required when you add a Coupling Facility
structure. The information about setting up the Coupling Facility in the WebSphere
MQ for z/OS System Setup Guide describes the rules for naming Coupling Facility
structures, and how to define structures in the CFRM policy data set.

Removing a Coupling Facility structure

To remove a Coupling Facility structure, follow this procedure:
- Use the following command to get a list of all the queues using the Coupling
  Facility structure that you want to delete:

```
DISPLAY QUEUE(*) QSGDISP(SHARED) CFSTRUCT(structure-name)
```

- Delete all the queues that use the structure.
- Stop and restart each queue manager in the queue-sharing group in turn to
  disconnect WebSphere MQ and DB2 from the structure and delete information
  about it. (You do not need to stop all the queue managers at once; just one at a
  time.)

- Remove the structure definition from your CFRM policy data set and run the IXCMIAPU utility. (This is the reverse of customization task 10 (set up the Coupling Facility) described in the WebSphere MQ for z/OS System Setup Guide.)

Chapter 5. Recovery and restart

Restarting WebSphere MQ

This chapter discusses how to restart your queue manager in the following circumstances:

- "Restarting after a normal shutdown"
- "Restarting after an abnormal termination"
- "Restarting if you have lost your page sets"
- "Restarting if you have lost your log data sets" on page 110
- "Recovering a single queue manager at an alternative site" on page 111
- "Recovering a queue-sharing group at the alternative site" on page 115
- "Reinitializing a queue manager" on page 116

Restarting after a normal shutdown

If the queue manager was stopped with the STOP QMGR command, the system finishes its work in an orderly way and takes a termination checkpoint before stopping. When you restart the queue manager, it uses information from the system checkpoint and recovery log to determine the system status at shutdown.

To restart the queue manager, issue the START QMGR command as described in "Starting and stopping a queue manager" on page 15.

Restarting after an abnormal termination

WebSphere MQ automatically detects whether restart follows a normal shutdown or an abnormal termination.

Starting the queue manager after it has terminated abnormally is different to starting it after the STOP QMGR command has been issued. If the queue manager terminates abnormally, it terminates without being able to finish its work or take a termination checkpoint.

To restart the queue manager, issue the START QMGR command as described in "Starting and stopping a queue manager" on page 15. When you restart a queue manager after an abnormal termination, it refreshes its knowledge of its status at termination using information in the log, and notifies you of the status of various tasks.

Normally, the restart process resolves all inconsistent states. But, in some cases, you must take specific steps to resolve inconsistencies. This is described in "Recovering units of work manually" on page 123.

Restarting if you have lost your page sets

If you have lost your page sets, you need to restore them from your backup copies before you can restart the queue manager. This is described in "How to back up and recover page sets" on page 94.

The queue manager might take a long time to restart under these circumstances because of the length of time needed for media recovery.

Restarting if you have lost your log data sets

If, after stopping a queue manager (using the STOP QMGR command), both copies of the log are lost or damaged, you can restart the queue manager providing you have a consistent set of page sets (produced using "Method 1: Full backup" on page 95).

Follow this procedure:

1. Define new page sets to correspond to each existing page set in your queue manager. See the WebSphere MQ for z/OS System Setup Guide for information about page set definition.

   Ensure that each new page set is larger than the corresponding source page set.
2. Use the FORMAT function of CSQUTIL to format the destination page set. See "Formatting page sets (FORMAT)" on page 162 for more details.
3. Use the RESETPAGE function of CSQUTIL to copy the existing page sets or reset them in place, and reset the log RBA in each page. See "Copying a page set and resetting the log (RESETPAGE)" on page 169 for more information about this function.
4. Redefine your queue manager log data sets and BSDS using CSQJU003 (see "The change log inventory utility (CSQJU003)" on page 191).
5. Restart the queue manager using the new page sets. To do this, you do one of the following:
   - Change the queue manager started task procedure to reference the new page sets. See the WebSphere MQ for z/OS System Setup Guide for more information.
   - Use Access Method Services to delete the old page sets and then rename the new page sets, giving them the same names as the old page sets.

Attention: Before you delete any WebSphere MQ page set, ensure that you have made the required backup copies.

If the queue manager is a member of a queue-sharing group, GROUP and SHARED object definitions are not normally affected by lost or damaged logs. However, if any shared-queue messages are involved in a unit of work that was covered by the lost or damaged logs, the effect on such uncommitted messages is unpredictable.

Note: If logs are damaged and the queue manager is a member of a queue-sharing group, the ability to recover shared persistent messages might be lost. Issue a BACKUP CFSTRUCT command immediately on another active queue manager in the queue-sharing group for all CF structures with the RECOVER(YES) attribute.

Restarting if you have lost your CF structures

You do not need to restart if you lose your CF structures, because the queue manager does not terminate.

Recovering a single queue manager at an alternative site

In the case of a total loss of a WebSphere MQ computing center, you can recover on another queue manager or queue-sharing group at a recovery site. (See "Recovering a queue-sharing group at the alternative site" on page 115 for the alternative site recovery procedure for a queue-sharing group.)

To recover on another queue manager at a recovery site, you must regularly back up the page sets and the logs. As with all data recovery operations, the objectives of disaster recovery are to lose as little data, workload processing (updates), and time, as possible.

At the recovery site:
- The recovery queue managers **must** have the same names as the lost queue managers.
- The system parameter module (for example, CSQZPARM) used on each recovery queue manager must contain the same parameters as the corresponding lost queue manager.

When you have done this, reestablish all your queue managers as described in the following procedure. This can be used to perform disaster recovery at the recovery site for a single queue manager. It assumes that all that is available are:
- Copies of the archive logs and BSDSs created by normal running at the primary site (the active logs will have been lost along with the queue manager at the primary site).
- Copies of the page sets from the queue manager at the primary site that are the same age or older than the most recent archive log copies available.

You can use dual logging for the active and archive logs, in which case you need to apply the BSDS updates to both copies:

1. Define new page set data sets and load them with the data in the copies of the page sets from the primary site.
2. Define new active log data sets.
3. Define a new BSDS data set and use Access Method Services REPRO to copy the *most recent* archived BSDS into it.
4. Use the print log map utility CSQJU004 to print information from this most recent BSDS. At the time this BSDS was archived, the most recent archived log you have would have just been truncated as an active log, and does not appear as an archived log. Record the STARTRBA and ENDRBA of this log.
5. Use the change log inventory utility, CSQJU003, to register this latest archive log data set in the BSDS that you have just restored, using the STARTRBA and ENDRBA recorded in Step 4.
6. Use the DELETE option of CSQJU003 to remove all active log information from the BSDS.
7. Use the NEWLOG option of CSQJU003 to add active logs to the BSDS, do not specify STARTRBA or ENDRBA.
8. Use CSQJU003 to add a restart control record to the BSDS. Specify `CRESTART CREATE,ENDRBA=highrba`, where `highrba` is the high RBA of the most recent archive log available (found in Step 4), plus 1.

   The BSDS now describes all active logs as being empty, all the archived logs you have available, and no checkpoints beyond the end of your logs.
9. Restart the queue manager with the usual START QMGR command. During initialization, an operator reply message such as the following is issued:

```
CSQJ245D +CSQ1 RESTART CONTROL INDICATES TRUNCATION AT RBA highrba.
         REPLY Y TO CONTINUE, N TO CANCEL
```

Type Y to start the queue manager. The queue manager starts, and recovers data up to ENDRBA specified in the CRESTART statement.

See Chapter 6, "Using the WebSphere MQ Utilities," on page 155 for information about using CSQJU003 and CSQJU004.

Figure 47 shows sample input statements for CSQJU003 for steps 6, 7, and 8:

```
* Step 6
        DELETE DSNAME=MQM2.LOGCOPY1.DS01
        DELETE DSNAME=MQM2.LOGCOPY1.DS02
        DELETE DSNAME=MQM2.LOGCOPY1.DS03
        DELETE DSNAME=MQM2.LOGCOPY1.DS04
        DELETE DSNAME=MQM2.LOGCOPY2.DS01
        DELETE DSNAME=MQM2.LOGCOPY2.DS02
        DELETE DSNAME=MQM2.LOGCOPY2.DS03
        DELETE DSNAME=MQM2.LOGCOPY2.DS04

* Step 7
        NEWLOG DSNAME=MQM2.LOGCOPY1.DS01,COPY1
        NEWLOG DSNAME=MQM2.LOGCOPY1.DS02,COPY1
        NEWLOG DSNAME=MQM2.LOGCOPY1.DS03,COPY1
        NEWLOG DSNAME=MQM2.LOGCOPY1.DS04,COPY1
        NEWLOG DSNAME=MQM2.LOGCOPY2.DS01,COPY2
        NEWLOG DSNAME=MQM2.LOGCOPY2.DS02,COPY2
        NEWLOG DSNAME=MQM2.LOGCOPY2.DS03,COPY2
        NEWLOG DSNAME=MQM2.LOGCOPY2.DS04,COPY2

* Step 8
        CRESTART CREATE,ENDRBA=063000
```

Figure 47. Sample input statements for CSQJU003

The things you need to consider for restarting the channel initiator at the recovery site are similar to those faced when using ARM to restart the channel initiator on a different z/OS image. See "Using ARM in a WebSphere MQ network" on page 121 for more information. Your recovery strategy should also cover recovery of the WebSphere MQ product libraries and the application programming environments that use WebSphere MQ (CICS, for example).

Other functions of the change log inventory utility (CSQJU003) can also be used in disaster recovery scenarios. The HIGHRBA function allows the update of the highest RBA written and highest RBA off-loaded values within the bootstrap data set. The CHECKPT function allows the addition of new checkpoint queue records or the deletion of existing checkpoint queue records in the BSDS.

Attention: These functions might affect the integrity of your WebSphere MQ data. Only use them in disaster recovery scenarios under the guidance of IBM® service personnel.

Fast copy techniques

If copies of all the page sets and logs are made while the queue manager is frozen, the copies will be a consistent set that can be used to restart the queue manager at

an alternative site. They typically enable a much faster restart of the queue manager, as there is very little media recovery to be performed.

Use the SUSPEND QMGR LOG command to freeze the queue manager. This command flushes buffer pools to the page sets, takes a checkpoint, and stops any further log write activity. Once log write activity has been suspended, the queue manager is effectively frozen until you issue a RESUME QMGR LOG command. Whilst the queue manager is frozen, the page sets and logs can be copied.

By using copying tools such as FLASHCOPY or SNAPSHOT to rapidly copy the page sets and logs, the time during which the queue manager is frozen can be reduced to a minimum.

Within a queue-sharing group, however, the SUSPEND QMGR LOG command might not be such a good solution. To be effective, the copies of the logs must all contain the same point in time for recovery, which means that the SUSPEND QMGR LOG command must be issued on all queue managers within the queue-sharing group simultaneously, and therefore the entire queue-sharing group will be frozen for some time.

Recovering a queue-sharing group

In the event of a prime site disaster, you can restart a queue-sharing group at a remote site using backup data sets from the prime site. To recover a queue-sharing group you need to coordinate the recovery across all the queue managers in the queue-sharing group, and coordinate with other resources, primarily DB2. This section describes these tasks in detail.

CF structure media recovery

Media recovery of a CF structure used to hold persistent messages on a shared queue, relies on having a backup of the media that can be forward recovered by the application of logged updates. Take backups of your CF structures periodically using the MQSC BACKUP CFSTRUCT command. All updates to shared queues (**MQGET**s and **MQPUT**s) are written on the log of the queue manager where the update is performed. To perform media recovery of a CF structure you must apply logged updates to that backup from the logs of **all** the queue managers that have used that CF structure. When you use the MQSC RECOVER CFSTRUCT command, WebSphere MQ automatically merges the logs from relevant queue managers, and applies the updates to the most recent backup.

The CF structure backup is written to the log of the queue manager that processed the BACKUP CFSTRUCT command, so there are no additional data sets to be collected and transported to the alternative site.

Backing up the queue-sharing group at the prime site

At the prime site you need to establish a consistent set of backups on a regular basis, which can be used in the event of a disaster to rebuild the queue-sharing group at an alternative site. For a single queue manager, recovery can be to an arbitrary point in time, usually the end of the logs available at the remote site. However, where persistent messages have been stored on a shared queue, the logs of all the queue managers in the queue-sharing group must be merged to recover shared queues, as any queue manager in the queue-sharing group might have performed updates (**MQPUT**s or **MQGET**s) on the queue.

For recovery of a queue-sharing group, you need to establish a point in time that is within the log range of the log data of all queue managers. However, as you can only **forward** recover media from the log, this point in time must be after the BACKUP CFSTRUCT command has been issued and after any page set backups have been performed. (Typically, the point in time for recovery might correspond to the end of a business day or week.)

The following diagram shows time lines for two queue managers in a queue-sharing group. For each queue manager, fuzzy backups of page sets are taken (see "Method 2: Fuzzy backup" on page 95). On queue manager A, a BACKUP CFSTRUCT command is issued. Subsequently, an ARCHIVE LOG command is issued on each queue manager to truncate the active log, and copy it to media offline from the queue manager, which can be transported to the alternative site. 'End of log' identifies the time at which the ARCHIVE LOG command was issued, and therefore marks the extent of log data typically available at the alternative site. The point in time for recovery must lie between the end of any page set or CF structure backups, and the earliest end of log available at the alternative site.
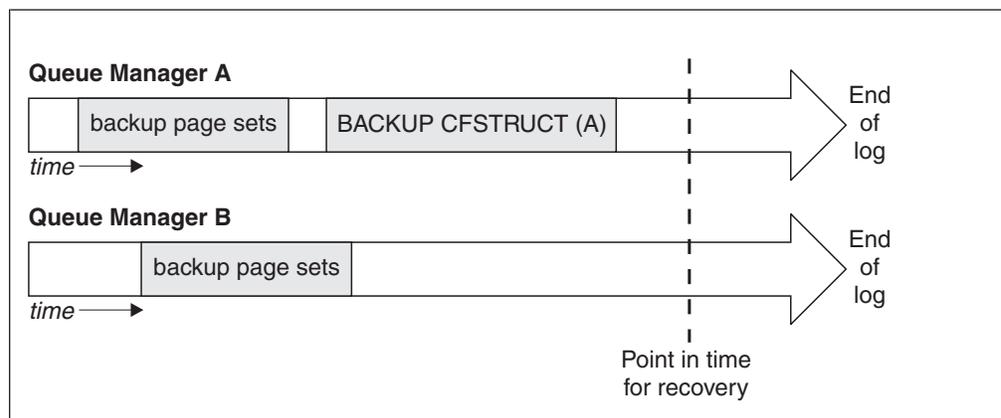

Figure 48. Point in time for recovery for 2 queue managers in a queue-sharing group

WebSphere MQ records information associated with the CF structure backups in a table in DB2. Depending on your requirements, you might want to coordinate the point in time for recovery of WebSphere MQ with that for DB2, or it might be sufficient to take a copy of the WebSphere MQ CSQ.ADMIN_B_STRBACKUP table after the BACKUP CFSTRUCT commands have finished.

To prepare for a recovery:
1. Create page set backups for each queue manager in the queue-sharing group.
2. Issue a BACKUP CFSTRUCT command for each CF structure with the RECOVER(YES) attribute. You can issue these commands from a single queue manager, or from different queue managers within the queue-sharing group to balance the workload.
3. Once all the backups have completed, issue an ARCHIVE LOG command to switch the active log and create copies of the logs and BSDSs of each queue manager in the queue-sharing group.
4. Transport the page set backups, the archived logs, the archived BSDS of all the queue managers in the queue-sharing group, and your chosen DB2 backup information, offsite.

Recovering a queue-sharing group at the alternative site

Before you can recover the queue-sharing group, you need to prepare the environment:

1. If you have old information in your Coupling Facility from practice startups when you installed the queue-sharing group, you need to clean this out first (if you do not have old information in the Coupling Facility, you can omit this step:

   a. Enter the following z/OS command to display the CF structures for this queue-sharing group:

```
D XCF,STRUCTURE,STRNAME=qsgname
```

   b. For all structures that start with the queue-sharing group name, use the z/OS command SETXCF FORCE CONNECTION to force the connection off those structures:

```
SETXCF FORCE,CONNECTION,STRNAME=strname,CONNAME=ALL
```

   c. Delete all the CF structures using the following command for each structure:

```
SETXCF FORCE,STRUCTURE,STRNAME=strname
```

2. Restore DB2 systems and data-sharing groups
3. Recover the CSQ.ADMIN_B_STRBACKUP table so that it contains information about the most recent structure backups taken at the prime site

Note: It is important that the STRBACKUP table contains the most recent structure backup information. Older structure backup information might require data sets that you have discarded as a result of the information given by a recent DISPLAY USAGE TYPE(DATASET) command, which would mean that your recovered CF structure would not contain accurate information.

To recover the queue managers in the queue-sharing group:

1. Define new page set data sets and load them with the data in the copies of the page sets from the primary site.
2. Define new active log data sets.
3. Define a new BSDS data set and use Access Method Services REPRO to copy the *most recent* archived BSDS into it.
4. Use the print log map utility CSQJU004 to print information from this most recent BSDS. At the time this BSDS was archived, the most recent archived log you have would have just been truncated as an active log, and does not appear as an archived log. Record the STARTRBA, STARTLRSN, ENDRBA, and ENDLRSN values of this log.
5. Use the change log inventory utility, CSQJU003, to register this latest archive log data set in the BSDS that you have just restored, using the values recorded in Step 4.
6. Use the DELETE option of CSQJU003 to remove all active log information from the BSDS.

7. Use the NEWLOG option of CSQJU003 to add active logs to the BSDS, do not specify STARTRBA or ENDRBA.

8. Calculate the *recoverylrsn* for the queue-sharing group. The *recoverylrsn* is the lowest of the ENDLRSNs across all queue managers in the queue-sharing group (as recorded in Step 4 on page 115), minus 1. For example, if there are two queue managers in the queue-sharing group, and the ENDLRSN for one of them is B713 3C72 22C5, and for the other is B713 3D45 2123, the *recoverylrsn* is B713 3C72 22C4.

9. Use CSQJU003 to add a restart control record to the BSDS. Specify:

```
CRESTART CREATE,ENDLRSN=recoverylrsn
```

where *recoverylrsn* is the value you recorded in Step 8.

The BSDS now describes all active logs as being empty, all the archived logs you have available, and no checkpoints beyond the end of your logs.

You must add the CRESTART record to the BSDS for each queue manager within the queue-sharing group.

10. Restart each queue manager in the queue-sharing group with the usual START QMGR command. During initialization, an operator reply message such as the following is issued:

```
CSQJ245D +CSQ1 RESTART CONTROL INDICATES TRUNCATION AT RBA highrba.
         REPLY Y TO CONTINUE, N TO CANCEL
```

Reply Y to start the queue manager. The queue manager starts, and recovers data up to ENDRBA specified in the CRESTART statement.

11. Once all the queue managers are active, issue a RECOVER CFSTRUCT command for each CF application structure.

If you issue the RECOVER CFSTRUCT command for all structures on a single queue manager, the log merge process is only performed once, so is quicker than issuing the command on a different queue manager for each CF structure, where each queue manager has to perform the log merge step.

Reinitializing a queue manager

If the queue manager has terminated abnormally you might not be able to restart it. This could be because your page sets or logs have been lost, truncated, or corrupted. If this has happened, you might have to reinitialize the queue manager (perform a cold start).

Attention

You should only perform a cold start if you cannot restart the queue manager any other way. Performing a cold start enables you to recover your queue manager and your object definitions; you will **not** be able to recover your message data. Check that none of the other restart scenarios described in this chapter will work for you before you do this.

When you have restarted, all your WebSphere MQ objects are defined and available for use, but there is no message data.

Note: Do not reinitialize a queue manager while it is part of a cluster. You must first remove the queue manager from the cluster (using RESET CLUSTER commands on the other queue managers in the cluster), then reinitialize it, and finally reintroduce it to the cluster as a new queue manager.

This is because during reinitialization, the queue manager identifier (QMID) is changed, so any cluster object with the old queue manager identifier must be removed from the cluster.

Reinitializing a queue manager that is not in a queue-sharing group

To cold start a queue manager, follow this procedure:

1. Prepare the object definition statements that will be used when you restart the queue manager. To do this, either:
   - If page set zero is available, use the CSQUTIL SDEFS function (see "Producing a list of WebSphere MQ define commands (SDEFS)" on page 178). You must get definitions for all object types (authentication information objects, CF structures, channels, namelists, processes, queues, and storage classes).
   - If page set zero is not available, use the definitions from the last time you backed up your object definitions.
2. Redefine your queue manager data sets (do not do this until you have completed step 1). See the WebSphere MQ for z/OS System Setup Guide for information about redefining your log data sets, BSDS, and page sets.
3. Restart the queue manager using the newly defined and initialized log data sets, BSDS, and page sets. Use the object definition input statements that you created in step 1 as input in the CSQINP2 initialization input data set.

Reinitializing queue managers in a queue-sharing group

In a queue-sharing group, reinitializing a queue manager is more complex. It might be necessary to reinitialize one or more queue managers because of page set or log problems, but there might also be problems with DB2 or the Coupling Facility to deal with. Because of this, there are a number of alternatives:

Cold start
Reinitializing the entire queue-sharing group involves forcing all the Coupling Facilities structures, clearing all object definitions for the queue-sharing group from DB2, deleting or redefining the logs and BSDS, and formatting page sets for all the queue managers in the queue-sharing group.

Shared definitions retained
Delete or redefine the logs and BSDS, format page sets for all queue managers in the queue-sharing group, and force all the Coupling Facilities structures. On restart, all messages will have been deleted. The queue managers recreate COPY objects that correspond to GROUP objects that still exist in the DB2 database. Any shared queues still exist and can be used.

Single queue manager reinitialized
Delete or redefine the logs and BSDS, and format page sets for the single queue manager (this deletes all its private objects and messages). On restart, the queue manager recreates COPY objects that correspond to GROUP objects that still exist in the DB2 database. Any shared queues still exist, as do the messages on them, and can be used.

Point in time recovery of a queue-sharing group
This is the alternative site disaster recovery scenario.

Shared objects are recovered to the point in time achieved by DB2 recovery (described in "A DB2 system fails" on page 148). Each queue manager can be recovered to a point in time achievable from the backup copies available at the alternative site.

Persistent messages can be used in queue-sharing groups, and can be recovered using the MQSC RECOVER CFSTRUCT command. Note that this command recovers to the time of failure. However, there is no recovery of nonpersistent shared queue messages; they will all be lost unless you have made backup copies independently using the COPY function of the CSQUTIL utility program.

It is not necessary to try to restore each queue manager to the same point in time because there are no interdependencies between the local objects on different queue managers (which are what is actually being recovered), and the queue manager resynchronization with DB2 on restart creates or deletes COPY objects as necessary on a queue manager by queue manager basis.

Using the z/OS Automatic Restart Manager (ARM)

This chapter discusses the following topics:
- "What is the ARM?"
- "ARM couple data sets" on page 119
- "ARM policies" on page 119
- "Using ARM in a WebSphere MQ network" on page 121

What is the ARM?

The z/OS Automatic Restart Manager (ARM) is a z/OS recovery function that can improve the availability of your queue managers. When a job or task fails, or the system on which it is running fails, ARM can restart the job or task without operator intervention.

If a queue manager or a channel initiator has failed, ARM restarts it on the same z/OS image. If z/OS, and hence a whole group of related subsystems and applications have failed, ARM can restart all the failed systems automatically, in a predefined order, on another z/OS image within the sysplex. This is called a *cross-system restart*.

The channel initiator should be restarted by ARM only in exceptional circumstances. If the queue manager is restarted by ARM, you should restart the channel initiator from the CSQINP2 initialization data set (see "Using ARM in a WebSphere MQ network" on page 121).

You can use ARM to restart a queue manager on a different z/OS image within the sysplex in the event of z/OS failure. The network implications of WebSphere MQ ARM restart on a different z/OS image are discussed in "Using ARM in a WebSphere MQ network" on page 121.

To enable automatic restart:
- Set up an ARM couple data set.

- Define the automatic restart actions that you want z/OS to perform in an *ARM policy*.
- Start the ARM policy.

Also, WebSphere MQ must register with ARM at startup (this happens automatically).

Note: If you want to restart queue managers in different z/OS images automatically, you must define every queue manager as a subsystem in each z/OS image on which that queue manager might be restarted, with a sysplex wide unique 4-character subsystem name.

ARM couple data sets

Ensure that you define the couple data sets required for ARM, and that they are online and active before you start any queue manager for which you want ARM support. WebSphere MQ automatic ARM registration fails if the couple data sets are not available at queue manager startup. In this situation, WebSphere MQ assumes that the absence of the couple data set means that you do not want ARM support, and initialization continues.

See the *OS/390 MVS Setting up a Sysplex* manual for information about ARM couple data sets.

ARM policies

ARM functions are controlled by a user-defined *ARM policy*. Each z/OS image running a queue manager instance that is to be restarted by ARM must be connected to an ARM couple data set with an active ARM policy.

IBM provides a default ARM policy. You can define new policies, or override the policy defaults by using the *administrative data utility* (IXCMIAPU) provided with z/OS. The *OS/390 MVS Setting up a Sysplex* manual describes this utility, and includes full details of how to define an ARM policy.

Figure 49 shows an example of an ARM policy. This sample policy restarts any queue manager within a sysplex, in the event that either the queue manager failed, or a whole system failed.

```
//IXCMIAPU EXEC PGM=IXCMIAPU,REGION=2M
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  DATA TYPE(ARM)
DEFINE POLICY NAME(ARMPOL1) REPLACE(YES)
 RESTART_GROUP(DEFAULT)
   ELEMENT(*)
     RESTART_ATTEMPTS(0)    /* Jobs not to be restarted by ARM   */
 RESTART_GROUP(GROUP1)
   ELEMENT(SYSMQMGRMQ*)     /* These jobs to be restarted by ARM */
/*
```

Figure 49. Sample ARM policy

Defining an ARM policy

Set up your ARM policy as follows:

- Define RESTART_GROUPs for each queue manager instance that also contain any CICS or IMS subsystems that connect to that queue manager instance. If you use a subsystem naming convention, you might be able to use the '?' and '*' wild-card characters in your element names to define RESTART_GROUPs with minimum definition effort.
- Specify TERMTYPE(ELEMTERM) for your channel initiators to indicate that they will be restarted only if the channel initiator has failed and the z/OS image has not failed.
- Specify TERMTYPE(ALLTERM) for your queue managers to indicate that they will be restarted if either the queue manager has failed or the z/OS image has failed.
- Specify RESTART_METHOD(BOTH, PERSIST) for both queue managers and channel initiators. This tells ARM to restart using the JCL it saved (after resolution of system symbols) during the last startup. It tells ARM to do this irrespective of whether the individual element failed, or the z/OS image failed.
- Accept the default values for all the other ARM policy options.

Activating an ARM policy

To start your automatic restart management policy, issue the following z/OS command:

```
SETXCF START,POLICY,TYPE=ARM,POLNAME=mypol
```

When the policy is started, all systems connected to the ARM couple data set use the same active policy.

Use the SETXCF STOP command to disable automatic restarts.

Registering with ARM:

WebSphere MQ registers automatically as an *ARM element* during queue manager startup (subject to ARM availability). It deregisters during its shutdown phase, unless requested not to.

At startup, the queue manager determines whether ARM is available. If it is, WebSphere MQ registers using the name SYSMQMGR*ssid*, where *ssid* is the 4-character queue manager name, and SYSMQMGR is the element type.

The STOP QMGR MODE(QUIESCE) and STOP QMGR MODE(FORCE) commands deregister the queue manager from ARM (if it was registered with ARM at startup). This prevents ARM restarting this queue manager. The STOP QMGR MODE(RESTART) command does not deregister the queue manager from ARM, so it is eligible for immediate automatic restart.

Each channel initiator address space determines whether ARM is available, and if so registers with the element name SYSMQCH*ssid*, where *ssid* is the queue manager name, and SYSMQCH is the element type.

The channel initiator is always deregistered from ARM when it stops normally, and remains registered only if it ends abnormally. The channel initiator is always deregistered if the queue manager fails.

Using ARM in a WebSphere MQ network

Set up your queue manager so that the channel initiators and associated listeners are started automatically when the queue manager is restarted. To ensure fully automatic queue manager restart on the same z/OS image for both LU 6.2 and TCP/IP communication protocols:

- Start your channel initiator automatically by adding the appropriate START CHINIT command to the CSQINP2 data set.
- Start your listeners automatically by adding the appropriate START LISTENER command to the CSQINPX data set.

See the WebSphere MQ for z/OS System Setup Guide for information about the CSQINP2 and CSQINPX data sets.

Restarting on a different z/OS image with LU 6.2

If you use only LU 6.2 communication protocols, carry out the following procedure to enable network reconnect after automatic restart of a queue manager on a different z/OS image within the sysplex:

- Define each queue manager within the sysplex with a unique subsystem name.
- Define each channel initiator within the sysplex with a unique LUNAME. This is specified in both the queue manager attributes and in the START LISTENER command.

  Note: The LUNAME names an entry in the APPC side table, which in turn maps this to the actual LUNAME.

- Set up a shared APPC side table, which is referenced by each z/OS image within the sysplex. This should contain an entry for each channel initiator's LUNAME. See the *MVS Planning: APPC/MVS Management* manual for information about this.
- Set up an APPCPM*xx* member of SYS1.PARMLIB for each channel initiator within the sysplex to contain an LUADD to activate the APPC side table entry for that channel initiator. These members should be shared by each z/OS image. The appropriate SYS1.PARMLIB member is activated by a z/OS command SET APPC=*xx*, which is issued automatically during ARM restart of the queue manager (and its channel initiator) on a different z/OS image, as described below.
- Use the LU62ARM queue manager attribute to specify the *xx* suffix of this SYS1.PARMLIB member for each channel initiator. This causes the channel initiator to issue the required z/OS command SET APPC=*xx* to activate its LUNAME.

Define your ARM policy so that it restarts the channel initiator only if it fails while its z/OS image stays up; the user ID associated with the XCFAS address space must be authorized to issue the WebSphere MQ command START CHINIT. Do not restart the channel initiator automatically if its z/OS image also fails, instead use commands in the CSQINP2 and CSQINPX data sets to start the channel initiator and listeners.

Restarting on a different z/OS image with TCP/IP

If you are using TCP/IP as your communication protocol, and you are using virtual IP addresses, you can configure these to recover on other z/OS images, allowing channels connecting to that queue manager to reconnect without any

changes. Otherwise, you can reallocate a TCP/IP address after moving a queue manager to a different z/OS image only if you are using clusters or if you are connecting to a queue-sharing group using a WLM dynamic Domain Name System (DNS) logical group name.

When using clustering:

z/OS ARM responds to a system failure by restarting the queue manager on a different z/OS image in the same sysplex; this system has a different TCP/IP address to the original z/OS image. The following explains how you can use WebSphere MQ clusters to reassign a queue manager's TCP/IP address after it has been moved by ARM restart to a different z/OS image.

When a client queue manager detects the queue manager failure (as a channel failure), it responds by reallocating suitable messages on its cluster transmission queue to a different server queue manager that hosts a different instance of the target cluster queue. However, it cannot reallocate messages that are bound to the original server by affinity constraints, or messages that are in doubt because the server queue manager failed during end-of-batch processing. To process these messages, do the following:

1. Allocate a different cluster-receiver channel name and a different TCP/IP port to each z/OS queue manager. Each queue manager needs a different port so that two systems can share a single TCP/IP stack on a z/OS image. One of these is the queue manager originally running on that z/OS image, and the other is the queue manager that ARM will restart on that z/OS image following a system failure. Configure each port on each z/OS image, so that ARM can restart any queue manager on any z/OS image.

2. Create a different channel initiator command input file (CSQINPX) for each queue manager and z/OS image combination, to be referenced during channel initiator startup.

   Each CSQINPX file must include a START LISTENER PORT(port) command specific to that queue manager, and an ALTER CHANNEL command for a cluster-receiver channel specific to that queue manager and z/OS image combination. The ALTER CHANNEL command needs to set the connection name to the TCP/IP name of the z/OS image on which it is restarted. It must include the port number specific to the restarted queue manager as part of the connection name.

   The start-up JCL of each queue manager can have a fixed data set name for this CSQINPX file, and each z/OS image must have a different version of each CSQINPX file on a non-shared DASD volume.

In the event of ARM restart, WebSphere MQ advertises the changed channel definition to the cluster repository, which in turn publishes it to all the client queue managers that have expressed an interest in the server queue manager.

The client queue manager sees the server queue manager failure as a channel failure, and tries to restart the failed channel. When the client queue manager learns the new server connection-name, the channel restart reconnects the client queue manager to the restarted server queue manager. The client queue manager can then resynchronize its messages, resolve any in-doubt messages on the client queue manager's transmission queue, and normal processing can continue.

When connecting to a queue-sharing group:

When connecting to a queue-sharing group through a TCP/IP dynamic Domain Name System (DNS) logical group name, the connection name in your channel definition specifies the logical group name of your queue-sharing group, not the hostname or IP address of a physical machine. When this channel starts, it connects to the dynamic DNS and is then connected to one of the queue managers in the queue-sharing group. This process is explained in the WebSphere MQ Intercommunication manual.

In the unlikely event of an image failure, one of the following occurs:

* The queue managers on the failing image deregister from the dynamic DNS running on your sysplex. The channel responds to the connection failure by entering RETRYING state and then connects to the dynamic DNS running on the sysplex. The dynamic DNS allocates the inbound request to one of the remaining members of the queue-sharing group that is still running on the remaining images.
* If no other queue manager in the queue-sharing group is active and ARM restarts the queue manager and channel initiator on a different image, the group listener registers with dynamic DNS from this new image. This means that the logical group name (from the connection name field of the channel) connects to the dynamic DNS and is then connected to the same queue manager, now running on a different image. No change was required to the channel definition.

For this type of transparent recovery to occur, the following points must be noted:

* On z/OS, the dynamic DNS runs on one of the z/OS images in the sysplex. If this image were to fail, the dynamic DNS needs to be configured so that there is a secondary name server active in the sysplex, acting as an alternative to the primary name server. Information about primary and secondary dynamic DNS servers can be found in the *OS/390 SecureWay CS IP Configuration* manual.
* The TCP/IP group listener might have been started on a particular IP address that might not be available on this z/OS image. If this is the case, the listener might need to be started on a different IP address on the new image. If you are using virtual IP addresses, you can configure these to recover on other z/OS images so that no change to the START LISTENER command is required.

Recovering units of work manually

This chapter discusses the following topics:

* "Displaying connections and threads"
* "Recovering CICS units of recovery manually" on page 124
* "Recovering IMS units of recovery manually" on page 128
* "Recovering RRS units of recovery manually" on page 130
* "Recovering units of recovery on another queue manager in the queue-sharing group" on page 131

Displaying connections and threads

You can use the DISPLAY CONN command to get information about connections to queue managers and their associated units of work. You can display active units of work to see what is currently happening, or to see what needs to be terminated to allow the queue manager to shut down, and you can display unresolved units of work to help with recovery.

For more information see WebSphere MQ Script (MQSC) Command Reference.

Active units of work

To display only active units of work, use

```
DISPLAY CONN(*) WHERE(UOWSTATE EQ ACTIVE)
```

Unresolved units of work

An unresolved unit of work, also known as an "in-doubt thread", is one that is in the second pass of the two-phase commit operation. Resources are held in WebSphere MQ on its behalf. To display unresolved units of work, use

```
DISPLAY CONN(*) WHERE(UOWSTATE EQ UNRESOLVED)
```

External intervention is needed to resolve the status of unresolved units of work. This might only involve starting the recovery coordinator (CICS, IMS, or RRS) or might involve more, as described in the following sections.

Recovering CICS units of recovery manually

This section describes what happens when the CICS adapter restarts, and then explains how to deal with any unresolved units of recovery that arise.

What happens when the CICS adapter restarts

For background information, see the WebSphere MQ for z/OS Concepts and Planning Guide.

Whenever a connection is broken, the adapter has to go through a *restart phase* during the *reconnect process*. The restart phase resynchronizes resources. Resynchronization between CICS and WebSphere MQ enables in-doubt units of work to be identified and resolved.

Resynchronization can be caused by:
- An explicit request from the distributed queuing component
- An implicit request when a connection is made to WebSphere MQ

If the resynchronization is caused by connecting to WebSphere MQ, the sequence of events is:
1. The connection process gets a list of unit of work (UOW) IDs that WebSphere MQ thinks are in doubt.
2. The UOW IDs are displayed on the console in CSQC313I messages.
3. The UOW IDs are passed to CICS.
4. CICS initiates a resynchronization task (CRSY) for each in-doubt UOW ID.
5. The result of the task for each in-doubt UOW is displayed on the console.

You need to check the messages that are displayed during the connect process:

CSQC313I
> Shows that a UOW is in doubt.

CSQC400I
> Identifies the UOW and is followed by one of these messages:
> - CSQC402I or CSQC403I shows that the UOW was resolved successfully (committed or backed out).
> - CSQC404E, CSQC405E, CSQC406E, or CSQC407E shows that the UOW was not resolved.

CSQC409I
> Shows that all UOWs were resolved successfully.

CSQC408I
> Shows that not all UOWs were resolved successfully.

CSQC314I
> Warns that UOW IDs highlighted with a * are not resolved automatically. These UOWs must be resolved explicitly by the distributed queuing component when it is restarted.

Figure 50 shows an example set of restart messages displayed on the z/OS console.

```
CSQ9022I +CSQ1 CSQYASCP ' START QMGR' NORMAL COMPLETION
+CSQC323I VICIC1 CSQCQCON CONNECT received from TERMID=PB62 TRANID=CKCN
+CSQC303I VICIC1 CSQCCON CSQCSERV loaded. Entry point is 850E8918
+CSQC313I VICIC1 CSQCCON  UOWID=VICIC1.A6E5A6F0E2178D25 is in doubt
+CSQC313I VICIC1 CSQCCON  UOWID=VICIC1.A6E5A6F055B2AC25 is in doubt
+CSQC313I VICIC1 CSQCCON  UOWID=VICIC1.A6E5A6EFFD60D425 is in doubt
+CSQC313I VICIC1 CSQCCON  UOWID=VICIC1.A6E5A6F07AB56D22 is in doubt
+CSQC307I VICIC1 CSQCCON Successful connection to subsystem VC2
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008BAD18) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008BAA10) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008BA708) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008CAE88) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008CAB80) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008CA878) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008CA570) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008CA268) connect
successful
+CSQC403I VICIC1 CSQCTRUE Resolved BACKOUT for
+CSQC400I VICIC1 CSQCTRUE UOWID=VICIC1.A6E5A6F0E2178D25
+CSQC403I VICIC1 CSQCTRUE Resolved BACKOUT for
+CSQC400I VICIC1 CSQCTRUE UOWID=VICIC1.A6E5A6F055B2AC25
+CSQC403I VICIC1 CSQCTRUE Resolved BACKOUT for
+CSQC400I VICIC1 CSQCTRUE UOWID=VICIC1.A6E5A6F07AB56D22
+CSQC403I VICIC1 CSQCTRUE Resolved BACKOUT for
+CSQC400I VICIC1 CSQCTRUE UOWID=VICIC1.A6E5A6EFFD60D425
+CSQC409I VICIC1 CSQCTRUE Resynchronization completed successfully
```

Figure 50. Example restart messages

> The total number of CSQC313I messages should equal the total number of CSQC402I plus CSQC403I messages. If the totals are not equal, there are UOWs that the connection process cannot resolve. Those UOWs that cannot be resolved are caused by problems with CICS (for example, a cold start) or with WebSphere MQ, or by distributing queuing. When these problems have been fixed, you can initiate another resynchronization by disconnecting and then reconnecting.

> Alternatively, you can resolve each outstanding UOW yourself using the RESOLVE INDOUBT command and the UOW ID shown in message CSQC400I. You must then initiate a disconnect and a connect to clean up the *unit of recovery descriptors* in CICS. You need to know the correct outcome of the UOW to resolve UOWs manually.

All messages that are associated with unresolved UOWs are locked by WebSphere MQ and no Batch, TSO, or CICS task can access them.

If CICS fails and an emergency restart is necessary, *do not* vary the GENERIC APPLID of the CICS system. If you do and then reconnect to WebSphere MQ, data integrity with WebSphere MQ cannot be guaranteed. This is because WebSphere MQ treats the new instance of CICS as a different CICS (because the APPLID is different). In-doubt resolution is then based on the wrong CICS log.

How to resolve CICS units of recovery manually

If the adapter ends abnormally, CICS and WebSphere MQ build in-doubt lists either dynamically or during restart, depending on which subsystem caused the abend.

Note: If you use the DFH$INDB sample program to show units of work, you might find that it does not always show WebSphere MQ UOWs correctly.

When CICS connects to WebSphere MQ, there might be one or more units of recovery that have not been resolved.

One of the following messages is sent to the console:
- CSQC404E
- CSQC405E
- CSQC406E
- CSQC407E
- CSQC408I

For details of what these messages mean, see the WebSphere MQ for z/OS Messages and Codes manual.

CICS retains details of units of recovery that were not resolved during connection startup. An entry is purged when it no longer appears on the list presented by WebSphere MQ.

Any units of recovery that CICS cannot resolve must be resolved manually using WebSphere MQ commands. This manual procedure is rarely used within an installation, because it is required only where operational errors or software problems have prevented automatic resolution. *Any inconsistencies found during in-doubt resolution must be investigated.*

To resolve the units of recovery:
1. Obtain a list of the units of recovery from WebSphere MQ using the following command:

```
+CSQ1 DISPLAY CONN(*) WHERE(UOWSTATE EQ UNRESOLVED)
```

You receive the following message:

```
CSQM201I +CSQ1 CSQMDRTC  DISPLAY CONN DETAILS
CONN(BC85772CBE3E0001)
EXTCONN(C3E2D8C3C7D9F0F94040404040404040)
TYPE(CONN)
CONNOPTS(
 MQCNO_STANDARD_BINDING
 )
UOWLOGDA(2005-02-04)
UOWLOGTI(10.17.44)
UOWSTDA(2005-02-04)
UOWSTTI(10.17.44)
UOWSTATE(UNRESOLVED)
NID(IYRCSQ1 .BC8571519B60222D)
EXTURID(BC8571519B60222D)
QMURID(0000002BDA50)
URTYPE(CICS)
USERID(MQTEST)
APPLTAG(IYRCSQ1)
ASID(0000)
APPLTYPE(CICS)
TRANSID(GP02)
TASKNO(0000096)
 END CONN DETAILS
```

For CICS connections, the NID consists of the CICS applid and a unique
number provided by CICS at the time the syncpoint log entries are written.
This unique number is stored in records written to both the CICS system log
and the WebSphere MQ log at syncpoint processing time. This value is referred
to in CICS as the *recovery token*.

2. Scan the CICS log for entries related to a particular unit of recovery.

   Look for a PREPARE record for the task-related installation where the recovery
   token field (JCSRMTKN) equals the value obtained from the network ID. The
   network ID is supplied by WebSphere MQ in the DISPLAY CONN command
   output.

   The PREPARE record in the CICS log for the units of recovery provides the
   CICS task number. All other entries on the log for this CICS task can be located
   using this number.

   You can use the CICS journal print utility DFHJUP when scanning the log. For
   details of using this program, see the *CICS Operations and Utilities Guide*.

3. Scan the WebSphere MQ log for records with the NID related to a particular
   unit of recovery. Then use the URID from this record to obtain the rest of the
   log records for this unit of recovery.

   When scanning the WebSphere MQ log, note that the WebSphere MQ startup
   message CSQJ001I provides the start RBA for this session.

   The print log records program (CSQ1LOGP) can be used for that purpose.

4. If you need to, do in-doubt resolution in WebSphere MQ.

   WebSphere MQ can be directed to take the recovery action for a unit of
   recovery using a WebSphere MQ RESOLVE INDOUBT command.

   For information about RESOLVE INDOUBT, see the WebSphere MQ Script
   (MQSC) Command Reference manual.

   To recover all threads associated with a specific *connection-name*, use the NID(*)
   option.

   The command produces one of the following messages showing whether the
   thread is committed or backed out:

```
CSQV414I +CSQ1 THREAD network-id COMMIT SCHEDULED
CSQV415I +CSQ1 THREAD network-id ABORT SCHEDULED
```

When performing in-doubt resolution, CICS and the adapter are not aware of the commands to WebSphere MQ to commit or back out units of recovery, because only WebSphere MQ resources are affected. However, CICS keeps details about the in-doubt threads that could not be resolved by WebSphere MQ. This information is purged either when the list presented is empty, or when the list does not include a unit of recovery of which CICS has details.

Recovering IMS units of recovery manually

This section describes what happens when the IMS adapter restarts, and then explains how to deal with any unresolved units of recovery that arise.

What happens when the IMS adapter restarts

For background information, see the WebSphere MQ for z/OS Concepts and Planning Guide.

Whenever the connection to WebSphere MQ is restarted, either following a queue manager restart or an IMS /START SUBSYS command, IMS initiates the following resynchronization process:

1. IMS presents the list of unit of work (UOW) IDs that it believes are in doubt to the WebSphere MQ IMS adapter one at a time with a resolution parameter of Commit or Backout.
2. The IMS adapter passes the resolution request to WebSphere MQ and reports the result back to IMS.
3. Having processed all the IMS resolution requests, the IMS adapter gets from WebSphere MQ a list of all UOWs that WebSphere MQ still holds in doubt that were initiated by the IMS system. These are reported to the IMS master terminal in message CSQQ008I.

Note: While a UOW is in doubt, any associated WebSphere MQ message is locked by WebSphere MQ and is not available to any application.

How to resolve IMS units of recovery manually

When IMS connects to WebSphere MQ, WebSphere MQ might have one or more in-doubt units of recovery that have not been resolved.

If WebSphere MQ has in-doubt units of recovery that IMS did not resolve, the following message is issued at the IMS master terminal:

```
CSQQ008I  nn units of recovery are still in doubt in queue manager qmgr-name
```

If this message is issued, IMS was either cold-started or it was started with an incomplete log tape. This message can also be issued if WebSphere MQ or IMS terminates abnormally because of a software error or other subsystem failure.

After receiving the CSQQ008I message:
• The connection remains active.

- IMS applications can still access WebSphere MQ resources.
- Some WebSphere MQ resources remain locked out.

If the in-doubt thread is not resolved, IMS message queues can start to build up. If the IMS queues fill to capacity, IMS terminates. You must be aware of this potential difficulty, and you must monitor IMS until the in-doubt units of recovery are fully resolved.

Recovery procedure:

Use the following procedure to recover the IMS units of work:

1. Force the IMS log closed, using /SWI OLDS, and then archive the IMS log. Use the utility, DFSERA10, to print the records from the previous IMS log tape. Type X'3730' log records indicate a phase-2 commit request and type X'38' log records indicate an abort request. Record the requested action for the last transaction in each dependent region.
2. Run the DL/I batch job to back out each PSB involved that has not reached a commit point. The process might take some time because transactions are still being processed. It might also lock up a number of records, which could impact the rest of the processing and the rest of the message queues.
3. Produce a list of the in-doubt units of recovery from WebSphere MQ using the following command:

```
+CSQ1 DISPLAY CONN(*) WHERE(UOWSTATE EQ UNRESOLVED)
```

You receive the following message:

```
CSQM201I +CSQ1 CSQMDRTC  DISPLAY CONN DETAILS
CONN(BC45A794C4290001)
EXTCONN(C3E2D8C3E2C5C3F240404040404040404040)
TYPE(CONN)
CONNOPTS(
 MQCNO_STANDARD_BINDING
 )
UOWLOGDA(2005-02-15)
UOWLOGTI(16.39.43)
UOWSTDA(2005-02-15)
UOWSTTI(16.39.43)
UOWSTATE(UNRESOLVED)
NID(IM8F     .BC45A794D3810344)
EXTURID(
 0000052900000000
 )
QMURID(00000354B76E)
URTYPE(IMS)
USERID(STCPI)
APPLTAG(IM8F)
ASID(0000)
APPLTYPE(IMS)
PSTID(0004)
PSBNAME(GP01MPP)
```

For IMS, the NID consists of the IMS connection name and a unique number provided by IMS. The value is referred to in IMS as the *recovery token*. For more information, see the *IMS Customization Guide*.

4. Compare the NIDs (IMSID plus OASN in hexadecimal) displayed in the DISPLAY THREAD messages with the OASNs (4 bytes decimal) shown in the DFSERA10 output. Decide whether to commit or back out.

5. Perform in-doubt resolution in WebSphere MQ with the RESOLVE INDOUBT command, as follows:

```
RESOLVE INDOUBT(connection-name)
       ACTION(COMMIT|BACKOUT)
       NID(network-id)
```

For information about the RESOLVE INDOUBT command, see the WebSphere MQ Script (MQSC) Command Reference manual.

To recover all threads associated with *connection-name*, use the NID(*) option. The command results in one of the following messages to indicate whether the thread is committed or backed out:

```
CSQV414I   THREAD network-id COMMIT SCHEDULED
CSQV415I   THREAD network-id BACKOUT SCHEDULED
```

When performing in-doubt resolution, IMS and the adapter are not aware of the commands to WebSphere MQ to commit or back out in-doubt units of recovery because only WebSphere MQ resources are affected.

Recovering RRS units of recovery manually

When RRS connects to WebSphere MQ, WebSphere MQ might have one or more in-doubt units of recovery that have not been resolved. If WebSphere MQ has in-doubt units of recovery that RRS did not resolve, one of the following messages is issued at the z/OS console:

- CSQ3011I
- CSQ3013I
- CSQ3014I
- CSQ3016I

Both WebSphere MQ and RRS provide tools to display information about in-doubt units of recovery, and techniques for manually resolving them.

In WebSphere MQ, use the DISPLAY CONN command to display information about in-doubt WebSphere MQ threads. The output from the command includes RRS unit of recovery IDs for those WebSphere MQ threads that have RRS as a coordinator. This can be used to determine the outcome of the unit of recovery.

Use the RESOLVE INDOUBT command to resolve the WebSphere MQ in-doubt thread manually. This command can be used to either commit or back out the unit of recovery after you have determined what the correct decision is.

Recovering units of recovery on another queue manager in the queue-sharing group

If a queue manager that is a member of a queue-sharing group fails and cannot be restarted, other queue managers in the group can perform peer recovery, and take over from it. However, the queue manager might have in-doubt units of recovery that cannot be resolved by peer recovery because the final disposition of that unit of recovery is known only to the failed queue manager. These units of recovery will be resolved when the queue manager is eventually restarted, but until then, they remain in doubt.

This means that certain resources (for example, messages) might be locked, making them unavailable to other queue managers in the group. In this situation, you can use the DISPLAY THREAD command to display these units of work on the inactive queue manager. If you want to resolve these units of recovery manually to make the messages available to other queue managers in the group, you can use the RESOLVE INDOUBT command.

When you issue the DISPLAY THREAD command to display units of recovery that are in doubt, you can use the QMNAME keyword to specify the name of the inactive queue manager. For example, if you issue the following command:

```
+CSQ1 DISPLAY THREAD(*) TYPE(INDOUBT) QMNAME(QM01)
```

You receive the following messages:

```
CSQV436I +CSQ1 INDOUBT THREADS FOR QM01 -
NAME     THREAD-XREF        URID NID
USER1    00000000000000000000000 CSQ:0001.0
USER2    00000000000000000000000 CSQ:0002.0
 DISPLAY THREAD REPORT COMPLETE
```

If the queue manager specified is active, WebSphere MQ does not return information about in-doubt threads, but issues the following message:

```
CSQV435I CANNOT USE QMNAME KEYWORD, QM01 IS ACTIVE
```

Use the WebSphere MQ command RESOLVE INDOUBT to resolve the in-doubt threads manually. Use the QMNAME keyword to specify the name of the inactive queue manager in the command.

This command can be used to commit or back out the unit of recovery. The command resolves the shared portion of the unit of recovery only; any local messages are unaffected and remain locked until the queue manager restarts, or reconnects to CICS, IMS, or RRS batch.

Example recovery scenarios

This chapter describes procedures for recovering WebSphere MQ after various error conditions. These error conditions are grouped in the following categories:

Table 4. Example recovery scenarios

Problem category	Problem	Where to look next
Shared queue problems	Queue both private and shared.	"Shared queue problems"
Active log problems	• Dual logging is lost. • Active log has stopped. • One or both copies of the active log data set are damaged. • Write errors on active log data set. • Active log is becoming full or is full. • Read errors on active log data set.	"Active log problems" on page 133
Archive log problems	• Insufficient DASD space to complete off-loading active log data sets. • Off-load task has terminated abnormally. • Archive data set allocation problem. • Read I/O errors on the archive data set during restart.	"Archive log problems" on page 139
BSDS problems	• Error opening BSDS. • Log content does not correspond with BSDS information. • Both copies of the BSDS are damaged. • Unequal time stamps. • Dual BSDS data sets are out of synchronization. • I/O error on BSDS.	"BSDS problems" on page 142
Page set problems	• Page set full. • A page set has an I/O error.	"Page set problems" on page 146
Coupling Facility and DB2 problems	• Storage medium full. • DB2 system fails. • DB2 data-sharing group fails. • DB2 and the Coupling Facility fail.	"Coupling Facility and DB2 problems" on page 147
Unit of work problems	A long-running unit of work is encountered.	"Problems with long-running units of work" on page 150
IMS problems	• An IMS application terminates abnormally. • The IMS adapter cannot connect to WebSphere MQ. • IMS not operational.	"IMS-related problems" on page 151

Shared queue problems

This section explains what to do if a queue is found to be both private and shared.

Queue is both private and shared

Symptoms

WebSphere MQ issues the following message:

```
CSQI063E +CSQ1 QUEUE queue-name IS BOTH PRIVATE AND SHARED
```

During queue manager restart, WebSphere MQ discovered that a page set based queue and a shared queue of the same name coexist.

System action
Once restart processing has completed, any **MQOPEN** request to that queue name fails, indicating the coexistence problem.

System programmer action
None.

Operator action
Delete one version of the queue to allow processing of that queue name. If there are messages on the queue that must be kept, you can use the MOVE QLOCAL command to move them to the other queue.

Active log problems

This section covers the following active log problems:
- "Dual logging is lost"
- "Active log stopped"
- "One or both copies of the active log data set are damaged" on page 134
- "Write I/O errors on an active log data set" on page 135
- "I/O errors occur while reading the active log" on page 136
- "Active log is becoming full" on page 137

Dual logging is lost

Symptoms
WebSphere MQ issues the following message:

```
CSQJ004I +CSQ1 ACTIVE LOG COPY n INACTIVE, LOG IN SINGLE MODE,
          ENDRBA=...
```

Having completed one active log data set, WebSphere MQ found that the subsequent (COPY n) data sets were not off-loaded or were marked stopped.

System action
WebSphere MQ continues in single mode until off-loading has been completed, then returns to dual mode.

System programmer action
None.

Operator action
Check that the off-load is proceeding and is not waiting for a tape mount. You might need to run the print log map utility to determine the state of all data sets. You might also need to define additional data sets.

Active log stopped

Symptoms
WebSphere MQ issues the following message:

```
CSQJ030E +CSQ1 RBA RANGE startrba TO endrba NOT AVAILABLE IN ACTIVE
            LOG DATA SETS
```

System action

The active log data sets that contain the RBA range reported in message CSQJ030E are unavailable to WebSphere MQ. The status of these logs is STOPPED in the BSDS. The queue manager terminates with a dump.

System programmer action

You must resolve this problem before restarting the queue manager. The log RBA range must be available for WebSphere MQ to be recoverable. An active log that is marked as STOPPED in the BSDS will never be reused or archived and this creates a hole in the log.

Look for messages that indicate why the log data set has stopped, and follow the instructions for those messages.

Modify the BSDS active log inventory to reset the STOPPED status. To do this, follow this procedure after the queue manager has terminated:

1. Use the print log utility (CSQJU004) to obtain a copy of the BSDS log inventory. This shows the status of the log data sets.

2. Use the DELETE function of the change log inventory utility (CSQJU003) to delete the active log data sets that are marked as STOPPED.

3. Use the NEWLOG function of CSQJU003 to add the active logs back into the BSDS inventory. The starting and ending RBA for each active log data set must be specified on the NEWLOG statement. (The correct values to use can be found from the print log utility report obtained in Step 1.)

4. Rerun CSQJU004. The active log data sets that were marked as STOPPED are now shown as NEW and NOT REUSABLE. These active logs will be archived in due course.

5. Restart the queue manager.

Note: If your queue manager is running in dual BSDS mode, you must update both BSDS inventories.

One or both copies of the active log data set are damaged

Symptoms

WebSphere MQ issues the following messages:

```
CSQJ102E +CSQ1 LOG RBA CONTENT OF LOG DATA SET DSNAME=...,
            STARTRBA=..., ENDRBA=...,
            DOES NOT AGREE WITH BSDS INFORMATION
CSQJ232E +CSQ1 OUTPUT DATA SET CONTROL INITIALIZATION PROCESS FAILED
```

System action

Queue manager startup processing is terminated.

System programmer action

If one copy of the data set is damaged, carry out these steps:

1. Rename the damaged active log data set and define a replacement data set.

2. Copy the undamaged data set to the replacement data set.
3. Use the change log inventory utility to:
   - Remove information relating to the damaged data set from the BSDS.
   - Add information relating to the replacement data set to the BSDS.
4. Restart the queue manager.

If both copies of the active log data sets are damaged, the current page sets are available, **and the queue manager shut down cleanly**, carry out these steps:

1. Rename the damaged active log data sets and define replacement data sets.
2. Use the change log records utility to:
   - Remove information relating to the damaged data set from the BSDS.
   - Add information relating to the replacement data set to the BSDS.
3. Rename the current page sets and define replacement page sets.
4. Use CSQUTIL (FORMAT and RESETPAGE) to format the replacement page sets and copy the renamed page sets to them. The RESETPAGE function also resets the log information in the replacement page sets.

If the queue manager did not shut down cleanly, you must either restore your system from a previous known point of consistency, or perform a cold start (described in "Reinitializing a queue manager" on page 116).

Operator action
> None.

Write I/O errors on an active log data set

Symptoms
> WebSphere MQ issues the following message:

```
CSQJ105E +CSQ1 csect-name LOG WRITE ERROR DSNAME=...,
         LOGRBA=..., ERROR STATUS=ccccffss
```

System action
> WebSphere MQ carries out these steps:
> 1. Marks the log data set that has the error as TRUNCATED in the BSDS.
> 2. Goes on to the next available data set.
> 3. If dual active logging is used, truncates the other copy at the same point.
>
> The data in the truncated data set is off-loaded later, as usual.
>
> The data set will be reused on the next cycle.

System programmer action
> None.

Operator action
> If errors on this data set still exist, shut down the queue manager after the next off-load. Then use Access Method Services (AMS) and the change log inventory utility to add a replacement. (For instructions, see "Changing the BSDS" on page 83.)

I/O errors occur while reading the active log

Symptoms

WebSphere MQ issues the following message:

```
CSQJ106E +CSQ1 LOG READ ERROR DSNAME=..., LOGRBA=...,
          ERROR STATUS=ccccffss
```

System action

This depends on when the error occurred:

- If the error occurs during the off-load process, the process tries to read the RBA range from a second copy.
  - If no second copy exists, the active log data set is stopped.
  - If the second copy also has an error, only the original data set that triggered the off-load is stopped. The archive log data set is then terminated, leaving a gap in the archived log RBA range.
  - This message is issued:

```
CSQJ124E +CSQ1 OFFLOAD OF ACTIVE LOG SUSPENDED FROM
          RBA xxxxxx TO RBA xxxxxx DUE TO I/O ERROR
```

  - If the second copy is satisfactory, the first copy is not stopped.
- If the error occurs during recovery, WebSphere MQ provides data from specific log RBAs requested from another copy or archive. If this is unsuccessful, recovery does not succeed, and the queue manager terminates abnormally.
- If the error occurs during restart, if dual logging is used, WebSphere MQ continues with the alternative log data set, otherwise the queue manager ends abnormally.

System programmer action

Look for system messages, such as IEC prefixed messages, and try to resolve the problem using the recommended actions for these messages.

If the active log data set has been stopped, it is not used for logging. The data set is not deallocated; it is still used for reading. Even if the data set is not stopped, an active log data set that gives persistent errors should be replaced.

Operator action

None.

Replacing the data set:

How you replace the data set depends on whether you are using single or dual active logging.

If you are using dual active logging:

1. Ensure that the data has been saved.

   The data is saved on the other active log and this can be copied to a replacement active log.

2. Stop the queue manager and delete the data set with the error using Access Method Services.

3. Redefine a new log data set using Access Method Services DEFINE so that you can write to it. Use DFDSS or Access Method Services REPRO to copy the good log into the redefined data set so that you have two consistent, correct logs again.

4. Use the change log inventory utility, CSQJU003, to update the information in the BSDS about the corrupt data set as follows:

   a. Use the DELETE function to remove information about the corrupt data set.

   b. Use the NEWLOG function to name the new data set as the new active log data set and give it the RBA range that was successfully copied.

      You can run the DELETE and NEWLOG functions in the same job step. Put the DELETE statement before NEWLOG statement in the SYSIN input data set.

5. Restart the queue manager.

If you are using single active logging:

1. Ensure that the data has been saved.

2. Stop the queue manager.

3. Determine whether the data set with the error has been off-loaded:

   a. Use the CSQJU003 utility to list information about the archive log data sets from the BSDS.

   b. Search the list for a data set whose RBA range includes the RBA of the corrupt data set.

4. If the corrupt data set has been off-loaded, copy its backup in the archive log to a new data set. Then, skip to step 6.

5. If an active log data set is stopped, an RBA is not off-loaded. Use DFDSS or Access Method Services REPRO to copy the data from the corrupt data set to a new data set.

   If further I/O errors prevent you from copying the entire data set, a gap occurs in the log.

   Note: Queue manager restart will not be successful if a gap in the log is detected.

6. Use the change log inventory utility, CSQJU003, to update the information in the BSDS about the corrupt data set as follows:

   a. Use the DELETE function to remove information about the corrupt data set.

   b. Use the NEWLOG function to name the new data set as the new active log data set and to give it the RBA range that was successfully copied.

      The DELETE and NEWLOG functions can be run in the same job step. Put the DELETE statement before NEWLOG statement in the SYSIN input data set.

7. Restart the queue manager.

Active log is becoming full

The active log can fill up for several reasons, for example, delays in off-loading and excessive logging. If an active log runs out of space, this has serious consequences. When the active log becomes full, the queue manager halts processing until an off-load has been completed. If the off-load processing stops when the active log is full, the queue manager can end abnormally. Corrective action is required before the queue manager can be restarted.

Symptoms

Because of the serious implications of an active log becoming full, the queue manager issues the following warning message when the last available active log data set is 5% full:

```
CSQJ110E +CSQ1 LAST COPYn ACTIVE LOG DATA SET IS nnn PERCENT FULL
```

and reissues the message after each additional 5% of the data set space is filled. Each time the message is issued, the off-load process is started.

System action

Messages are issued and off-load processing started, as detailed above. If the active log becomes full, further actions are taken. See "Active log is full"

System programmer action

Use the DEFINE LOG command to dynamically add further active log data sets. This permits WebSphere MQ to continue its normal operation while the error causing the off-load problems is corrected. For more information about the DEFINE LOG command, see WebSphere MQ Script (MQSC) Command Reference.

Active log is full

Symptoms

When the active log becomes full, the queue manager halts processing until an off-load has been completed. If the off-load processing stops when the active log is full, the queue manager can end abnormally. Corrective action is required before the queue manager can be restarted.

WebSphere MQ issues the following message:

```
CSQJ111A +CSQ1 OUT OF SPACE IN ACTIVE LOG DATA SETS
```

and an off-load is started. The queue manager then halts processing until the off-load has been completed.

System action

WebSphere MQ waits for an available active log data set before resuming normal WebSphere MQ processing. Normal shutdown, with either QUIESCE or FORCE, is not possible because the shutdown sequence requires log space to record system events related to shutdown (for example, checkpoint records). If the off-load processing stops when the active log is full, the queue manager stops with an X'6C6' abend; restart in this case requires special attention. For more details, see the WebSphere MQ for z/OS Problem Determination Guide.

System programmer action

You can provide additional active log data sets before restarting the queue manager. This permits WebSphere MQ to continue its normal operation while the error causing the off-load problems is corrected. To add new active log data sets, use the change log inventory utility (CSQJU003) when the queue manager is not active. For more details about adding new active log data sets, see "Changing the BSDS" on page 83.

Consider increasing the number of logs by:

1. Making sure the queue manager is stopped, then using the Access Method Services DEFINE command to define a new active log data set.
2. Defining the new active log data set in the BSDS using the change log inventory utility (CSQJU003).

When you restart the queue manager, off-load starts automatically during startup, and any work that was in progress when WebSphere MQ was forced to stop is recovered.

Operator action
Check whether the off-load process is waiting for a tape drive. If it is, mount the tape. If you cannot mount the tape, force WebSphere MQ to stop by using the z/OS CANCEL command.

Archive log problems

This section covers the following archive log problems:
- "Allocation problems"
- "Off-load task terminated abnormally" on page 140
- "Insufficient DASD space to complete off-load processing" on page 140
- "Read I/O errors on the archive data set while WebSphere MQ is restarting" on page 141

Allocation problems

Symptoms
WebSphere MQ issues the following message:

```
CSQJ103E +CSQ1 LOG ALLOCATION ERROR DSNAME=dsname,
          ERROR STATUS=eeeeiiii, SMS REASON CODE=sss
```

z/OS dynamic allocation provides the ERROR STATUS. If the allocation was for off-load processing, the following message is also displayed:

```
CSQJ115E +CSQ1 OFFLOAD FAILED, COULD NOT ALLOCATE AN ARCHIVE
          DATA SET
```

System action
The following actions take place:
- If the input is needed for recovery, recovery is not successful, and the queue manager ends abnormally.
- If the active log had become full and an off-load was scheduled but not completed, off-load tries again the next time it is triggered. The active log does not reuse a data set that has not yet been archived.

System programmer action
None.

Operator action
Check the allocation error code for the cause of the problem, and correct it. Ensure that drives are available, and either restart or wait for the off-load to be retried. Be careful if a DFP/DFSMS ACS user-exit filter has been

written for an archive log data set, because this can cause a device allocation error when the queue manager tries to read the archive log data set.

Off-load task terminated abnormally

Symptoms

No specific WebSphere MQ message is issued for write I/O errors.

Only a z/OS error recovery program message appears. If you get WebSphere MQ message CSQJ128E, the off-load task has terminated abnormally, and you should consult the WebSphere MQ for z/OS Messages and Codes manual.

System action

The following actions take place:

- Off-load abandons the output data set; no entry is made in the BSDS.
- Off-load dynamically allocates a new archive and restarts off-loading from the point at which it was previously triggered.
- If an error occurs on the new data set:
  - In dual archive mode, the following message is generated and the off-load processing changes to single mode:

```
CSQJ114I +CSQ1 ERROR ON ARCHIVE DATA SET, OFFLOAD
         CONTINUING WITH ONLY ONE ARCHIVE DATA SET BEING
         GENERATED
```

  - In single archive mode, the output data set is abandoned. Another attempt to off-load this RBA range is made the next time off-load is triggered.
  - The active log does not wrap around; if there are no more active logs, data is not lost.

System programmer action

None.

Operator action

Ensure that off-load is allocated on a reliable drive and control unit.

Insufficient DASD space to complete off-load processing

Symptoms

While off-loading the active log data sets to DASD, the process terminates unexpectedly. WebSphere MQ issues the following message:

```
CSQJ128E +CSQ1 LOG OFF-LOAD TASK FAILED FOR ACTIVE LOG nnnnn
```

The error is preceded by z/OS messages IEC030I, IEC031I, or IEC032I.

System action

WebSphere MQ de-allocates the data set on which the error occurred. If WebSphere MQ is running in dual archive mode, WebSphere MQ changes to single archive mode and continues the off-load. If the off-load cannot be completed in single archive mode, the active log data sets cannot be off-loaded, and the state of the active log data sets remains NOT

REUSABLE. Another attempt to off-load the RBA range of the abandoned active log data sets is made the next time the off-load process is triggered.

System programmer action
Quiesce the queue manager (using the WebSphere MQ command STOP QMGR MODE(QUIESCE)) to restrict logging activity until the z/OS abend is resolved.

The most likely causes of these symptoms are:
- The size of the archive log data set is too small to contain the data from the active log data sets during off-load processing. All the secondary space allocations have been used. This condition is normally accompanied by z/OS message IEC030I.

  To solve the problem, either increase the primary or secondary allocations (or both) for the archive log data set (in the CSQ6ARVP system parameters), or reduce the size of the active log data set. If the data to be off-loaded is particularly large, you can mount another online storage volume or make one available to WebSphere MQ.
- All available space on the DASD volumes to which the archive data set is being written has been exhausted. This condition is normally accompanied by z/OS message IEC032I.

  To solve the problem, make more space available on the DASD volumes, or make another online storage volume available for WebSphere MQ.
- The primary space allocation for the archive log data set (as specified in the CSQ6ARVP system parameters) is too large to allocate to any available online DASD device. This condition is normally accompanied by z/OS message IEC032I.

  To solve the problem, make more space available on the DASD volumes, or make another online storage volume available for WebSphere MQ. If this is not possible, you must adjust the value of PRIQTY in the CSQ6ARVP system parameters to reduce the primary allocation. (For details, see the WebSphere MQ for z/OS System Setup Guide.)

  Note: If you reduce the primary allocation, you might have to increase the size of the secondary space allocation to avoid future abends.

Operator action
None.

Read I/O errors on the archive data set while WebSphere MQ is restarting

Symptoms
No specific WebSphere MQ message is issued; only the z/OS error recovery program message appears.

System action
This depends on whether a second copy exists:
- If a second copy exists, it is allocated and used.
- If a second copy does not exist, restart is not successful.

System programmer action
None.

Operator action
Try to restart, using a different drive.

BSDS problems

For background information about the bootstrap data set (BSDS), see the WebSphere MQ for z/OS Concepts and Planning Guide.

This section describes the following BSDS problems:
- "Error occurs while opening the BSDS"
- "Log content does not agree with the BSDS information" on page 143
- "Both copies of the BSDS are damaged" on page 143
- "Unequal time stamps" on page 144
- "Out of synchronization" on page 144
- "I/O error" on page 145

Normally, there are two copies of the BSDS, but if one is damaged, WebSphere MQ immediately changes to single BSDS mode. However, the damaged copy of the BSDS must be recovered before restart. If you are in single mode and damage the only copy of the BSDS, or if you are in dual mode and damage both copies, use the procedure described in "Recovering the BSDS" on page 85.

This section covers some of the BSDS problems that can occur at startup. Problems *not* covered here include:
- RECOVER BSDS command errors (messages CSQJ301E - CSQJ307I)
- Change log inventory utility errors (message CSQJ123E)
- Errors in the BSDS backup being dumped by off-load (message CSQJ125E)

For information about the above problems, see the WebSphere MQ for z/OS Messages and Codes manual.

Error occurs while opening the BSDS

Symptoms

WebSphere MQ issues the following message:

```
CSQJ100E +CSQ1 ERROR OPENING BSDSn DSNAME=..., ERROR STATUS=eeii
```

where *eeii* is the VSAM return code. For information about VSAM codes, see the *DFSMS/MVS Macro Instructions for Data Sets* manual. For an explanation of this message, see the WebSphere MQ for z/OS Messages and Codes manual.

System action

During system initialization, the startup is terminated.

During a RECOVER BSDS command, the system continues in single BSDS mode.

System programmer action

None.

Operator action

Carry out these steps:
1. Run the print log map utility on both copies of the BSDS, and compare the lists to determine which copy is accurate or current.

2. Rename the data set that had the problem, and define a replacement for it.
3. Copy the accurate data set to the replacement data set, using Access Method Services.
4. Restart the queue manager.

Log content does not agree with the BSDS information

Symptoms
WebSphere MQ issues the following message:

```
CSQJ102E +CSQ1 LOG RBA CONTENT OF LOG DATA SET DSNAME=...,
           STARTRBA=..., ENDRBA=...,
           DOES NOT AGREE WITH BSDS INFORMATION
```

This message indicates that the change log inventory utility was used incorrectly or that a down-level data set is being used.

System action
Queue manager startup processing is terminated.

System programmer action
None.

Operator action
Run the print log map utility and the change log inventory utility to print and correct the contents of the BSDS.

Both copies of the BSDS are damaged

Symptoms
WebSphere MQ issues the following messages:

```
CSQJ107E +CSQ1 READ ERROR ON BSDS
           DSNAME=... ERROR STATUS=0874
CSQJ117E +CSQ1 REG8 INITIALIZATION ERROR READING BSDS
           DSNAME=... ERROR STATUS=0874
CSQJ119E +CSQ1 BOOTSTRAP ACCESS INITIALIZATION PROCESSING FAILED
```

System action
Queue manager startup processing is terminated.

System programmer action
Carry out these steps:
1. Rename the data set, and define a replacement for it.
2. Locate the BSDS associated with the most recent archive log data set, and copy it to the replacement data set.
3. Use the print log map utility to print the contents of the replacement BSDS.
4. Use the print log records utility to print a summary report of the active log data sets missing from the replacement BSDS, and to establish the RBA range.
5. Use the change log inventory utility to update the missing active log data set inventory in the replacement BSDS.

6. If dual BSDS data sets had been in use, copy the updated BSDS to the second copy of the BSDS.
7. Restart the queue manager.

Operator action
None.

Unequal time stamps

Symptoms
WebSphere MQ issues the following message:

```
CSQJ120E +CSQ1 DUAL BSDS DATA SETS HAVE UNEQUAL TIME STAMPS,
         SYSTEM BSDS1=...,BSDS2=...,
         UTILITY BSDS1=...,BSDS2=...
```

The possible causes are:
- One copy of the BSDS has been restored. All information on the restored BSDS is down-level. The down-level BSDS has the lower time stamp.
- One of the volumes containing the BSDS has been restored. All information on the restored volume is down-level. If the volume contains any active log data sets or WebSphere MQ data, they are also down-level. The down-level volume has the lower time stamp.
- Dual logging has degraded to single logging, and you are trying to start without recovering the damaged log.
- The queue manager terminated abnormally after updating one copy of the BSDS but before updating the second copy.

System action
WebSphere MQ attempts to resynchronize the BSDS data sets using the more recent copy. If this fails, queue manager startup is terminated.

System programmer action
None.

Operator action
If automatic resynchronization fails, carry out these steps:
1. Run the print log map utility on both copies of the BSDS, compare the lists to determine which copy is accurate or current.
2. Rename the down-level data set and define a replacement for it.
3. Copy the good data set to the replacement data set, using Access Method Services.
4. If applicable, determine whether the volume containing the down-level BSDS has been restored. If it has been restored, all data on that volume, such as the active log data, is also down-level.

   If the restored volume contains active log data and you were using dual active logs on separate volumes, you need to copy the current version of the active log to the down-level log data set. See "Recovering logs" on page 78 for details of how to do this.

Out of synchronization

Symptoms
WebSphere MQ issues the following message:

```
CSQJ122E +CSQ1 DUAL BSDS DATA SETS ARE OUT OF SYNCHRONIZATION
```

The system time stamps of the two data sets are identical. Differences can
exist if operator errors occurred while the change log inventory utility was
being used. (For example, the change log inventory utility was only run on
one copy.) The change log inventory utility sets a private time stamp in the
BSDS control record when it starts, and a close flag when it ends.
WebSphere MQ checks the change log inventory utility time stamps and, if
they are different, or they are the same but one close flag is not set,
WebSphere MQ compares the copies of the BSDSs. If the copies are
different, CSQJ122E is issued.

System action
> Queue manager startup is terminated.

System programmer action
> None.

Operator action
> Carry out these steps:
> 1. Run the print log map utility on both copies of the BSDS, and compare
>   the lists to determine which copy is accurate or current.
> 2. Rename the data set that had the problem, and define a replacement for
>   it.
> 3. Copy the accurate data set to the replacement data set, using access
>   method services.
> 4. Restart the queue manager.

I/O error

Symptoms
> WebSphere MQ changes to single BSDS mode and issues the user message:

```
CSQJ126E +CSQ1 BSDS ERROR FORCED SINGLE BSDS MODE
```

This is followed by one of the following messages:

```
CSQJ107E +CSQ1 READ ERROR ON BSDS
           DSNAME=... ERROR STATUS=...

CSQJ108E +CSQ1 WRITE ERROR ON BSDS
           DSNAME=... ERROR STATUS=...
```

System action
> The BSDS mode changes from dual to single.

System programmer action
> None.

Operator action
> Carry out these steps:

1. Use Access Method Services to rename or delete the damaged BSDS and to define a new BSDS with the same name as the BSDS that had the error. Example control statements can be found in job CSQ4BREC in thlqual.SCSQPROC.

2. Issue the WebSphere MQ command RECOVER BSDS to make a copy of the good BSDS in the newly allocated data set and reinstate dual BSDS mode. See also "Recovering the BSDS" on page 85.

Page set problems

This section covers the problems that you might encounter with page sets:

- "Page set I/O errors" describes what happens if a page set is damaged.
- "Page set full" on page 147 describes what happens if there is not enough space on the page set for any more MQI operations.

Page set I/O errors

Problem
> A page set has an I/O error.

Symptoms
> This message is issued:

```
CSQP004E +CSQ1 csect-name I/O ERROR STATUS ret-code
PSID psid RBA rba
```

System action
> The queue manager terminates abnormally.

System programmer action
> None.

Operator action
> Repair the I/O error cause.
>
> If none of the page sets are damaged, restart the queue manager. WebSphere MQ automatically restores the page set to a consistent state from the logs.
>
> If one or more page sets are damaged:
>
> 1. Rename the damaged page sets and define replacement page sets.
> 2. Copy the most recent backup page sets to the replacement page sets.
> 3. Restart the queue manager. WebSphere MQ automatically applies any updates that are necessary from the logs.
>
> You cannot restart the queue manager if page set zero is not available. If one of the other page sets is not available, you can comment out the page set DD statement in the queue manager start-up JCL procedure. This lets you defer recovery of the defective page set, enabling other users to continue accessing WebSphere MQ.
>
> **When you add the page set back to the JCL procedure, system restart reads the log from the point where the page set was removed from the JCL to the end of the log. This could take a long time if a lot of data has been logged.**

A reason code of MQRC_PAGESET_ERROR is returned to any application that tries to access a queue defined on a page set that is not available.

When you have restored the defective page set, restore its associated DD statement and restart the queue manager.

The operator actions described here are only possible if all log data sets are available. If your log data sets are lost or damaged, see "Restarting if you have lost your log data sets" on page 110.

Page set full

Problem
There is not enough space on a page set for one of the following:
- **MQPUT** or **MQPUT1** calls to be completed
- Object manipulation commands to be completed (for example, DEFINE QLOCAL)
- **MQOPEN** calls for dynamic queues to be completed

Symptoms
The request fails with reason code MQRC_STORAGE_MEDIUM_FULL. The queue manager cannot complete the request because there is not enough space remaining on the page set.

The cause of this problem could be due to messages accumulating on a transmission queue because they cannot be sent to another system.

System action
Further requests that use this page set are blocked until enough messages are removed or objects deleted to make room for the new incoming requests.

Operator action
Use the WebSphere MQ command DISPLAY USAGE PSID(*) to identify which page set is full.

System programmer action
You can either enlarge the page set involved or reduce the loading on that page set by moving queues to another page set. See "Managing page sets" on page 88 for more information about these tasks. If the cause of the problem is messages accumulating on the transmission queue, consider starting distributed queuing to transmit the messages.

Coupling Facility and DB2 problems

This section covers the problems that you might encounter with the Coupling Facility and DB2:
- "Storage medium full"
- "A DB2 system fails" on page 148
- "A DB2 data-sharing group fails" on page 149
- "DB2 and the Coupling Facility fail" on page 150

Storage medium full

Problem
A Coupling Facility structure is full.

Symptoms

If a queue structure becomes full, return code MQRC_STORAGE_MEDIUM_FULL is returned to the application.

If the administration structure becomes full, the exact symptoms depend on which processes experience the error, they might range from no responses to CMDSCOPE(GROUP) commands, to queue manager failure as a result of problems during commit processing.

System programmer action

You can use WebSphere MQ to inhibit **MQPUT** operations to some of the queues in the structure to prevent applications from writing more messages, start more applications to get messages from the queues, or quiesce some of the applications that are putting messages to the queue.

Alternatively you can use XES facilities to alter the structure size in place. The following z/OS command alters the size of the structure:

```
SETXCF START,ALTER,STRNAME=structure-name,SIZE=newsize
```

where *newsize* is a value that is less than the value of MAXSIZE specified on the CFRM policy for the structure, but greater than the current Coupling Facility size.

You can monitor the utilization of a Coupling Facility structure with the DISPLAY CFSTATUS command.

A DB2 system fails

If a DB2 subsystem that WebSphere MQ is connected to fails, WebSphere MQ attempts to reconnect to the subsystem and continue working. If you specified a DB2 group attach name in the QSGDATA parameter of the CSQ6SYSP system parameter module, WebSphere MQ reconnects to another active DB2 that is a member of the same data-sharing group as the failed DB2, if one is available on the same z/OS image.

There are some queue manager operations that do not work while WebSphere MQ is not connected to DB2. These are:

- Deleting a shared queue or group object definition.
- Altering, or issuing **MQSET** on, a shared queue or group object definition. The restriction of **MQSET** on shared queues means that operations such as triggering or the generation of performance events do not work correctly.
- Defining new shared queues or group objects.
- Displaying shared queues or group objects.
- Starting, stopping, or other actions for shared channels.

Other WebSphere MQ API operations continue to function as normal for shared queues, and all WebSphere MQ operations can be performed against the queue manager private versions (COPY objects) built from GROUP objects. Similarly, any shared channels that are running continue normally until they end or have an error, when they go into retry state.

When WebSphere MQ reconnects to DB2, resynchronization is performed between the queue manager and DB2. This involves notifying the queue manager of new objects that have been defined in DB2 while it was disconnected (other queue

managers might have been able to continue working as normal on other z/OS images through other DB2 subsystems), and updating object attributes of shared queues that have changed in DB2. Any shared channels in retry state are recovered.

If a DB2 fails, it might have owned locks on DB2 resources at the time of failure. In some cases, this might make certain WebSphere MQ objects unavailable to other queue managers that are not otherwise affected. To resolve this, restart the failed DB2 so that it can perform recovery processing and release the locks.

A DB2 data-sharing group fails

If an entire DB2 data-sharing group fails, recovery might be to the time of failure, or to a previous point in time.

In the case of recovery to the point of failure, WebSphere MQ reconnects when DB2 has been recovered, the resynchronization process takes places and normal queue manager function is resumed.

However, if DB2 is recovered to a previous point in time, there might be inconsistencies between the actual queues in the Coupling Facility structures and the DB2 view of those queues. For example, at the point in time DB2 is recovered to, a queue existed that has since been deleted and its location in the Coupling Facility structure reused by the definition of a new queue that now contains messages.

If you find yourself in this sort of situation, you must stop all the queue managers in the queue-sharing group, clear out the Coupling Facility structures, and restart the queue managers. You must then use WebSphere MQ commands to define any missing objects. To do this, use the following procedure:

1. Prevent WebSphere MQ from reconnecting to DB2 by starting DB2 in utility mode, or by altering security profiles.
2. If you have any important messages on shared queues, you might be able to off-load them using the COPY function of the CSQUTIL utility program, but this might not work.
3. Terminate all queue managers.
4. Use the following z/OS command to clear all structures:

```
SETXCF FORCE,STRUCTURE,STRNAME=
```

5. Restore DB2 to a historical point in time.
6. Reestablish queue manager access to DB2.
7. Restart the queue managers.
8. Recover the WebSphere MQ definitions from backup copies.
9. Reload any off-loaded messages to the shared queues.

When the queue managers restart, they attempt to resynchronize local COPY objects with the DB2 GROUP objects. This might cause WebSphere MQ to attempt to do the following:

• Create COPY objects for old GROUP objects that existed at the point in time DB2 has recovered to.

- Delete COPY objects for GROUP objects that were created since the point in time DB2 has recovered to and so do not exist in the database.

The DELETE of COPY objects is attempted with the NOPURGE option, so it fails for queue managers that still have messages on these COPY queues.

DB2 and the Coupling Facility fail

If the Coupling Facility fails, the queue manager might fail, and DB2 will also fail if it is using this Coupling Facility.

Recover DB2 using DB2 recovery procedures. When DB2 has been restarted, you can restart the queue managers. The CF administration structure will also have failed, but this is rebuilt by restarting all the queue managers within the queue-sharing group.

If a single application structure within the Coupling Facility suffers a failure, the effect on the queue manager depends on the level of the queue manager and the CFLEVEL of the failed CF structure:

- If the CF application structure is CFLEVEL(3) or higher and RECOVER is set to YES, it will not be usable until you recover the CF structure by issuing an MQSC RECOVER CFSTRUCT command to the queue manager that will do the recovery. You can specify a single CF structure to be recovered, or you can recover several CF structures simultaneously. The queue manager performing the recovery locates the relevant backups on all the other queue managers' logs using the data in DB2 and the bootstrap data sets. The queue manager replays these backups in the correct time sequence across the queue sharing group, from just before the last backup through to the point of failure. If a recoverable application structure has failed, any further application activity is prevented until the structure has been recovered. If the administration structure has also failed, all the queue managers in the queue-sharing group must be started before the RECOVER CFSTRUCT command can be issued. All queue managers can continue working with local queues and queues in other CF structures during recovery of a failed CF structure.
- If the CF application structure is CFLEVEL(3) or higher and RECOVER is set to NO, the structure is automatically reallocated by the next **MQOPEN** request performed on a queue defined in the structure. All messages are lost, as the structure can only contain non-persistent messages.
- If the CF application structure has a CFLEVEL less than 3, the queue manager fails. On queue manager restart, peer recovery attempts to connect to the structure, detect that the structure has failed and allocate a new version of the structure. All messages on shared queues that were in CF structures affected by the Coupling Facility failure are lost.

If the structure has experienced a connection failure, the queue manager fails. On queue manager restart (after the connection failure has been rectified), the connection is reestablished, but no recovery takes place for a structure at any CFLEVEL.

See WebSphere MQ Script (MQSC) Command Reference for details of the RECOVER CFSTRUCT command.

Problems with long-running units of work

This section explains what to do if you encounter a long-running unit of work during restart. In this context, this means a unit of work that has been active for a

long period of time (possibly days or even weeks) so that the origin RBA of the unit of work is outside the scope of the current active logs. This means that restart could take a long time, because all the log records relating to the unit of work have to be read, which might involve reading archive logs.

Old unit of work found during restart

Problem
> A unit of work with an origin RBA that predates the oldest active log has been detected during restart.

Symptoms
> WebSphere MQ issues the following message:

```
CSQR020I +CSQ1 OLD UOW FOUND
```

System action
> Information about the unit of work is displayed, and message CSQR021D is issued, requesting a response from the operator.

System programmer action
> None.

Operator action
> Decide whether to commit the unit of work or not. If you choose not to commit the unit of work, it is handled by normal restart recovery processing. Because the unit of work is old, this is likely to involve using the archive log, and so takes longer to complete.

IMS-related problems

This section includes plans for the following problems that you might encounter in the IMS environment:
- "IMS cannot connect to WebSphere MQ"
- "IMS application problem" on page 152
- "IMS is not operational" on page 152

IMS cannot connect to WebSphere MQ

Problem
> The IMS adapter cannot connect to WebSphere MQ.

Symptoms
> IMS remains operative. The IMS adapter issues these messages for control region connect:
> - CSQQ001I
> - CSQQ002E
> - CSQQ003E
> - CSQQ004E
> - CSQQ005E
> - CSQQ007E
>
> For details, see the WebSphere MQ for z/OS Messages and Codes manual.
>
> If an IMS application program tries to access WebSphere MQ while the IMS adapter cannot connect, it can either receive a completion code and

reason code, or terminate abnormally. This depends on the value of the REO option in the SSM member of IMS PROCLIB.

System action

All connection errors are also reported in the IMS message DFS3611.

System programmer action

None.

Operator action

Analyze and correct the problem, then restart the connection with the IMS command:

```
/START SUBSYS subsysname
```

IMS requests the adapter to resolve in-doubt units of recovery.

IMS application problem

Problem

An IMS application terminates abnormally.

Symptoms

The following message is sent to the user's terminal:

```
DFS555I  TRANSACTION tran-id ABEND abcode
MSG IN PROCESS: message data:
```

where *tran-id* represents any IMS transaction that is terminating abnormally and *abcode* is the abend code.

System action

IMS requests the adapter to resolve the unit of recovery. IMS remains connected to WebSphere MQ.

System programmer action

None.

Operator action

As indicated in message DFS554A on the IMS master terminal.

IMS is not operational

Problem

IMS is not operational.

Symptoms

More than one symptom is possible:

*   IMS waits or loops

    WebSphere MQ cannot detect a wait or loop in IMS, so you must find the origin of the wait or loop. This can be IMS, IMS applications, or the IMS adapter.

*   IMS terminates abnormally.

    *   See the manuals *IMS/ESA Messages and Codes* and *IMS/ESA Failure Analysis Structure Tables* for more information.

– If threads are connected to WebSphere MQ when IMS terminates, WebSphere MQ issues message CSQ3201E. This message indicates that WebSphere MQ end-of-task (EOT) routines have been run to clean up and disconnect any connected threads.

System action
WebSphere MQ detects the IMS error and:
- Backs out in-flight work.
- Saves in-doubt units of recovery to be resolved when IMS is reconnected.

System programmer action
None.

Operator action
Resolve and correct the problem that caused IMS to terminate abnormally, then carry out an emergency restart of IMS. The emergency restart:
- Backs out in-flight transactions that changed IMS resources.
- Remembers the transactions with access to WebSphere MQ that might be in doubt.

You might need to restart the connection to WebSphere MQ with the IMS command:

```
/START SUBSYS subsysname
```

During startup, IMS requests the adapter to resolve in-doubt units of recovery.

Hardware problems

If a hardware error causes data to be unreadable, WebSphere MQ can still be recovered by using the *media recovery* technique:

1. To recover the data, you need a backup copy of the data. Use DFDSS or Access Method Services REPRO regularly to make a copy of your data.
2. Reinstate the most recent backup copy.
3. Restart the queue manager.

The more recent your backup copy, the more quickly your subsystem can be made available again.

When the queue manager restarts, it uses the archive logs to reinstate changes made since the backup copy was taken. You must keep sufficient archive logs to enable WebSphere MQ to reinstate the changes fully. Do not delete archive logs until there is a backup copy that includes all the changes in the log.

Chapter 6. Using the WebSphere MQ Utilities

Using the WebSphere MQ utilities

This chapter introduces the WebSphere MQ utility programs that are provided to help you perform various administrative tasks. The utility programs are described in the following chapters. Table 5, Table 6, Table 7 on page 156, Table 9 on page 156, and Table 10 on page 157, summarize what you can do with these utilities.

Table 5. The WebSphere MQ CSQUTIL utility program: Managing page sets

Purpose	Function	See topic
Format VSAM data sets as WebSphere MQ page sets.	FORMAT	"Formatting page sets (FORMAT)" on page 162
Control recovery processing used for WebSphere MQ page sets.	FORMAT	"Formatting page sets (FORMAT)" on page 162
Extract page set information.	PAGEINFO	"Page set information (PAGEINFO)" on page 166
Copy WebSphere MQ page sets.	COPYPAGE	"Expanding a page set (COPYPAGE)" on page 167
Copy WebSphere MQ page sets and reset the log information.	RESETPAGE	"Copying a page set and resetting the log (RESETPAGE)" on page 169

Table 6. The WebSphere MQ CSQUTIL utility program: Issuing commands

Purpose	Function	See topic
Issue WebSphere MQ commands.	COMMAND	"Issuing commands to WebSphere MQ (COMMAND)" on page 171
Produce a set of DEFINE, ALTER or DELETE commands for objects.	COMMAND	"Making a list of DEFINE commands" on page 174
Produce a client channel definition file.	COMMAND	"Making a client channel definition file" on page 176

Table 6. The WebSphere MQ CSQUTIL utility program: Issuing commands (continued)

Purpose	Function	See topic
Produce a set of DEFINE commands for objects (offline).	SDEFS	"Producing a list of WebSphere MQ define commands (SDEFS)" on page 178

Table 7. The WebSphere MQ CSQUTIL utility program: Managing queues

Purpose	Function	See topic
Copy contents of a queue to a data set.	COPY	"Copying queues into a data set while the queue manager is running (COPY)" on page 181
Copy contents of a queue to a data set (offline).	SCOPY	"Copying queues into a data set while the queue manager is not running (SCOPY)" on page 184
Delete contents of a queue.	EMPTY	"Emptying a queue of all messages (EMPTY)" on page 187
Restore contents of a queue.	LOAD	"Restoring messages from a data set to a queue (LOAD)" on page 188

Table 8. The WebSphere MQ CSQUTIL utility program: Migrating CSQXPARM

Purpose	Function	See topic
Produce an ALTER QMGR command from a channel initiator parameter module.	XPARM	"Migrating a channel initiator parameter module (XPARM)" on page 190

Table 9. The WebSphere MQ CSQJU003 Change log inventory utility

Purpose	Function	See topic
Add active or archive log data sets.	NEWLOG	"Adding information about a data set to the BSDS (NEWLOG)" on page 193

Table 9. The WebSphere MQ CSQJU003 Change log inventory utility (continued)

Purpose	Function	See topic
Delete active or archive log data sets.	DELETE	"Deleting information about a data set from the BSDS (DELETE)" on page 196
Supply passwords for archive logs.	ARCHIVE	"Supplying a password for archive log data sets (ARCHIVE)" on page 197
Control the next restart of the queue manager.	CRESTART	"Controlling the next restart (CRESTART)" on page 197
Set checkpoint records.	CHECKPT	"Setting checkpoint records (CHECKPT)" on page 198
Update the highest written log RBA.	HIGHRBA	"Updating the highest written log RBA (HIGHRBA)" on page 199

Table 10. The remaining WebSphere MQ utilities

Name	Purpose	See topic
CSQJU004 (Print log map utility)	List information about the log.	"The print log map utility (CSQJU004)" on page 200
CSQ1LOGP (Log print utility)	Print the log. Extract log records into sequential files.	"The log print utility (CSQ1LOGP)" on page 201
CSQ5PQSG (WebSphere MQ table update utility)	Add and remove queue-sharing group and queue manager entries in the WebSphere MQ tables held in the shared DB2 data-sharing group.	"The queue-sharing group utility (CSQ5PQSG)" on page 210
CSQJUFMT (Active log preformat utility)	Preformat log data sets	"The active log preformat utility (CSQJUFMT)" on page 213
CSQUDLQH (Dead-letter queue handler utility)	Process messages on the dead-letter queue.	"The dead-letter queue handler utility (CSQUDLQH)" on page 214

Table 10. The remaining WebSphere MQ utilities (continued)

Name	Purpose	See topic
CSQUCVX (Data conversion exit utility)	Generate data conversion exit routines. For information about the CSQUCVX utility, see the WebSphere MQ Application Programming Guide.	

These utilities are located in the thlqual.SCSQAUTH or thlqual.SCSQLOAD WebSphere MQ load libraries. Include the appropriate WebSphere MQ language load library thlqual.SCSQANLx (where x is the language letter) in the STEPLIB concatenation before thlqual.SCSQAUTH or thlqual.SCSQLOAD. The utility control statements are available only in U.S. English. In some cases, the DB2 library db2qual.SDSNLOAD is also needed.

Syntax diagrams

The syntax for commands is presented in the form of a diagram. There are two types of syntax diagram: railroad diagrams and dotted decimal diagrams.

Either type of syntax diagram tells you what you can do with the particular command, and indicates relationships between different options and, sometimes, different values of an option. Railroad diagrams are a visual format suitable for sighted users. Dotted decimal diagrams are text-based diagrams that are more helpful for blind or partially-sighted users.

Only railroad diagrams are available in PDFs: dotted decimal diagrams are only avaliable in the Information Center.

How to read railroad diagrams

Each railroad diagram begins with a double right arrow and ends with a right and left arrow pair. Lines beginning with a single right arrow are continuation lines. You read a railroad diagram from left to right and from top to bottom, following the direction of the arrows.

Other conventions used in railroad diagrams are:

Table 11. How to read railroad diagrams
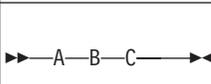
Convention	Meaning
►►—A—B—C———►◄	You must specify values A, B, and C. Required values are shown on the main line of a railroad diagram.
►►———————►◄ └A┘	You may specify value A. Optional values are shown below the main line of a railroad diagram.
►►——A——►◄ ├B┤ └C┘	Values A, B, and C are alternatives, one of which you must specify.
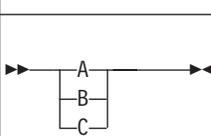
Table 11. How to read railroad diagrams (continued)

Convention	Meaning
►►─┬─A─┬─►◄ 　　├─B─┤ 　　└─C─┘	Values A, B, and C are alternatives, one of which you might specify.
►►─┬─,─┐───►◄ 　　▼─┬─A─┬─ 　　　├─B─┤ 　　　└─C─┘	You might specify one or more of the values A, B, and C. Any required separator for multiple or repeated values (in this example, the comma (,)) is shown on the arrow.
►►─┐─,─┐─►◄ 　　▼───A─┘	You might specify value A multiple times. The separator in this example is optional.
─A─ ►►─┬───┬─►◄ 　　├─B─┤ 　　└─C─┘	Values A, B, and C are alternatives, one of which you might specify. If you specify none of the values shown, the default A (the value shown above the main line) is used.
►►─┤─Name─├─►◄ **Name:** ├─A─┬───┤ 　　└─B─┘	The railroad fragment Name is shown separately from the main railroad diagram.
Punctuation and uppercase values	Specify exactly as shown.
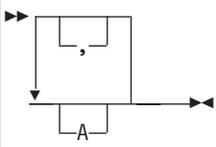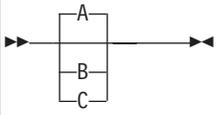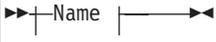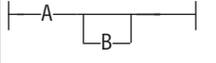
WebSphere MQ utility program (CSQUTIL)

The CSQUTIL utility program is provided with WebSphere MQ to help you to perform backup, restoration, and reorganization tasks, and to issue WebSphere MQ commands. Through this utility program, you can invoke functions in these groups:

Page set management
> These functions enable you to manage WebSphere MQ page sets. You can format data sets as page sets, change the recovery processing performed against page sets, extract page set information, increase the size of page sets and reset the log information contained in a page set. The page set must not belong to a queue manager that is currently running.

Command management

These functions enable you to:

- Issue commands to WebSphere MQ
- Produce a list of DEFINE, ALTER or DELETE commands for your WebSphere MQ objects

Queue management

These functions enable you to back up and restore queues and page sets, copy queues and page sets to another queue manager, reset your queue manager, or to migrate from one queue manager to another.

Specifically, you can:

- Copy messages from a queue to a data set
- Delete messages from a queue
- Restore previously copied messages to their respective queues

The scope of these functions can be either:

- A *queue*, in which case the function operates on all messages in the specified queue.
- A *page set*, in which case the function operates on all the messages, in all the queues, on the specified page set.

Use these functions only for your own queues; do not use them for system queues (those with names beginning SYSTEM).

All the page set management functions, and some of the other functions, operate while the queue manager is not running, so you do not need any special authorization other than the appropriate access to the page set data sets. For the functions that operate while the queue manager is running, CSQUTIL runs as an ordinary z/OS batch WebSphere MQ program, issuing commands through the command server and using the WebSphere MQ API to access queues.

You need the necessary authority to use the command server queues (SYSTEM.COMMAND.INPUT, SYSTEM.COMMAND.REPLY.MODEL, and SYSTEM.CSQUTIL.*), to use the WebSphere MQ DISPLAY commands, and to use the WebSphere MQ API to access any queues that you want to manage. See the usage notes for each function for more information.

Invoking the WebSphere MQ utility program

The CSQUTIL utility program runs as a z/OS batch program, below the 16 MB storage line. Specify the resources that the utility is to work with in the PARM parameter of the EXEC statement of the JCL.

```
// EXEC  PGM=CSQUTIL,PARM=
```

Figure 51. How to invoke the CSQUTIL utility program

where PARM= expands to :

```
►►──PARM=─'──────────────────────────────────────────'─────────────────────►◄
                │─QMGRname─────────────────────────│
                │─QSGname,DSGname,DB2ssid───────────│
                └─QMGRname,QSGname,DSGname,DB2ssid──┘
```
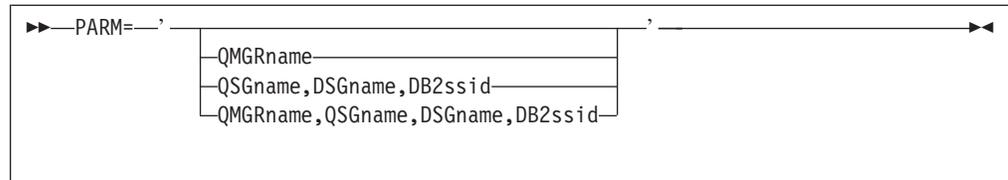
PARM parameters

QMGRname

> Specifies the 1– to 4– character name of the queue manager or queue-sharing group to which CSQUTIL is to connect.
>
> If you specify the name of a queue-sharing group, CSQUTIL connects to any queue manager in that group

QSGname

> Specifies the 1– to 4– character name of the queue-sharing group from which CSQUTIL is to extract definitions.

DSGname

> Specifies the 8–character name of the DB2 data-sharing group from which CSQUTIL is to extract definitions.

DB2ssid

> Specifies the 4–character name, or group attach name, of the DB2 database subsystem to which CSQUTIL is to attach for stand-alone functions.

Which PARM parameters do you need?:

Figure 51 on page 160 shows that you can specify one of four options on the PARM statement. The option you specify depends on the function you need to implement, as follows:

- Use PARM= (or omit it all together) if you are using only offline functions, and not QSGDISP(GROUP) or QSGDISP(SHARED).
- Use PARM='*QMGRname*' only if you intend to use functions that require the queue manager to be running, such as COPY and COMMAND.
- Use PARM='*QSGname,DSGname,DB2ssid*' if you intend to use the SDEFS function with either QSGDISP(GROUP) or QSGDISP(SHARED) specified. This is because CSQUTIL requires access to DB2 to perform the SDEFS function in this situation.
- Use PARM='*QMGRname,QSGname,DSGname,DB2ssid*' if you intend to combine the previous two functions in one CSQUTIL job.

If you specify a queue manager name as blanks, CSQUTIL uses the name of the default queue manager specified for z/OS batch programs in CSQBDEFV. The utility then uses this queue manager for the whole job step. When the utility connects to the queue manager, the authorization of the "signed-on user name" is checked to see which functions the invocation is allowed to use.

You specify the functions required by statements in the SYSIN data set according to these rules:

- The data set must have a record length of 80.
- Only columns 1 through 72 are significant. Columns 73 through 80 are ignored.
- Records with an asterisk (*) in column 1 are interpreted as comments and are ignored.
- Blank records are ignored.

- Each statement must start on a new line.
- A trailing - means continue from column 1 of the next record.
- A trailing + means continue from the first non-blank column of the next record.
- The keywords of statements are not case-sensitive. However, some arguments, such as queue name, are case-sensitive.

The utility statements refer to the default or explicitly named DDnames for input and output. Your job can use the COPY and LOAD functions repeatedly and process different page sets or queues during a single run of the utility.

All output messages are sent to the SYSPRINT data set, which must have a record format of VBA and a record length of 125.

While running, CSQUTIL uses temporary dynamic queues with names of the form SYSTEM.CSQUTIL.*

Return codes

When CSQUTIL returns to the operating system, the return code can be:

0   All functions completed successfully.

4   Some functions completed successfully, some did not or forced a syncpoint.

8   All the attempted functions failed.

12   No functions attempted; there was a syntax error in the statements or the expected data sets were missing.

In most cases, if a function fails or is forced to take a syncpoint, no further functions are attempted. In this case, the message CSQU147I replaces the normal completion message CSQU148I.

See the usage notes for each function for more information about success or failure.

Monitoring the progress of the WebSphere MQ utility program

To record the progress of CSQUTIL, every SYSIN statement is echoed to SYSPRINT.

The utility first checks the syntax of the statements in the SYSIN. The requested functions are started only if all the statements are syntactically correct.

Messages giving a commentary on the progress of each function are sent to SYSPRINT. When the processing of the utility is complete, statistics are printed with an indication of how the functions completed.

Formatting page sets (FORMAT)

Use the FORMAT function to format page sets on all data sets specified by DDnames CSQP0000 through CSQP0099. In this way, you can format up to 100 page sets in a single invocation of the utility program. Use the FORCE keyword to reuse existing data sets.

You can also use the FORMAT function to change the recovery processing that is performed against page sets when the queue manager starts, using the TYPE keyword. This can assist in changing or recovering page sets, or reintroducing page sets that have been offline.

In summary, to reinstate a page set with:
- No data, use TYPE(NEW)
- Old data, use TYPE(REPLACE)
- Old data brought up to date, do not use FORMAT but start the queue manager with a backed-up copy of the page set

Page sets have identifiers (PSIDs, in the range 00 through 99) which are established by the DDnames used for the data sets in the queue manager started task procedure; DDname CSQP00nn specifies the page set with identifier nn. The DDnames you use for the FORMAT function do not have to correspond to those used in the queue manager started task procedure, and do not therefore have any significance regarding page set identifiers.

Page set management (FORMAT)

```
>>─FORMAT─┬────────┬─┬─PAGES(1)───┬─┬─TYPE(RECOVER)─┬────────────><
          └─FORCE──┘ └─PAGES(nnn)─┘ ├─TYPE(NEW)─────┤
                                     └─TYPE(REPLACE)─┘
```
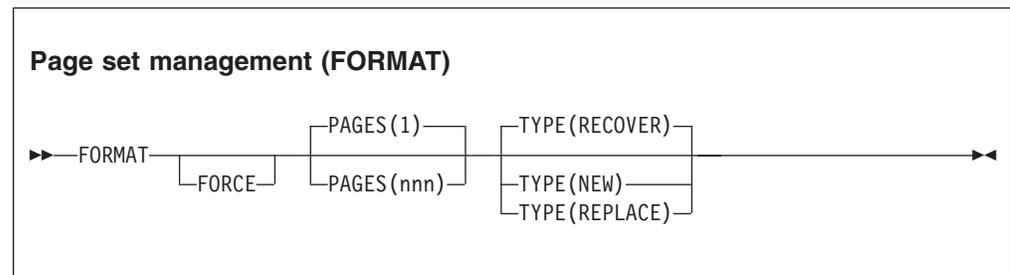
Keywords and parameters

FORCE
Specifies that existing data sets are to be reused without having to delete and redefine them first. You must define any page sets you want to reuse with the REUSE attribute in the AMS DEFINE CLUSTER statement. For more information about DEFINE CLUSTER, see the *DFSMS/MVS Access Method Services for VSAM* or the *DFSMS/MVS Access Method Services for the Integrated Catalog Facility* manual.

The FORCE keyword is not valid if TYPE(REPLACE) is specified.

PAGES(*nnn*)
Specifies the minimum number of pages to format in each page set. This enables a data set that spans more than one volume to be formatted.

Formatting of the data set is always done in whole space allocations, as specified as primary or secondary quantities when the data set is defined. The number of space allocations formatted is the minimum necessary to provide the requested number of pages; if there is insufficient data set space available, as many extents as can be obtained are formatted. If an existing page set is being reused (with the FORCE keyword), the whole page set is formatted, if that is larger.

The number of pages must be in the range 1 through 16 777 213 (because the maximum page set size is 64 GB (gigabytes)). The default is 1.

The PAGES keyword is not valid if TYPE(REPLACE) is specified.

TYPE

Specifies the type of recovery processing that is performed against queue manager page sets. Values are:

RECOVER

Use RECOVER for a data set that is to be a completely new page set for a queue manager (that is, to have a PSID which was never been used before).

This is the default.

The data set is formatted, and any messages or other data are erased. When the queue manager is restarted with a DDname for the new PSID that specifies this data set added to its started task procedure, it will be recognised as a new page set.

If such a data set was used as a page set with a PSID that has been used before, on restart the queue manager will attempt to recover all queues and their messages that use storage classes that reference the page set from the time the page set was first used. This may make restart a lengthy process, and is unlikely to be what is wanted.

NEW

Use NEW for a data set that is to be a page set with a PSID that has been used before for a queue manager and whose data can be discarded, in order to restart a failed queue manager quickly or to reintroduce the page set after it has been offline.

The data set is formatted, and any messages or other data are erased. When the queue manager is restarted, with a DDname for the old PSID that specfies this data set, it does not recover the page set but treats it as if it has been newly added to the queue manager, and any historical information about it is discarded. All queues that use storage classes referencing this page set are cleared of all messages, in a similar fashion to the way that nonpersistent messages are cleared during restart processing. This means that there will be no impact on restart time.

REPLACE

Use REPLACE for a data set with a PSID that has been used before for a queue manager and whose data is known to be consistent and up-to-date, in order to reintroduce the page set after being offline.

The data set is not formatted, and any messages or other data are preserved. When the queue manager is restarted with a DDname for the PSID that specifies this data set, it does not recover the page set but treats it as if it has never been offline, and any historical information about it is retained. All queues that use storage classes that reference the page set keep their messages. This means that there will be no impact on restart time.

This option will only be successful if the page set is in a consistent state; that is, on its last use the queue manager was terminated normally by a STOP QMGR MODE(FORCE) or MODE(QUIESCE) command.

Example

Figure 52 illustrates how the FORMAT command is invoked from CSQUTIL. In this example, two page sets, referenced by CSQP0000 and CSQP0003, are formatted by CSQUTIL.

```
//FORMAT   EXEC PGM=CSQUTIL
//STEPLIB  DD   DISP=SHR,DSN=thlqual.SCSQANLE
//         DD   DISP=SHR,DSN=thlqual.SCSQAUTH
//CSQP0000 DD   DISP=OLD,DSN=pageset.dsname0
//CSQP0003 DD   DISP=OLD,DSN=pageset.dsname3
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
FORMAT
/*
```

Figure 52. Sample JCL for the FORMAT function of CSQUTIL

Figure 53 illustrates how the FORMAT command with the TYPE option is invoked from CSQUTIL. In this example, the page set referenced by CSQP0003 is formatted by CSQUTIL.

```
//FORMAT   EXEC PGM=CSQUTIL
//STEPLIB  DD   DISP=SHR,DSN=thlqual.SCSQANLE
//         DD   DISP=SHR,DSN=thlqual.SCSQAUTH
//CSQP0003 DD   DISP=OLD,DSN=page set.dsname3
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
FORMAT TYPE(RECOVER)
/*
```

Figure 53. Sample JCL for the FORMAT function of CSQUTIL with the TYPE option

Usage notes

1. You cannot format page sets that belong to a queue manager that is still running.
2. When you use FORMAT, it is not necessary to specify a queue manager name.
3. If you use TYPE(REPLACE), recovery logs starting from when the page set was first used with the queue manager, or from when the page set was last formatted, must be available.
4. If you use data set names in which the queue manager name is a high-level qualifier, you can more easily identify which page sets are used by which queue manager, if more than one queue manager is defined.
5. Any update to a resource due to the resolution of an incomplete unit of work, where the update relates to a page on a page set that has been formatted with TYPE(REPLACE) or TYPE(NEW), is not honored. The update to the resource is lost.
6. If there is an error when formatting a page set, it does not prevent other page sets from being formatted, although the FORMAT function is considered to have failed.
7. Failure of this function does not prevent other CSQUTIL functions being attempted.

Page set information (PAGEINFO)

Use the PAGEINFO function to extract page set information from one or more page sets, specified by DDnames in the range CSQP0000 through CSQP0099, for the source data sets from which page set information is required.

Page set management (PAGEINFO)

```
►►──PAGEINFO─────────────────────────────────────────────────────►◄
```

Keywords and parameters

There are no keywords or parameters.

Example

In Figure 54, page set information is required from two existing page sets.

```
//PAGEINFO EXEC PGM=CSQUTIL
//STEPLIB  DD   DISP=SHR,DSN=thlqual.SCSQANLE
//         DD   DISP=SHR,DSN=thlqual.SCSQAUTH
//CSQP0001 DD   DISP=OLD,DSN=page set.existing.name1
//CSQP0006 DD   DISP=OLD,DSN=page set.existing.name6
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD
* Extract page set information for 2 existing page sets (CSQS0001 and CSQS0006)
 PAGEINFO
/*
```

Figure 54. Sample JCL showing the use of the PAGEINFO function

where:

CSQP0001, CSQP0006
> Are the DDnames of the source data sets from which you want to extract page set information.

Information returned from PAGEINFO might include:
- Page set number
- Number of pages in a page set
- Queue manager associated with a page set
- Utility status information
- Page set recovery RBA for each page set
- System recovery RBA for all the page sets reported on by the PAGEINFO function

Usage notes
1. You cannot use PAGEINFO on the page sets of a queue manager that is running.
2. Failure of this function does not prevent other CSQUTIL functions from being attempted.

3. If you attempt to use the PAGEINFO function after the queue manager has terminated abnormally, the page sets might not have been closed properly. If a page set has not been closed properly, you cannot successfully run the PAGEINFO function against it. To avoid this problem, run the AMS VERIFY command before using the PAGEINFO function. The AMS VERIFY command might produce error messages. However, it does close the page sets properly so that the PAGEINFO function can complete successfully.

   For more information about the AMS VERIFY command, see the *DFSMS/MVS®* *Access Method Services for VSAM* or the *DFSMS/MVS Access Method Services for the Integrated Catalog Facility* manual.

4. The system recovery RBA relates only to those page sets processed; it does not relate to the whole queue manager unless all the page sets for the queue manager are included. If the page sets are from more than one queue manager, no system recovery RBA can be determined.

Expanding a page set (COPYPAGE)

Note: The COPYPAGE function is only used for *expanding* page sets. It is not used for making backup copies of page sets. If you want to do this, use AMS REPRO as described in "How to back up and recover page sets" on page 94. When you have used the COPYPAGE function, the page sets cannot be used by a queue manager with a different name, so do not rename your queue manager.

Use the COPYPAGE function to copy one or more page sets to a larger page set. All queues and messages on the page set are copied. If you copy page set zero, all the WebSphere MQ object definitions are also copied. Each page set is copied to a destination data set that must be formatted as a page set. Copying to a smaller page set is not supported.

If you use this function, you must modify the page set definition in the started task procedure to reflect the change of the name of the data set on which the new page set resides.

To use the COPYPAGE function, define DDnames in the range CSQS0000 through CSQS0099 for the source data sets, and define DDnames for the target data sets from CSQT0000 through CSQT0099 respectively.

For more information, see "Managing page sets" on page 88.

Page set management (COPYPAGE)

►►──COPYPAGE──►◄

Keywords and parameters

There are no keywords or parameters.

Example

In Figure 55 on page 168, two existing page sets are copied onto two new page sets. The procedure for this is:

1. Set up the required DDnames, where:

CSQP0005, CSQP0006
> Identify the destination data sets. These DDnames are used by the FORMAT function.

CSQS0005, CSQS0006
> Identify the source data sets containing the two page sets you want to copy.

CSQT0005, CSQT0006
> Identify the destination data sets (page sets), but this time for the COPYPAGE function.

2. Format the destination data sets, referenced by DDnames CSQP0005 and CSQP0006, as page sets using the FORMAT function.

3. Copy the two existing page sets onto the new page sets using the COPYPAGE function.

```
//COPYPAGE EXEC PGM=CSQUTIL
//STEPLIB  DD   DISP=SHR,DSN=thlqual.SCSQANLE
//         DD   DISP=SHR,DSN=thlqual.SCSQAUTH
//CSQP0005 DD   DISP=OLD,DSN=pageset.newname5
//CSQP0006 DD   DISP=OLD,DSN=pageset.newname6
//CSQS0005 DD   DISP=OLD,DSN=pageset.oldname5
//CSQS0006 DD   DISP=OLD,DSN=pageset.oldname6
//CSQT0005 DD   DISP=OLD,DSN=pageset.newname5
//CSQT0006 DD   DISP=OLD,DSN=pageset.newname6
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
* Format new data sets (CSQP0005 and CSQP0006) as page sets
 FORMAT
* Copy old page sets CSQS0005 and CSQS0006 to new
* page sets CSQT0005 and CSQT0006
 COPYPAGE
/*
```

Figure 55. Sample JCL showing the use of the COPYPAGE function

Usage notes

1. You cannot use COPYPAGE on page sets of a queue manager that is running.

2. Using COPYPAGE involves stopping the queue manager. This results in the loss of nonpersistent messages.

3. Before you use COPYPAGE, the new data sets must be preformatted as page sets. To do this, use the FORMAT function, as shown in Figure 55.

4. Ensure that the new (destination) data sets are larger than the old (source) data sets.

5. You cannot change the page set identifier (PSID) associated with a page set. For example, you cannot 'make' page set 03 become page set 05.

6. Failure of this function does not prevent other CSQUTIL functions from being attempted.

7. If you attempt to use the COPYPAGE function after the queue manager has terminated abnormally, the page sets might not have been closed properly. If a page set has not been closed properly, you cannot successfully run the COPYPAGE function against it.

   To avoid this problem, run the AMS VERIFY command before using the COPYPAGE function. The AMS VERIFY command might produce error messages. However, it does close the page sets properly, so that the COPYPAGE function can complete successfully.

For more information about the AMS VERIFY command, see the *DFSMS/MVS Access Method Services for VSAM* or the *DFSMS/MVS Access Method Services for the Integrated Catalog Facility* manual.

Copying a page set and resetting the log (RESETPAGE)

The RESETPAGE function is similar to the COPYPAGE function except that it also resets the log information in the new page sets. RESETPAGE lets you restart the queue manager from a known, valid set of page sets, even if the corresponding log data sets have been corrupted.

The source page sets for RESETPAGE must be in a consistent state. They must be either:
- Page sets that have been through a successful queue manager shutdown using the WebSphere MQ command STOP QMGR.
- Copies of page sets that have been through a successful stop.

The RESETPAGE function must not be run against copies of page sets made using fuzzy backup (see "Method 2: Fuzzy backup" on page 95) , or against page sets that are from a queue manager that has terminated abnormally.

RESETPAGE either:
- Copies page sets on all data sets referenced by DDnames CSQS0000 through CSQS0099 to new data sets referenced by DDnames CSQT0000 through CSQT0099. If you use this function, modify the page set definition in the started task procedure to reflect the change of the name of the data set on which the new page set resides.
- Resets the log information in the page set referenced by DDnames CSQP0000 through CSQP0099.

For more information, see "Managing page sets" on page 88.

Using the RESETPAGE function

You can use the RESETPAGE function to update a set of consistent page sets so that they can be used with a set of new (clean) BSDS and log data sets to start the queue manager. You would only have to do this if both copies of the log have been lost or damaged; you can restart from backup copies of page sets (and accept the resulting loss of data from the time the copies were made), or from your existing page sets.

In this situation, use the RESETPAGE function on **all** the page sets of the affected queue manager. You must also create new BSDS and log data sets.

Note: Do not use the RESETPAGE function on a subset of the page sets known to WebSphere MQ.

If you run the RESETPAGE function against any page sets, but do not provide clean BSDS and log data sets for the queue manager, WebSphere MQ attempts to recover the logs from RBA zero, and treats the page sets as empty. For example, the following messages would be produced if you attempted to use the RESETPAGE function to generate page sets zero, 1, 2, and 3 without providing a clean set of BSDS and log data sets:

```
CSQI021I +CSQ1 CSQIECUR PAGE SET 0 IS EMPTY. MEDIA RECOVERY STARTED
CSQI021I +CSQ1 CSQIECUR PAGE SET 1 IS EMPTY. MEDIA RECOVERY STARTED
CSQI021I +CSQ1 CSQIECUR PAGE SET 2 IS EMPTY. MEDIA RECOVERY STARTED
CSQI021I +CSQ1 CSQIECUR PAGE SET 3 IS EMPTY. MEDIA RECOVERY STARTED
```

Page set management (RESETPAGE)

```
►►──RESETPAGE──┬──────────┬──────────────────────────────►◄
               └──FORCE──┘
```

Keywords and parameters

FORCE

Specifies that the page sets specified by DDnames CSQP0000 through CSQP00nn are to be reset in place.

If FORCE is not specified, the page sets specified by DDnames CSQS0000 through CSQS00nn are copied to new page sets specified by DDnames CSQT0000 through CSQT00nn. This is the default.

Example

An existing page set, referenced by DDname CSQS0007, is copied to a new data set referenced by DDname CSQT0007. The new data set, which is also referenced by DDname CSQP0007, is already formatted as a page set before the RESETPAGE function is called.

```
//RESTPAGE EXEC PGM=CSQUTIL
//STEPLIB  DD   DISP=SHR,DSN=thlqual.SCSQANLE
//         DD   DISP=SHR,DSN=thlqual.SCSQAUTH
//CSQP0007 DD   DISP=OLD,DSN=pageset.newname7
//CSQS0007 DD   DISP=OLD,DSN=pageset.oldname7
//CSQT0007 DD   DISP=OLD,DSN=pageset.newname7
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
* Format new data set, CSQP0007, as page set
 FORMAT
* Copy page set CSQS0007 to CSQT0007 and reset it
 RESETPAGE
/*
```

Figure 56. Sample JCL showing the use of the RESETPAGE function

Usage notes

1. Do not use the RESETPAGE function against page sets after the queue manager has terminated abnormally. Page sets from a queue manager that terminated abnormally will probably contain inconsistent data; using RESETPAGE on page sets in this state leads to data integrity problems.

2. You cannot use RESETPAGE on page sets belonging to a queue manager that is running.

3. Before you use RESETPAGE, the new data sets must be preformatted as page sets. To do this, use the FORMAT function, as shown in Figure 56.

4. Ensure that the new (destination) data sets are larger than the old (source) data sets.

5. You cannot change the page set identifier (PSID) associated with a page set. For example, you cannot 'make' page set 03 become page set 05.

6. Failure of this function does not prevent other CSQUTIL functions from being attempted.

Issuing commands to WebSphere MQ (COMMAND)

Use the COMMAND function to:

1. Pass commands from an input data set to the queue manager.

2. Produce a list of DEFINE commands that describe the objects in a queue manager. The commands can be used to keep a record of the object definitions or to regenerate all or part of a queue manager's objects as part of a migration from one queue manager to another.

3. Produce a list of commands to change or delete a set of objects in a queue manager.

4. Make a client channel definition file.

The queue manager specified in the PARM parameter of the EXEC statement must be running.

```
Command management (COMMAND)

►►──COMMAND──┬──────────────────┬──┬──MAKEDEF(ddname2)──────────────────────────┬──►
             └─DDNAME(ddname1)─┘  ├──MAKEDEL(ddname2)──────────────────────────┤
                                  ├──MAKEREP(ddname2)─┐                         │
                                  └──MAKEALT(ddname2)─┴──┬───────────────────┬──┘
                                                         └─DATA──(──ddname4──)─┘

             ┌─RESPTIME(30)──┐
►──TGTQMGR(qmgrname)──┼──────────────┼──┬──────────────────┬───────────────────►
                      └─RESPTIME(nnn)─┘  └─MAKECLNT(ddname3)─┬──────────────┬─┘
                                                            └─CCSID(ccsid)─┘

   ┌─FAILURE(IGNORE)───┐
►──┼───────────────────┼───────────────────────────────────────────────────►◄
   ├─FAILURE(CONTINUE)─┤
   └─FAILURE(STOP)─────┘
```
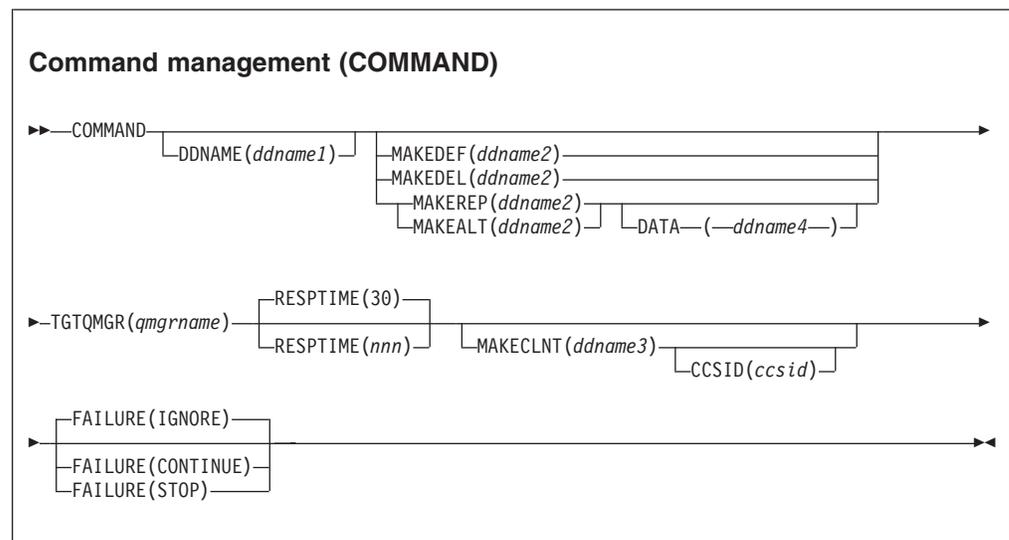
Keywords and parameters

DDNAME(*ddname1***)**

Specifies that the commands are to be read from a named input data set. If this keyword is omitted, the default DDname, CSQUCMD, is used.

ddname1 specifies the DDname that identifies the input data set from which commands are to be read.

MAKEDEF(*ddname2***), MAKEDEL(***ddname2***), MAKEREP(***ddname2***), MAKEALT(***ddname2***)**

Specify that commands are to be generated from any DISPLAY object commands in the input data set.

The commands that are generated are:

MAKEDEF
DEFINE NOREPLACE, with all the attributes and values returned by the DISPLAY commands. For the queue manager object, an ALTER command is generated with all the attributes and values.

MAKEDEL
DELETE. For local queues, NOPURGE is used.

MAKEREP
DEFINE REPLACE, with any keywords and values from the data set specified by the DATA keyword.

MAKEALT
ALTER, with any keywords and values from the data set specified by the DATA keyword.

Only one of these keywords may be specified. If these keywords are omitted, no commands are generated.

ddname2 specifies the DDname that identifies the output data set in which the DEFINE, DELETE or ALTER commands are to be stored. The data set should be RECFM=FB, LRECL=80. This data set can then be used as input for a later invocation of the COMMAND function or it can be incorporated into the initialization data sets CSQINP1 and CSQINP2.

DATA(*ddname4*) ddname4 specifes a data set from which command keywords and values are to be read, and appended to each command generated for MAKEREP or MAKEALT.

TGTQMGR(*qmgrname*)
Specifies the name of the queue manager where you want the commands to be performed. You can specify a target queue manager that is not the one you connect to. In this case, you would normally specify the name of a remote queue manager object that provides a queue manager alias definition (the name is used as the *ObjectQMgrName* when opening the command input queue). To do this, you must have suitable queues and channels set up to access the remote queue manager.

The default is that commands are performed on the queue manager to which you are connected, as specified in the PARM field of the EXEC statement.

RESPTIME(*nnn*)
Specifies the time in seconds to wait for a response to each command, in the range 5 through 999.

The default is 30 seconds.

MAKECLNT(*ddname3*)
Specifies that a client channel definition file is generated from any DISPLAY CHANNEL commands in the input data set that return information about client-connection channels, and any DISPLAY AUTHINFO commands that return information about authentication information objects for which the LDAPUSER and LDAPPWD attributes are not set.

If this keyword is omitted, no file is generated.

ddname3 specifies the DDname that identifies the output data set in which the generated file is to be stored; the data set should be RECFM=U, LRECL=6144. The file can then be downloaded as binary data to the client machine by a suitable file transfer program.

CCSID(*ccsid***)**

Specifies the coded character set identifier (CCSID) that is to be used for the data in a client channel definition file. The value must be in the range 1 through 65535; the default is 437. You can only specify CCSID if you also specify MAKECLNT.

Note: WebSphere MQ assumes that the data is to be in ASCII, and that the encoding for numeric data is to be MQENC_INTEGER_REVERSED.

FAILURE

Specifies what action to take if a WebSphere MQ command that is issued fails to execute successfully. Values are:

IGNORE

Ignore the failure; continue reading and issuing commands, and treat the COMMAND function as being successful. This is the default.

CONTINUE

Read and issue any remaining commands in the input data set, but treat the COMMAND function as being unsuccessful.

STOP Do not read or issue any more commands, and treat the COMMAND function as being unsuccessful.

Examples

This section gives examples of using the COMMAND function for the following:
*   "Issuing commands"
*   "Making a list of DEFINE commands" on page 174
*   "Making a list of ALTER commands" on page 175
*   "Making a client channel definition file" on page 176

Issuing commands:

In Figure 57 on page 174, the data sets referenced by DDnames CSQUCMD and OTHER contain sets of commands. The first COMMAND statement takes commands from the default input data set MY.COMMANDS(COMMAND1) and passes them to the queue manager. The second COMMAND statement takes commands from the input data set MY.COMMANDS(OTHER1), which is referenced by DDname OTHER, and passes them to the queue manager.

```
//COMMAND EXEC PGM=CSQUTIL,PARM='CSQ1'
//STEPLIB  DD   DISP=SHR,DSN=thlqual.SCSQANLE
//         DD   DISP=SHR,DSN=thlqual.SCSQAUTH
//CSQUCMD  DD DSN=MY.COMMANDS(COMMAND1),DISP=SHR
//OTHER    DD DSN=MY.COMMANDS(OTHER1),DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
* THE NEXT STATEMENT CAUSES COMMANDS TO BE READ FROM CSQUCMD DDNAME
COMMAND
* THE NEXT SET OF COMMANDS WILL COME FROM 'OTHER' DDNAME
COMMAND DDNAME(OTHER)
* THE NEXT STATEMENT CAUSES COMMANDS TO BE READ FROM CSQUCMD
* DDNAME AND ISSUED ON QUEUE MANAGER CSQ2 WITH A RESPONSE TIME
* OF 10 SECONDS
COMMAND TGTQMGR(CSQ2) RESPTIME(10)
/*
```

Figure 57. Sample JCL for issuing WebSphere MQ commands using CSQUTIL

Making a list of DEFINE commands:

In Figure 58 on page 175, the data set referenced by DDname CMDINP contains a set of DISPLAY commands. These DISPLAY commands specify generic names for each object type (except the queue manager itself). If you run these commands, a list is produced containing all the WebSphere MQ objects. In these DISPLAY commands, the ALL keyword is specified to ensure that all the attributes of all the objects are included in the list, and that all queue-sharing group dispositions are included.

The MAKEDEF keyword causes this list to be converted into a corresponding set of DEFINE NOREPLACE commands (ALTER for the queue manager). These commands are put into a data set referenced by the *ddname2* parameter of the MAKEDEF keyword, that is, OUTPUT1. If you run this set of commands, WebSphere MQ regenerates all the object definitions in the queue manager.

```
//QDEFS    EXEC PGM=CSQUTIL,PARM='CSQ1'
//STEPLIB  DD   DISP=SHR,DSN=thlqual.SCSQANLE
//         DD   DISP=SHR,DSN=thlqual.SCSQAUTH
//OUTPUT1  DD   DISP=OLD,DSN=MY.COMMANDS(DEFS)
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
COMMAND DDNAME(CMDINP) MAKEDEF(OUTPUT1)
/*
//CMDINP   DD *
DISPLAY STGCLASS(*) ALL QSGDISP(QMGR)
DISPLAY STGCLASS(*) ALL QSGDISP(GROUP)
DISPLAY QUEUE(*) ALL QSGDISP(QMGR)
DISPLAY QUEUE(*) ALL QSGDISP(GROUP)
DISPLAY QUEUE(*) ALL QSGDISP(SHARED)
DISPLAY TOPIC(*) ALL QSGDISP(QMGR)
DISPLAY TOPIC(*) ALL QSGDISP(GROUP)
DISPLAY NAMELIST(*) ALL QSGDISP(QMGR)
DISPLAY NAMELIST(*) ALL QSGDISP(GROUP)
DISPLAY PROCESS(*) ALL QSGDISP(QMGR)
DISPLAY PROCESS(*) ALL QSGDISP(GROUP)
DISPLAY CHANNEL(*) ALL QSGDISP(QMGR)
DISPLAY CHANNEL(*) ALL QSGDISP(GROUP)
DISPLAY AUTHINFO(*) ALL QSGDISP(QMGR)
DISPLAY AUTHINFO(*) ALL QSGDISP(GROUP)

DISPLAY CFSTRUCT(*) ALL
DISPLAY QMGR ALL

/*
```

Figure 58. Sample JCL for using the MAKEDEF option of the COMMAND function

Making a list of ALTER commands:

In Figure 59 on page 176, the data set referenced by DDname CMDINP contains a
DISPLAY command that will produce a list of all local queues with names
beginning ″ABC″.

The MAKEALT keyword causes this list to be converted into a corresponding set
of ALTER commands, each of which includes the data from the data set referenced
by DDname CMDALT. These commands are put into a data set referenced by the
ddname2 parameter of the MAKEALT keyword, that is, OUTPUTA. If you run this
set of commands, all the local queues with names beginning ″ABC″ will be
disabled for PUT and GET.

```
//QALTS   EXEC PGM=CSQUTIL,PARM='CSQ1 '
//STEPLIB  DD DISP=SHR,DSN=thlqual.SCSQANLE
//         DD DISP=SHR,DSN=thlqual.SCSQAUTH
//OUTPUTA  DD DISP=OLD,DSN=MY.COMMANDS(ALTS)
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
COMMAND DDNAME(CMDINP) MAKEALT(OUTPUTA) DATA(CMDALT)
/*
//CMDINP DD *
DISPLAY QLOCAL(ABC*)
/*
//CMDALT DD *
PUT(DISABLED) +
GET(DISABLED)
/*
```

Figure 59. Sample JCL for using the MAKEALT option of the COMMAND function

Making a client channel definition file:

In Figure 60, the data set referenced by DDname CMDCHL contains a DISPLAY CHANNEL command and a DISPLAY AUTHINFO command. The DISPLAY commands specify a generic name and the ALL keyword is specified to ensure that all the attributes are included.

The MAKECLNT keyword converts these attributes into a corresponding set of client channel definitions. These are put into a data set referenced by the *ddname3* parameter of the MAKECLNT keyword, that is, OUTCLNT, which is ready to be downloaded to the client machine.

```
//CLIENT   EXEC PGM=CSQUTIL,PARM='CSQ1'
//STEPLIB  DD   DISP=SHR,DSN=thlqual.SCSQANLE
//         DD   DISP=SHR,DSN=thlqual.SCSQAUTH
//OUTCLNT  DD   DISP=OLD,DSN=MY.CLIENTS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
COMMAND DDNAME(CMDCHL) MAKECLNT(OUTCLNT)
/*
//CMDCHL   DD *
DISPLAY CHANNEL(*) ALL TYPE(CLNTCONN)
DISPLAY AUTHINFO(*) ALL
/*
```

Figure 60. Sample JCL for using the MAKECLNT option of the COMMAND function

Usage notes

1. The rules for specifying commands in the input data set are the same as for the initialization data sets:
   - The data set must have a record length of 80.
   - Only columns 1 through 72 are significant. Columns 73 through 80 are ignored.
   - Records with an asterisk (*) in column 1 are interpreted as comments and are ignored.
   - Blank records are ignored.
   - Each command must start on a new record.
   - A trailing - means continue from column 1 of the next record.

- A trailing + means continue from the first non-blank column of the next record.
- The maximum number of characters permitted in a command is 32 762.

With the additional rule:

- A semicolon (;) can be used to terminate a command; the remaining data in the record is ignored.

See the WebSphere MQ Script (MQSC) Command Reference manual for more information about the rules for building WebSphere MQ commands.

2. If you specify the MAKEDEF keyword:
   - In the input data set, the DISPLAY commands for objects must contain the ALL parameter so that the complete definition of each object is produced. See Figure 58 on page 175.
   - To obtain a complete definition, you must DISPLAY the following:
     - queues
     - namelists
     - process definitions
     - channels
     - storage classes
     - authentication information objects
     - CF structures
     - queue manager

     Note: DEFINE commands are not generated for any local queues that can be identified as dynamic, or for channels that were defined automatically.
   - Do not specify the same MAKEDEF data set for more than one COMMAND function, unless its DD statement specifies a sequential data set with DISP=MOD.

3. If you specify the MAKEREP, MAKEALT, or MAKEDEL keywords:
   - In the input data set, include DISPLAY commands that select the set of objects for which you want to generate commands.
   - For MAKEREP and MAKEALT, the data (if any) from the data set specified by the DATA keyword is appended to each generated command, exactly as entered. The format of the data set and the rules for specifying command data are the same as for the command input data set. Because the same data is appended to each command, if you want to process several sets of objects, you will need to use several separate COMMAND functions, each with a different DATA data set.
   - Commands are not generated for channels that were defined automatically.

4.
   If you specify the MAKEDEF, MAKEREP, MAKEALT, or MAKEDEL keywords, commands are generated only for objects reported by the target queue manager (as specified by the TGTQMGR keyword or defaulted), even if CMDSCOPE is used in the DISPLAY commands. To generate commands for several queue managers in a queue-sharing group, use a separate COMMAND function for each.

   In a queue-sharing group, queues, processes, channels, storage classes and authentication information objects should each have two DISPLAY commands, one with QSGDISP(QMGR) and one with QSGDISP(GROUP). Queues should have a third with QSGDISP(SHARED). It is not necessary to specify

QSGDISP(COPY) because the required commands will be generated automatically when the commands for objects with QSGDISP(GROUP) are issued.

5. Do not specify the same MAKEDEF, MAKEREP, MAKEALT, or MAKEDEL data set for more than one COMMAND function, unless its DD statement specifies a sequential data set with DISP=MOD.

6. If you specify the MAKECLNT keyword:
   - In the input data set, the display commands for channels and authentication information objects must contain the ALL parameter so that the complete definition of each channel and authentication information object is produced.
   - If the DISPLAY commands return information for a given channel more than once, only the last set of information is used.
   - Do not specify the same client definition file data set for more than one COMMAND function, unless its DD statement specifies a sequential data set with DISP=MOD.

7. The results of DISPLAY commands used in conjunction with MAKEDEF, MAKEREP, MAKEALT, MAKEDEL or MAKECLNT are also sent to SYSPRINT.

8. If you specify the FAILURE keyword, a command is determined to be a success or failure according to the codes returned in message CSQN205I. If the return code is 00000000 and the reason code is 00000000 or 00000004, it is a success; for all other values it is a failure.

9. The COMMAND function is determined to be a success only if both:
   - All the commands in the input data set are read and issued and get a response from WebSphere MQ, regardless of whether the response indicates successful execution of the command or not.
   - Every command issued executes successfully, if FAILURE(CONTINUE) or FAILURE(STOP) is specified.

   If COMMAND fails, no further CSQUTIL functions are attempted.

10. You need the necessary authority to use command server queues (SYSTEM.COMMAND.INPUT, SYSTEM.COMMAND.REPLY.MODEL, and SYSTEM.CSQUTIL.*) and to use the WebSphere MQ commands that you want to issue.

Producing a list of WebSphere MQ define commands (SDEFS)

Use the SDEFS function to produce a list of DEFINE commands describing the objects in your queue manager or queue-sharing group.

```
┌──────────────────────────────────────────────────────────────────────────────────┐
│                                                                                    │
│  Command management (SDEFS)                                                         │
│                                                                                    │
│                                        ┌─QSGDISP(QMGR)────┐                         │
│  ▶▶──SDEFS──OBJECT──(──┬─AUTHINFO─┬──)──┤                 ├──────────────────▶      │
│                        ├─CHANNEL──┤    │                 │                          │
│                        ├─NAMELIST─┤    ├─QSGDISP(COPY)────┤                         │
│                        ├─PROCESS──┤    ├─QSGDISP(PRIVATE)─┤                         │
│                        ├─QALIAS───┤    ├─QSGDISP(GROUP)───┤                         │
│                        ├─QLOCAL───┤    └─QSGDISP(SHARED)──┘                         │
│                        ├─QMODEL───┤                                                 │
│                        ├─QREMOTE──┤                                                 │
│                        ├─QUEUE────┤                                                 │
│                        ├─STGCLASS─┤                                                 │
│                        └─TOPIC────┘                                                 │
│                                                                                    │
│  ▶──MAKEDEF──(──ddname2──)──────────────────────────────────────────────────◀     │
│                                                                                    │
└──────────────────────────────────────────────────────────────────────────────────┘
```
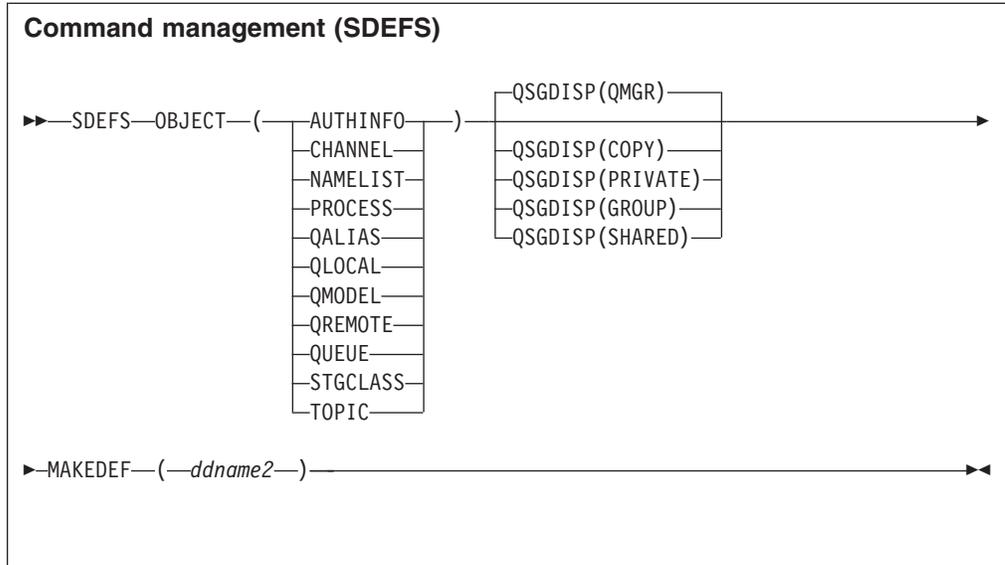
Keywords and parameters

OBJECT

Specifies the type of object to be listed.

A value of QUEUE lists queues of all types, as if you had specified QALIAS, QLOCAL, QMODEL and QREMOTE.

QSGDISP

Specifies from where the object definition information is obtained. Depending on how the object has been defined, this information is either:

- On the page set zero referred to by the CSQP0000 DD statement, or
- In a DB2 shared repository.

Permitted values are shown in Table 12.

Table 12. SDEFS QSGDISP parameters and their actions

QSGDISP parameter	What the SDEFS utility does
QMGR	Creates DEFINE statements for the specified object type from definitions held on the page set zero referred to by the CSQP0000 DD statement. (1) Only objects defined with QSGDISP(QMGR) are included.
COPY	Creates DEFINE statements for the specified object type from definitions held on the page set zero referred to by the CSQP0000 DD statement. (1) Only objects defined with QSGDISP(COPY) are included.
PRIVATE	Creates DEFINE statements for the specified object type from definitions held on the page set zero referred to by the CSQP0000 DD statement. (1) Both QSGDISP(QMGR) and QSGDISP(COPY) objects are included.

Table 12. SDEFS QSGDISP parameters and their actions (continued)

QSGDISP parameter	What the SDEFS utility does
GROUP	Creates DEFINE statements for the specified object type from definitions held on DB2 resource definition tables for the specified queue-sharing group.
	Only objects defined with QSGDISP(GROUP) are included.
	No CSQP0000 DD statement is required; the DB2 subsystem specified at object definition is accessed. The DB2 library db2qual.SDSNLOAD is required.
SHARED	Creates DEFINE statements for all local queues defined with QSGDISP(SHARED) by accessing the DB2 resource definition table for the specified queue-sharing group.
	This parameter is permitted only with OBJECT(QLOCAL) or OBJECT(QUEUE).
	No CSQP0000 DD statement is required; the DB2 subsystem specified at object definition is accessed. The DB2 library db2qual.SDSNLOAD is required.

Notes:

1. Because only page set zero is accessed, you must ensure that the queue manager is not running.

MAKEDEF(*ddname2***)**

> Specifies that define commands generated for the object are to be placed in the output data set identified by the DDname. The data set should be RECFM=FB, LRECL=80. This data set can then be used as input for a later invocation of the COMMAND function or it can be incorporated into the initialization data sets CSQINP1 and CSQINP2.

> The commands generated are DEFINE NOREPLACE, with all the attributes and values for the object.

> **Note:** DEFINE commands are not generated for any local queues that can be identified as dynamic, or for channels that were defined automatically.

Examples

```
//SDEFS    EXEC PGM=CSQUTIL
//STEPLIB  DD   DISP=SHR,DSN=thlqual.SCSQANLE
//         DD   DISP=SHR,DSN=thlqual.SCSQAUTH
//CSQP0000 DD   DISP=OLD,DSN=pageset.dsname0
//OUTPUT1  DD   DISP=OLD,DSN=MY.COMMANDS(DEFS)
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
SDEFS OBJECT(QUEUE) MAKEDEF(OUTPUT1)
/*
```

Figure 61. Sample JCL for the SDEFS function of CSQUTIL

```
//SDEFS    EXEC PGM=CSQUTIL
//STEPLIB DD   DISP=SHR,DSN=thlqual.SCSQANLE
//        DD   DISP=SHR,DSN=thlqual.SCSQAUTH
//        DD   DISP=SHR,DSN=db2qual.SDSNLOAD
//OUTPUT1 DD   DISP=OLD,DSN=MY.COMMANDS(DEFS)
//SYSPRINT DD  SYSOUT=*
//SYSIN   DD   *
SDEFS OBJECT(QLOCAL) QSGDISP(SHARED) MAKEDEF(OUTPUT1)
/*
```

Figure 62. Sample JCL for the SDEFS function of CSQUTIL for objects in the DB2 shared repository

Usage notes

1. For local queues, do not use SDEFS for a queue manager that is running because results will be unpredictable. You can avoid doing this accidentally by using DISP=OLD in the CSQP0000 DD statement. For shared or group queue definitions, this does not matter because the information is derived from DB2.

2. When you use SDEFS for local queues you do not need to specify a queue manager name. However, for shared and group queue definitions, a queue manager name is required to access DB2.

3. To use the SDEFS function more than once in a job, specify different DDnames and data sets for each invocation of the function, or specify a sequential data set and DISP=MOD in the DD statements.

4. If the SDEFS function fails, no further CSQUTIL functions are attempted.

Copying queues into a data set while the queue manager is running (COPY)

Use the COPY function to copy queued messages to a sequential data set while the queue manager is running, without destroying any messages in the original queues.

The scope of the COPY function is determined by the keyword that you specify in the first parameter. You can either copy all the messages from a named queue, or all the messages from all the queues on a named page set.

Use the complementary function, LOAD, to restore the messages to their respective queues.

Note:

1. If you want to copy the object definitions from the named page set, use COPYPAGE.

2. If you want to copy messages to a data set when the queue manager is stopped, use SCOPY.

3. See "Syncpoints" on page 183 for information about how to avoid problems with duplicate messages if this function fails.

```
┌─────────────────────────────────────────────────────────────────────────────┐
│  Queue management (COPY)                                                      │
│                                                                               │
│  ►►──COPY──┤ Object Selection ├──┤ DDname Selection ├──────────────────►◄      │
│                                                                               │
│  Object  Selection                                                            │
│                                                                               │
│  ├──────QUEUE──(──q-name──)─────────────────────────────────────────────┤     │
│         │                         ┌─DEFTYPE(ALL)───────┐                       │
│         └─PSID──(──psid-number──)──┤                    ├──┘                   │
│                                    └─DEFTYPE(PREDEFINED)─┘                     │
│                                                                               │
│  DDname  Selection                                                            │
│                                                                               │
│  ├────────DDNAME──(──ddname──)────────────────────────────────────────────┤   │
│                                                                               │
└─────────────────────────────────────────────────────────────────────────────┘
```

Keywords and parameters

QUEUE(*q-name*)
Specifies that messages in the named queue are to be copied. The keyword
QUEUE can be abbreviated to Q.

q-name specifies the name of the queue to be copied. This name is
case-sensitive.

PSID(*psid-number*)
Specifies that all the messages in all the queues in the specified page set are to
be copied.

psid-number is the page set identifier, which specifies the page set to be used.
This identifier is a two-digit integer (whole number) representing a single page
set.

DEFTYPE
Specifies whether to copy dynamic queues:

ALL   Copy all queues; this is the default.

PREDEFINED
Do not include dynamic queues; this is the same set of queues that are
selected by the COMMAND and SDEFS functions with the MAKEDEF
parameter.

DDNAME(*ddname*)
Specifies that the messages are to be copied to a named data set. If this
keyword is omitted, the default DDname, CSQUOUT, is used. The keyword
DDname can be abbreviated to DD.

ddname specifies the DDname of the destination data set, which is used to store
the messages. The record format of this data set must be variable block
spanned (VBS).

Example

```
//COPY    EXEC PGM=CSQUTIL,PARM='CSQ1'
//STEPLIB DD   DISP=SHR,DSN=thlqual.SCSQANLE
//        DD   DISP=SHR,DSN=thlqual.SCSQAUTH
//OUTPUTA  DD DSN=SAMPLE.UTILITY.COPYA,DISP=(NEW,CATLG),
//           SPACE=(CYL,(5,1),RLSE),UNIT=SYSDA,
//           DCB=(RECFM=VBS,BLKSIZE=23200)
//CSQUOUT  DD DSN=SAMPLE.UTILITY.COPY3,DISP=(NEW,CATLG),
//           SPACE=(CYL,(5,1),RLSE),UNIT=SYSDA,
//           DCB=(RECFM=VBS,BLKSIZE=23200)
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
* COPY WHOLE PAGE SET TO 'CSQUOUT'
COPY PSID(03)
* COPY ONE QUEUE TO 'OUTPUT'
COPY QUEUE(ABC123A) DDNAME(OUTPUTA)
/*
```

Figure 63. Sample JCL for the CSQUTIL COPY functions. The sample shows two instances of the COPY function—one COPY to the default DDNAME, CSQUOUT; the other to DDNAME OUTPUTA, which overrides CSQUOUT.

Usage notes

1. The queues involved must not be in use when the function is invoked.
2. If you want to operate on a range of page sets, repeat the COPY function for each page set.
3. The function operates only on local queues.
4. A COPY PSID function is considered successful only if it successfully copies all the queues on the page set.
5. If you try to copy an empty queue (either explicitly by COPY QUEUE or because there are one or more empty queues on a page set that you are copying), data indicating this is written to the sequential data set, and the copy is considered to be a success. However, if you attempt to copy a nonexistent queue, or a page set containing no queues, the COPY function fails, and no data is written to the data set.
6. If COPY fails, no further CSQUTIL functions are attempted.
7. To use the COPY function more than once in the job, specify different DDnames and data sets for each invocation of the function, or specify a sequential data set and DISP=MOD in the DD statements.
8. You need the necessary authority to use the command server queues (SYSTEM.COMMAND.INPUT, SYSTEM.COMMAND.REPLY.MODEL, and SYSTEM.CSQUTIL.*), to use the DISPLAY QUEUE and DISPLAY STGCLASS MQSC commands, and to open the queues that you want to copy with the MQOO_INPUT_EXCLUSIVE and MQOO_BROWSE options.

Syncpoints

The queue management functions used when the queue manager is running operate within a syncpoint so that, if a function fails, its effects can be backed out. The queue manager attribute, MAXUMSGS, specifies the maximum number of messages that a task can get or put within a single unit of recovery.

MAXUMSGS should normally be set to a low value, both to protect against looping applications, and because there might be a very large CPU cost in committing a large number of messages.

The utility forcibly takes syncpoints as required and issues the warning message CSQU087I. If the function subsequently fails, the changes already committed are not backed out. Do not just rerun the job to correct the problem or you might get duplicate messages on your queues. Instead, use the current depth of the queue to work out, from the utility output, which messages have not been backed out. Then determine the most appropriate course of action. For example, if the function is LOAD, you can empty the queue and start again, or you can choose to accept duplicate messages on the queues.

To avoid such difficulties if the function fails, but at the risk of incurring a large CPU cost, set MAXUMSGS to be greater than:

- The number of messages in the queue, if you are working with a single queue.
- The number of messages in the longest queue in the page set, if you are working with an entire page set.

Use the DISPLAY QSTATUS command to find out the value of the CURDEPTH attribute, which is the current depth of the queue. To find out the value of MAXUMSGS, use the DISPLAY QMGR MAXUMSGS command. See the WebSphere MQ Script (MQSC) Command Reference manual for more information about these commands.

Copying queues into a data set while the queue manager is not running (SCOPY)

Use the SCOPY function to copy queued messages to a sequential data set when the queue manager is not running, without destroying any messages in the original queues.

The scope of the SCOPY function is determined by the keyword that you specify in the first parameter. You can either copy all the messages from a named queue, or all the messages from all the queues on a named page set.

Use the complementary function, LOAD, to restore the messages to their respective queues.

To use the SCOPY function, DDname CSQP0000 must specify the data set with page set zero for the subsystem required.

Note: The SCOPY function does not operate on shared queues.
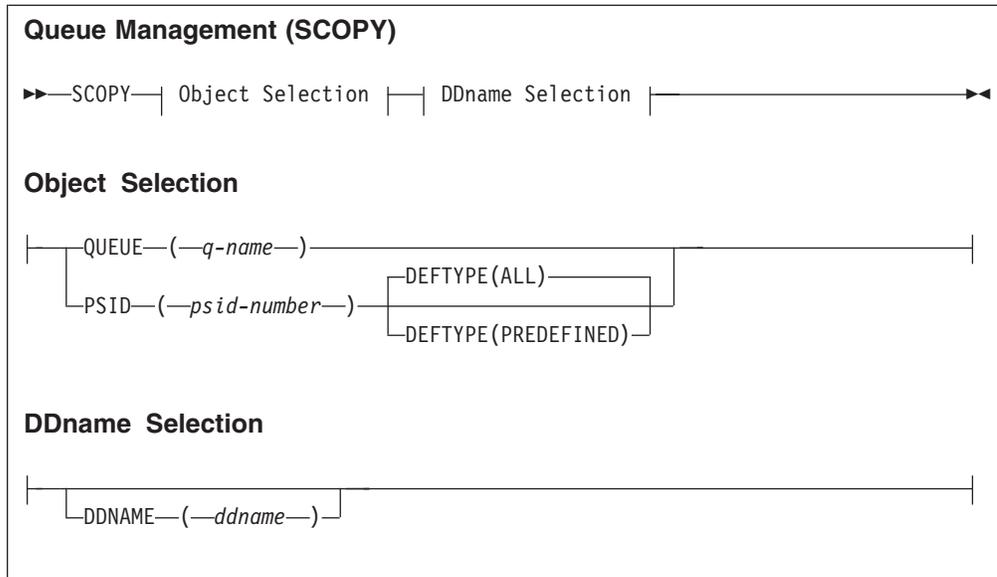
```
┌──────────────────────────────────────────────────────────────────────────┐
│  Queue Management (SCOPY)                                                   │
│                                                                             │
│  ►►──SCOPY──┤ Object Selection ├──┤ DDname Selection ├──────────────────►◄  │
│                                                                             │
│  Object Selection                                                           │
│                                                                             │
│  ├───┬──QUEUE──(──q-name──)────────────────────────────────────────────┬─┤ │
│      │                          ┌──DEFTYPE(ALL)─────────┐               │   │
│      └──PSID──(──psid-number──)──┤                       ├──────────────┘   │
│                                  └──DEFTYPE(PREDEFINED)──┘                   │
│                                                                             │
│  DDname Selection                                                           │
│                                                                             │
│  ├─────┬──────────────────────────────┬─────────────────────────────────┤  │
│        └──DDNAME──(──ddname──)─────────┘                                     │
│                                                                             │
└──────────────────────────────────────────────────────────────────────────┘
```

Keywords and parameters

QUEUE(*q-name*)
>   Specifies that messages in the named queue are to be copied. The keyword
>   QUEUE can be abbreviated to Q.
>
>   *q-name* specifies the name of the queue to be copied. This name is
>   case-sensitive.
>
>   DDname CSQP00*nn* must specify the data set with page set *nn* for the
>   subsystem required, where nn is the number of the page set where the queue
>   resides.

PSID(*psid-number*)
>   Specifies that all the messages in all the queues in the specified page set are to
>   be copied.
>
>   *psid-number* is the page set identifier, which specifies the page set to be used.
>   This identifier is a two-digit integer (whole number) representing a single page
>   set.
>
>   DDname CSQP00*psid-number* must specify the data set with the required page
>   set for the subsystem required.

DEFTYPE
>   Specifies whether to copy dynamic queues:
>
>   **ALL**   Copy all queues; this is the default.
>
>   **PREDEFINED**
>   >   Do not include dynamic queues; this is the same set of queues that are
>   >   selected by the COMMAND and SDEFS functions with the MAKEDEF
>   >   parameter.
>
>   This parameter is only valid if you specify PSID.

DDNAME(*ddname*)
>   Specifies that the messages are to be copied to a named data set. If this
>   keyword is omitted, the default DDname, CSQUOUT, is used. The keyword
>   DDname can be abbreviated to DD.

ddname specifies the DDname of the destination data set, which is used to store the messages. The record format of this data set must be variable block spanned (VBS).

Do not specify the same DDname on more than one SCOPY statement, unless its DD statement specifies a sequential data set with DISP=MOD.

Example

```
//SCOPY    EXEC PGM=CSQUTIL
//STEPLIB  DD   DISP=SHR,DSN=thlqual.SCSQANLE
//         DD   DISP=SHR,DSN=thlqual.SCSQAUTH
//OUTPUTA  DD DSN=SAMPLE.UTILITY.COPYA,DISP=(NEW,CATLG),
//           SPACE=(CYL,(5,1),RLSE),UNIT=SYSDA,
//           DCB=(RECFM=VBS,BLKSIZE=23200)
//CSQUOUT  DD DSN=SAMPLE.UTILITY.COPY3,DISP=(NEW,CATLG),
//           SPACE=(CYL,(5,1),RLSE),UNIT=SYSDA,
//           DCB=(RECFM=VBS,BLKSIZE=23200)
//CSQP0000 DD DISP=OLD,DSN=pageset.dsname0
//CSQP0003 DD DISP=OLD,DSN=pageset.dsname3
//CSQP0006 DD DISP=OLD,DSN=pageset.dsname6
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
* COPY WHOLE PAGE SET TO 'CSQUOUT'
SCOPY PSID(03)
* COPY ONE QUEUE TO 'OUTPUT' - QUEUE IS ON PAGE SET 6
SCOPY QUEUE(ABC123A) DDNAME(OUTPUTA)
/*
```

Figure 64. Sample JCL for the CSQUTIL SCOPY functions. The sample shows two instances of the SCOPY function—one SCOPY to the default DDNAME, CSQUOUT; the other to DDNAME OUTPUTA, which overrides CSQUOUT.

Usage notes

1. Do not use SCOPY for a queue manager that is running because results will be unpredictable. You can avoid doing this accidentally by using DISP=OLD in the page set DD statement.

2. When you use SCOPY, you do not need to specify a queue manager name.

3. If you want to operate on a range of page sets, repeat the SCOPY function for each page set.

4. The function operates only on local queues and only for persistent messages.

5. A SCOPY PSID function is considered successful only if it successfully copies all the queues on the page set that have messages; empty queues are ignored. If the page set has no queues with messages, the SCOPY function fails, and no data is written to the data set.

6. If you try to copy an empty queue explicitly by SCOPY QUEUE, data indicating this is written to the sequential data set, and the copy is considered to be a success. However, if you attempt to copy a nonexistent queue, the SCOPY function fails, and no data is written to the data set.

7. If the SCOPY function fails, no further CSQUTIL functions are attempted.

8. To use the SCOPY function more than once in the job, specify different DDnames and data sets for each invocation of the function, or specify a sequential data set and DISP=MOD in the DD statements.

Emptying a queue of all messages (EMPTY)

Use the EMPTY function to delete all messages from a named queue or all the queues on a page set. The queue manager must be running. The scope of the function is determined by the keyword that you specify in the first parameter.

Use this function with care. Only delete messages of which copies have already been made.

Note: See "Syncpoints" on page 183 for information about how to avoid problems with duplicate messages if this function fails.

```
Queue management (EMPTY)

►►──EMPTY──┤ Object Selection ├────────────────────────►◄


Object Selection

├──┬─QUEUE──(──q-name──)───┬──────────────────────────────────┤
   └─PSID──(──psid-number──)─┘
```
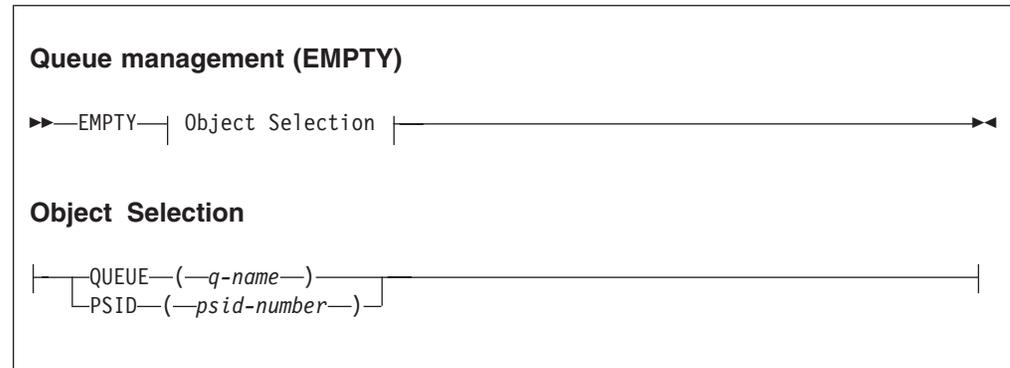
Keywords and parameters

You must specify the scope of the EMPTY function. Choose one of these:

QUEUE(*q-name*)
> Specifies that messages are to be deleted from a named queue. This keyword can be abbreviated to Q.

> *q-name* specifies the name of the queue from which messages are to be deleted. This name is case-sensitive.

PSID(*psid-number*)
> Specifies that all the messages are to be deleted from all queues in the named page set.

> *psid-number* specifies the page-set identifier. This identifier is a two-digit integer (whole number) representing a single page set.

Example

```
//EMPTY  EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD   DISP=SHR,DSN=thlqual.SCSQANLE
//        DD   DISP=SHR,DSN=thlqual.SCSQAUTH
//SYSPRINT DD SYSOUT=*
//SYSIN   DD *
EMPTY QUEUE(SPARE)
EMPTY PSID(66)
/*
```

Figure 65. Sample JCL for the CSQUTIL EMPTY function

Usage notes

1. The queues involved must not be in use when the function is invoked.
2. This function operates only on local queues.
3. If you want to operate on a range of page sets, repeat the EMPTY function for each page set.
4. You cannot empty the system-command input queue (SYSTEM.COMMAND.INPUT).
5. An EMPTY PSID function is considered successful only if it successfully empties all the queues on the page set.
6. If you empty a queue that is already empty (either explicitly by EMPTY QUEUE or because there are one or more empty queues on a page set that you are emptying), the EMPTY function is considered to be a success. However, if you attempt to empty a nonexistent queue, or a page set containing no queues, the EMPTY function fails.
7. If EMPTY fails or is forced to take a syncpoint, no further CSQUTIL functions are attempted.
8. You need the necessary authority to use the command server queues (SYSTEM.COMMAND.INPUT, SYSTEM.COMMAND.REPLY.MODEL, and SYSTEM.CSQUTIL.*), to use the DISPLAY QUEUE and DISPLAY STGCLASS MQSC commands, and to use the WebSphere MQ API to get messages from the queues that you want to empty.

Restoring messages from a data set to a queue (LOAD)

The LOAD function of CSQUTIL is complementary to the COPY or SCOPY function. LOAD restores messages from the destination data set of an earlier COPY or SCOPY operation. The queue manager must be running.

The data set can contain messages from one queue only if it was created by COPY or SCOPY QUEUE, or from a number of queues if it was created by COPY PSID or several successive COPY or SCOPY QUEUE operations. Messages are restored to queues with the same name as those from which they were copied. You can specify that the first or only queue is loaded to a queue with a different name. (This would normally be used with a data set created with a single COPY queue operation to restore the messages to a queue with a different name.)

Note: See "Syncpoints" on page 183 for information about how to avoid problems with duplicate messages if this function fails.

```
┌────────────────────────────────────────────────────────────────────────────┐
│  Queue management (LOAD)                                                      │
│                                                                              │
│  ►►──LOAD──┤ Object Selection ├──┤ DDname Selection ├──┤ Record Selection ├──►◄│
│                                                                              │
│  Object  Selection                                                           │
│                                                                              │
│  ├──┬──────────────────────┬──────────────────────────────────────────────┤ │
│     └─QUEUE──(──q-name──)──┘                                                  │
│                                                                              │
│  DDname  Selection                                                           │
│                                                                              │
│  ├──┬──────────────────────────┬──────────────────────────────────────────┤ │
│     └─DDNAME──(──ddname──)──┘                                                 │
│                                                                              │
│  Record  Selection                                                           │
│                                                                              │
│  ├──┬─────────────────────┬──┬─────────────────────────┬───────────────────┤ │
│     └─SKIPMSGS──(──n──)──┘    └─MSGCOUNT──(──m──)──┘                          │
│                                                                              │
└────────────────────────────────────────────────────────────────────────────┘
```

Keywords and parameters

QUEUE(*q-name*)

Specifies that the messages from the first or only queue on the destination data
set of a prior COPY or SCOPY operation are to be loaded to a named queue.
Messages from any subsequent queues are loaded to queues with the same
names as those they came from. The keyword QUEUE can be abbreviated to
Q.

q-name specifies the name of the queue to which the messages are to be loaded.
This name is case-sensitive. It must not be a model queue.

DDNAME(*ddname*)

Specifies that messages are to be loaded from a named data set. This keyword
can be abbreviated to DD.

ddname specifies the DDname that identifies the destination data set of a prior
COPY or SCOPY operation, from which the messages are to be loaded. This
name is not case-sensitive, and can be up to 8 characters long.

If you omit DDname(*ddname*) the default DDname, CSQUINP, is used.

SKIPMSGS(*n*)

Specifies that the first *n* messages in the sequential data set are to be skipped
before commencing the load of the queue.

If you omit SKIPMSGS(*n*) no messages will be skipped; the load will start at
the first message.

MSGCOUNT(*m*)

Specifies that only *m* messages are to be read from the data set and loaded to
the queue.

If you omit MSGCOUNT(*m*) the number of messages read is unlimited.

Example

```
//LOAD    EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD   DISP=SHR,DSN=thlqual.SCSQANLE
//        DD   DISP=SHR,DSN=thlqual.SCSQAUTH
//OUTPUTA  DD DSN=MY.UTILITY.OUTPUTA,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
LOAD QUEUE(ABC123) DDNAME(OUTPUTA)
/*
```

Figure 66. Sample JCL for the CSQUTIL LOAD function

Usage notes

1. To use the LOAD function, the queues or page sets involved must not be in use when the function is invoked.

2. If the data set contains multiple queues, the LOAD function is considered successful only if it successfully loads all the queues on the data set.

3. If LOAD fails, or is forced to take a syncpoint, no further CSQUTIL functions are attempted.

4. CSQUTIL uses MQPMO_SET_ALL_CONTEXT to ensure that the message descriptor fields remain the same as the original copy. It therefore needs an access of CONTROL in the queue's CONTEXT profile. For full details, see the section "Profiles For Context Security" in WebSphere MQ for z/OS System Setup Guide.

Migrating a channel initiator parameter module (XPARM)

In versions of WebSphere MQ for z/OS before Version 6.0, you could tailor the channel initiator by creating a channel initiator parameter load module. In Version 6.0, you do it by setting queue manager attributes. To make it easier to migrate to Version 6.0, this command generates an ALTER QMGR command from a pre-Version 6.0 channel initiator parameter module.

Migration (XPARM)

►►—XPARM—DDNAME(*ddname*)—MEMBER(*membername*)—MAKEALT(*ddname2*)————————►◄

Keywords and parameters

DDNAME(*ddname***)**
    Specifies that an ALTER QMGR command is to be generated from a channel initiator parameter module in this data set.

MEMBER(*membername***)**
    Specifies the name of the channel initiator parameter module in the data set specified by DDNAME(ddname2).

MAKEALT(*ddname2***)**
    Specifies the DDname that identifies the output data set in which the ALTER command is to be stored. The data set should be RECFM=FB, LRECL=80. This

data set can then be used as input for a later invocation of the COMMAND function or it can be incorporated into the CSQINP2 initialization input data sets.

Example

```
//MIGRATE1 EXEC PGM=CSQUTIL
//STEPLIB  DD DISP=SHR,DSN=thlqual.SCSQANLE
//         DD DISP=SHR,DSN=thlqual.SCSQAUTH
//CSQXPARM DD DISP=SHR,DSN=user.loadlib
//SYSPRINT DD SYSOUT=*
//ALTQMGR  DD DISP=OLD,DSN=user.commands(ALTQMGR)
//SYSIN DD *
XPARM DDNAME(CSQXPARM) MEMBER(MQ3AXPRM) MAKEALT(ALTQMGR)
/*
```

Figure 67. Sample JCL for the CSQUTIL XPARM function

The change log inventory utility (CSQJU003)

The WebSphere MQ change log inventory utility runs as a z/OS batch job to change the bootstrap data set (BSDS).

Through this utility, you can invoke these functions:

NEWLOG
> Add active or archive log data sets.

DELETE
> Delete active or archive log data sets.

ARCHIVE
> Supply passwords for archive logs.

CRESTART
> Control the next restart of WebSphere MQ.

CHECKPT
> Set checkpoint records.

HIGHRBA
> Update the highest written log RBA.

Only run this utility when WebSphere MQ is stopped. This is because the active log data sets named in the BSDS are dynamically added for exclusive use to WebSphere MQ and remain allocated exclusively to WebSphere MQ until it terminates.

Invoking the CSQJU003 utility

The utility runs as a z/OS batch program. Figure 68 on page 192 gives an example of the JCL required.

```
//JU003   EXEC  PGM=CSQJU003
//STEPLIB  DD   DISP=SHR,DSN=thlqual.SCSQANLE
//         DD   DISP=SHR,DSN=thlqual.SCSQAUTH
//SYSPRINT DD   SYSOUT=*,DCB=BLKSIZE=629
//SYSUT1   DD   DISP=SHR,DSN=bsds.dsname
//SYSIN    DD   *
  NEWLOG DSNAME=CSQREPAL.A0001187,COPY1VOL=CSQV04,UNIT=SYSDA,
  STARTRBA=3A190000,ENDRBA=3A1F0FFF,CATALOG=YES,PASSWORD=PASSWRD
/*
```

Figure 68. Sample JCL to invoke the CSQJU003 utility

Data definition (DD) statements

CSQJU003 requires DD statements with these DDnames:

SYSUT1
> This statement is required; it names the BSDS.

SYSUT2
> This statement is required if you use dual BSDSs; it names the second copy of the BSDS.

> **Dual BSDSs and CSQJU003**

> Each time you run the CSQJU003 utility, the BSDS time stamp field is updated with the current system time. If you run CSQJU003 separately for each copy of a dual copy BSDS, the time stamp fields are not synchronized, so the queue manager fails at startup, issuing error message CSQJ120E. Therefore, if CSQJU003 is used to update dual copy BSDSs, both BSDSs must be updated within a single run of CSQJU003.

SYSPRINT
> This statement is required; it names a data set for print output. The logical record length (LRECL) is 125. The block size (BLKSIZE) must be 629.

SYSIN
> This statement is required; it names the input data set for statements that specify what the utility is to do. The logical record length (LRECL) is 80.

You can use more than one statement of each type. In each statement, separate the operation name (NEWLOG, DELETE, ARCHIVE, CRESTART) from the first parameter by one or more blanks. You can use parameters in any order; separate them by commas with no blanks. Do not split a parameter description across two SYSIN records.

A statement containing an asterisk (*) in column 1 is considered to be a comment, and is ignored. However, it appears in the output listing. To include a comment or sequence number in a SYSIN record, separate it from the last comma by a blank. When a blank follows a comma, the rest of the record is ignored.

Multiple statement operation

When running CSQJU003, a significant error in any statement causes the control statements for the statement in error and all following statements to be skipped. Therefore, BSDS updates cannot occur for any operation specified in the statement in error, or any following statements. However, all the remaining statements are checked for syntax errors.

Adding information about a data set to the BSDS (NEWLOG)

The NEWLOG function declares one of the following data sets:

- A VSAM data set that is available for use as an active log data set.

  Use the keywords DSNAME, COPY1, COPY2, and PASSWORD.

- An active log data set that is replacing one that encountered an I/O error.

  Use the keywords DSNAME, COPY1, COPY2, STARTRBA, ENDRBA, and PASSWORD.

- An archive log data set volume.

  Use the keywords DSNAME, COPY1VOL, COPY2VOL, STARTRBA, ENDRBA, STRTLRSN, ENDLRSN, UNIT, CATALOG, and PASSWORD.

  In a queue-sharing group environment, you should always supply LRSN information. Run the print log map utility (CSQJU004) to find RBAs and LRSNs to use for archive log data sets.

A maximum of 31 data sets can be defined for each log copy, either by this NEWLOG function or the MQSC DEFINE LOG command.

```
  ┌─────────────────────────────────────────────────────────────────────────────┐
  │ NEWLOG                                                                        │
  │                                                                               │
  │ ►►──NEWLOG──DSNAME──=──dsname──┬─┤ New active log ├──┬──────────────────────► │
  │                                └─┤ New archive log ├─┘                        │
  │                                                                               │
  │ ►─┬──────────────────────────────┬──────────────────────────────────────────►◄│
  │   └─,──PASSWORD──=──password──────┘                                           │
  │                                                                               │
  │ New active log:                                                               │
  │                                                                               │
  │ ├─┬─,──COPY1─┬───────────────────────────────────────────────────────────────►│
  │   └─,──COPY2─┘                                                                 │
  │                                                                               │
  │ ►─┬─────────────────────────────────────────────────────────────────┬────────►│
  │   └─,──STARTRBA──=──startrba──,──ENDRBA──=──endrba──┤ Time ├──────────┘        │
  │                                                                               │
  │ Time:                                                                         │
  │                                                                               │
  │ ├──,──STARTIME──=──startime──,──ENDTIME──=──endtime─────────────────────────┤ │
  │                                                                               │
  │ New archive log:                                                              │
  │                                                                               │
  │ ├─┬─,──COPY1VOL──=──vol-id─┬─────────────────────────────────────────────────►│
  │   └─,──COPY2VOL──=──vol-id─┘                                                   │
  │                                                                               │
  │ ►─,──STARTRBA──=──startrba──,──ENDRBA──=──endrba─────────────────────────────►│
  │                                                                               │
  │ ►─┬──────────────────────────────────────────────────┬──────────────────────►│
  │   └─,──STARTIME──=──startime──,──ENDTIME──=──endtime──┘                        │
  │                                                                               │
  │ ►─┬──────────────────────────────────────────────┬──,──UNIT──=──unit-id──────►│
  │   └─,──STRTLRSN──=──strtlrsn──,──ENDLRSN──=──endlrsn─┘                          │
  │                                                                               │
  │    ┌─,CATALOG=NO──┐                                                           │
  │ ►──┤              ├───────────────────────────────────────────────────────────│
  │    └─,CATALOG=YES─┘                                                           │
  └─────────────────────────────────────────────────────────────────────────────┘
```

Keywords and parameters

DSNAME=*dsname*

Names a log data set.

dsname can be up to 44 characters long.

PASSWORD=*password*

Assigns a password to the data set. It is stored in the BSDS and subsequently used in any access to the active or archive log data sets.

The password is a data set password, and should follow standard VSAM convention: 1 through 8 alphanumeric characters (A through Z, 0 through 9) or special characters (& * + - . ; ' /).

We recommend that you use an ESM such as RACF® to provide your data set security requirements.

COPY1
Makes the data set an active log copy-1 data set.

COPY2
Makes the data set an active log copy-2 data set.

STARTRBA=_startrba_
Gives the log RBA (relative byte address within the log) of the beginning of the replacement active log data set or the archive log data set volume specified by DSNAME.

startrba is a hexadecimal number of up to 12 characters. The value must end with 000. If you use fewer than 12 characters, leading zeros are added. The RBA can be obtained from messages or by printing the log map.

ENDRBA=_endrba_
Gives the log RBA (relative byte address within the log) of the end of the replacement active log data set or the archive log data set volume specified by DSNAME.

endrba is a hexadecimal number of up to 12 characters. The value must end with FFF. If you use fewer than 12 characters, leading zeros are added.

STARTIME=_startime_
Start time of the RBA in the BSDS. This is an optional field. The time stamp format (with valid values in parentheses) is yyyydddhhmmsst, where:

yyyy   Indicates the year (1993 through 2099)

ddd   Indicates the day of the year (1 through 365; 366 in leap years)

hh   Indicates the hour (zero through 23)

mm   Indicates the minutes (zero through 59)

ss   Indicates the seconds (zero through 59)

t   Indicates tenths of a second

If fewer than 14 digits are specified for the STARTIME and ENDTIME parameter, trailing zeros are added.

STARTRBA is required when STARTIME is specified.

ENDTIME=_endtime_
End time of the RBA in the BSDS. This is an optional field. For time stamp format, see the STARTIME option. The ENDTIME value must be greater than or equal to the value of STARTIME.

STRTLRSN=_strtlrsn_
Gives the LRSN (logical record sequence number) of the first complete log record on the new archive data set.

strtlrsn is a hexadecimal number of up to 12 characters. If you use fewer than 12 characters, leading zeros are added.

ENDLRSN=_endlrsn_
Gives the LRSN (logical record sequence number) of the last last log record on the new archive data set.

endlrsn is a hexadecimal number of up to 12 characters. If you use fewer than 12 characters, leading zeros are added.

COPY1VOL=vol-id
The volume serial of the copy-1 archive log data set named after DSNAME.

COPY2VOL=vol-id
The volume serial of the copy-2 archive log data set named after DSNAME.

UNIT=unit-id
The device type of the archive log data set named after DSNAME.

CATALOG
Specifies whether the archive log data set is cataloged:

NO The archive log data set is not cataloged. All subsequent allocations of the data set are made using the unit and volume information specified on the function. This is the default.

YES The archive log data set is cataloged. A flag is set in the BSDS indicating this, and all subsequent allocations of the data set are made using the catalog.

WebSphere MQ requires that all archive log data sets on DASD be cataloged. Select CATALOG=YES if the archive log data set is on DASD.

Deleting information about a data set from the BSDS (DELETE)

Use the DELETE function to delete all information about a specified log data set or data set volume from the bootstrap data sets. For example, you can use this function to delete outdated archive log data sets.

DELETE

```
►►──DELETE──DSNAME──=──dsname──────────────────────────────────►◄
                        ├─,─COPY1VOL──=──vol-id─┤
                        └─,─COPY2VOL──=──vol-id─┘
```
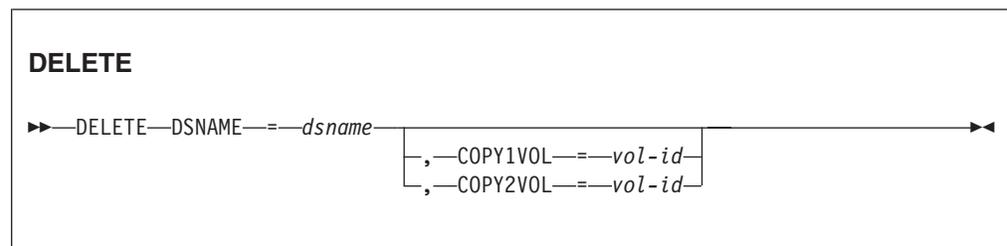
Keywords and parameters

DSNAME=dsname
Specifies the name of the log data set.

dsname can be up to 44 characters long.

COPY1VOL=vol-id
The volume serial number of the copy-1 archive log data set named after DSNAME.

COPY2VOL=vol-id
The volume serial number of the copy-2 archive log data set named after DSNAME.

Supplying a password for archive log data sets (ARCHIVE)

Use the ARCHIVE function to assign a password to all archive data sets created after this operation. This password is added to the installation's z/OS password data set each time a new archive log data set is created.

Use the NOPASSWD keyword to remove the password protection for all archives created after the archive operation.

Note: You should normally use an external security manager (ESM), such as RACF, if you want to implement security on any WebSphere MQ data sets.

```
ARCHIVE

►►──ARCHIVE──┬──PASSWORD──=──password──┬────────────────────────►◄
             └──NOPASSWD───────────────┘
```
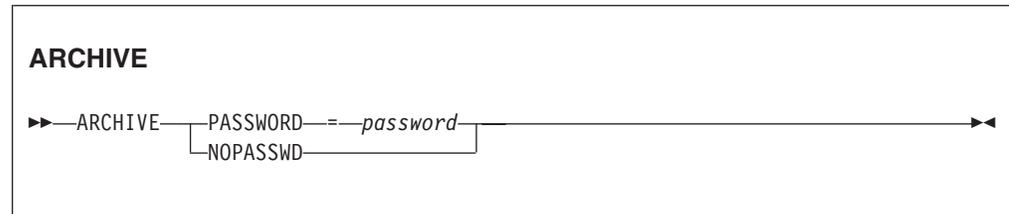
Keywords and parameters

PASSWORD=*password*
  Specifies that a password is to be assigned to the archive log data sets.

  password specifies the password, which is a data set password and it must follow the standard VSAM convention; that is, 1 through 8 alphanumeric characters (A through Z, 0 through 9) or special characters (& * + - . ; ' /).

NOPASSWD
  Specifies that archive password protection is not to be active for all archives created after this operation. No other keyword can be used with NOPASSWD.

Controlling the next restart (CRESTART)

Use the CRESTART function to control the next restart of the queue manager, either by creating a new conditional restart control record or by cancelling the one currently active. These records limit the scope of the log data that will be used during restart (truncating the log, in effect) . Any existing conditional restart control record governs every restart until one of these events occurs:

- A restart operation completes
- A CRESTART CANCEL is issued
- A new conditional restart control record is created

Attention: This can override WebSphere MQ efforts to maintain data in a consistent state. Only use this function when implementing the disaster recovery process described in "Recovering a single queue manager at an alternative site" on page 111 and "Recovering a queue-sharing group at the alternative site" on page 115, or under the guidance of IBM service.

```
  CRESTART

►►──CRESTART──┬─CREATE──┬─,ENDRBA=endrba──┬─────────────────────►◄
              │         └─,ENDLRSN=endlrsn─┘
              └─CANCEL──────────────────────
```
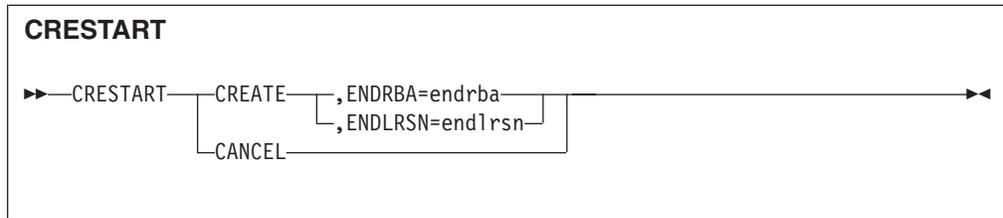
Keywords and parameters

CREATE
> Creates a new conditional restart control record. When the new record is created, the previous control record becomes inactive.

CANCEL
> Makes the currently active conditional restart control record inactive. The record remains in the BSDS as historical information.

> No other keyword can be used with CANCEL.

ENDRBA=_endrba_
> Gives the last RBA of the log to be used during restart (the point at which the log is to be truncated), and the starting RBA of the next active log to be written after restart. Any log information in the bootstrap data set and the active logs, with an RBA greater than _endrba_, is discarded.

> _endrba_ is a hexadecimal number of up to 12 digits. If you use fewer than 12 digits, leading zeros are added.

> The value of ENDRBA must be a multiple of 4096. (The hexadecimal value must end in 000.)

ENDLRSN=_endlrsn_
> Gives the LRSN of the last log record to be used during restart (the point at which the log is to be truncated). Any log information in the bootstrap data set and the active logs with an LRSN greater than `endlrsn` is discarded.

Setting checkpoint records (CHECKPT)

Use the CHECKPT function to add or delete a record in the BSDS checkpoint queue. Use the STARTRBA and ENDRBA keywords to add a record, or the STARTRBA and CANCEL keywords to delete a record.

Attention: This can override WebSphere MQ efforts to maintain data in a consistent state. Only use this function when implementing the disaster recovery process described in "Recovering a single queue manager at an alternative site" on page 111 and "Recovering a queue-sharing group at the alternative site" on page 115, or under the guidance of IBM service.

```
  CHECKPT

►►──CHECKPT──STARTRBA──=──startrba─────────────────────────────────►

►──┬─,──ENDRBA──=──offlrba──,──TIME──=──time─┬────►◄
   └─,──CANCEL─────────────────────────────────┘
```
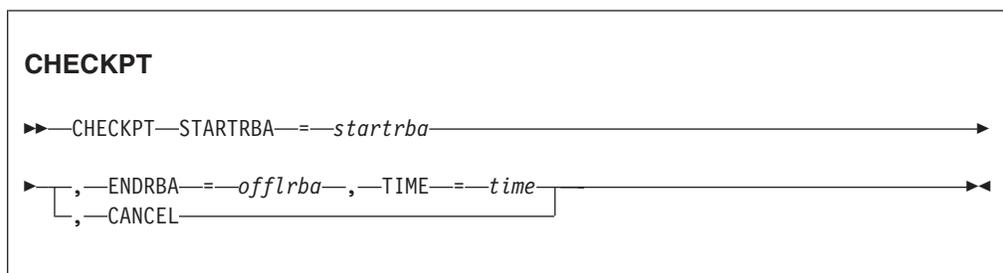
Keywords and parameters

STARTRBA=_startrba_
Indicates the start checkpoint log record.

startrba is a hexadecimal number of up to 12 digits. If you use fewer than 12 digits, leading zeros are added. The RBA can be obtained from messages or by printing the log map.

ENDRBA=_endrba_
Indicates the end checkpoint log record corresponding to the start checkpoint record.

endrba is a hexadecimal number of up to 12 digits. If you use fewer than 12 digits, leading zeros are added. The RBA can be obtained from messages or by printing the log map.

TIME=_time_
Gives the time the start checkpoint record was written. The time stamp format (with valid values in parentheses) is yyyydddhhmmsst, where:

yyyy   Indicates the year (1993 through 2099)

ddd   Indicates the day of the year (1 through 365; 366 in leap years)

hh   Indicates the hour (zero through 23)

mm   Indicates the minutes (zero through 59)

ss   Indicates the seconds (zero through 59)

t   Indicates tenths of a second

If fewer than 14 digits are specified for the TIME parameter, trailing zeros are added.

CANCEL
Deletes the checkpoint queue record containing a starting RBA that matches the RBA specified by STARTRBA.

Updating the highest written log RBA (HIGHRBA)

Use the HIGHRBA function to update the highest written log RBA recorded in the BSDS for either the active or archive log data sets. Use the STARTRBA keyword to update the active log, and the OFFLRBA keyword to update the archive log.

Attention: This can override WebSphere MQ efforts to maintain data in a consistent state. Only use this function when implementing the disaster recovery process described in "Recovering a single queue manager at an alternative site" on page 111, or under the guidance of IBM service.

```
 HIGHRBA


▶▶──HIGHRBA──────────────────────────────────────────────────────▶

▶──┬─STARTRBA──=──startrba─┬──────────────────────┬──,──TIME──=──time─┬──▶◀
   │                       └─,──OFFLRBA──=──offlrba─┘                 │
   └─OFFLRBA──=──offlrba────────────────────────────────────────────┘
```
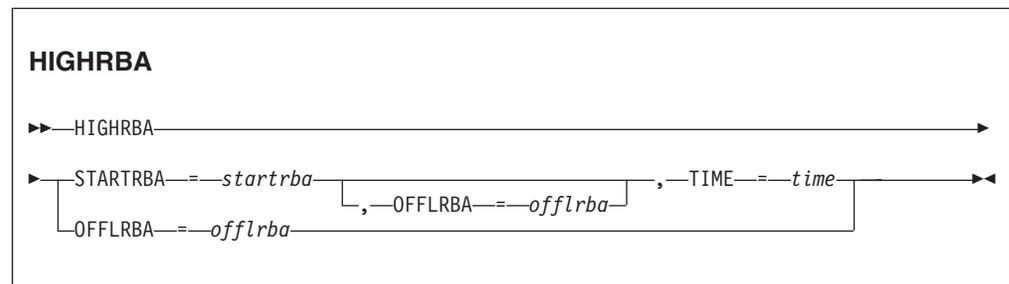
Keywords and parameters

STARTRBA=_startrba_
>Indicates the log RBA of the highest written log record in the active log data set.
>
>_startrba_ is a hexadecimal number of up to 12 digits. If you use fewer than 12 digits, leading zeros are added. The RBA can be obtained from messages or by printing the log map.

TIME=_time_
>Specifies when the log record with the highest RBA was written to the log. The time stamp format (with valid values in parentheses) is yyyydddhhmmsst, where:

>| | |
>|---|---|
>| **yyyy** | Indicates the year (1993 through 2099) |
>| **ddd** | Indicates the day of the year (1 through 365; 366 in leap years) |
>| **hh** | Indicates the hour (zero through 23) |
>| **mm** | Indicates the minutes (zero through 59) |
>| **ss** | Indicates the seconds (zero through 59) |
>| **t** | Indicates tenths of a second |

>If fewer than 14 digits are specified for the TIME parameter, trailing zeros are added.

OFFLRBA=_offlrba_
>Specifies the highest off-loaded RBA in the archive log.
>
>_offlrba_ is a hexadecimal number of up to 12 digits. If you use fewer than 12 digits, leading zeros are added. The value must end with hexadecimal 'FFF'.

The print log map utility (CSQJU004)

The WebSphere MQ print log map utility runs as a z/OS batch program to list the following information:
- Log data set name and log RBA association for both copies of all active and archive log data sets
- Active log data sets available for new log data
- Contents of the queue of checkpoint records in the bootstrap data set (BSDS)
- Contents of the quiesce history record
- System and utility time stamps
- Passwords for the active and archive log data sets, if provided

You can run the CSQJU004 program regardless of whether the queue manager is running. However, if the queue manager is running, consistent results from the utility can be ensured only if both the utility and the queue manager are running under control of the same z/OS system.

To use this utility, the user ID of the job must have the requisite security authorization, or, if the BSDS is password protected, the appropriate VSAM password for the data set.

Invoking the CSQJU004 utility

Figure 69 shows an example of the JCL used to invoke the CSQJU004 utility:

```
//JU004    EXEC  PGM=CSQJU004
//STEPLIB DD   DISP=SHR,DSN=thlqual.SCSQANLE
//        DD   DISP=SHR,DSN=thlqual.SCSQAUTH
//SYSPRINT DD  SYSOUT=*,DCB=BLKSIZE=629
//SYSUT1   DD  DISP=SHR,DSN=bsds.dsname
```

Figure 69. Sample JCL to invoke the CSQJU004 utility

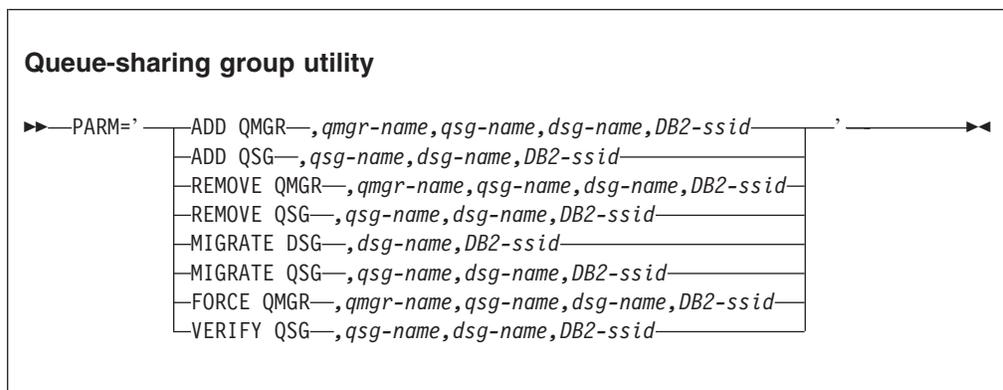### Data definition statements

The CSQJU004 utility requires DD statements with the following DDnames:

SYSUT1
> This statement is required to specify and allocate the bootstrap data set. If the BSDS must be shared with a concurrently running queue manager subsystem, use DISP=SHR on the DD statement.

SYSPRINT
> This statement is required to specify a data set or print spool class for print output. The logical record length (LRECL) is 125. The block size (BLKSIZE) must be 629.

"Finding out what the BSDS contains" on page 81 describes the output.

The log print utility (CSQ1LOGP)

Use this utility to print information contained in the WebSphere MQ log data sets or the BSDS.

Invoking the CSQ1LOGP utility

How to invoke the CSQ1LOGP log print utility.

You run the WebSphere MQ log print utility as a z/OS batch program. You can specify:
- A bootstrap data set (BSDS)
- Active log data sets (with no BSDS)
- Archive log data sets (with no BSDS)

Sample JCL to invoke the CSQ1LOGP utility is shown in figures Figure 70 on page 202, Figure 71 on page 203, Figure 72 on page 203 and Figure 73 on page 203.

These DD statements should be provided:

SYSPRINT
> All error messages, exception conditions and the detail report are written to this data set. The logical record length (LRECL) is 131.

SYSIN
> Input selection criteria can be specified in this data set (see "Input control parameters" on page 203 for more information).

The logical record length (LRECL) must be 80, but only columns 1 through 72 are significant; columns 73 through 80 are ignored. At most 50 records can be used. Records with an asterisk (*) in column 1 are interpreted as comments and are ignored.

SYSSUMRY
If a summary report is requested, by specifying the parameter **SUMMARY***(YES)* or **SUMMARY***(ONLY)*, the output is written to this data set. The logical record length (LRECL) is 131.

BSDS Name of the bootstrap data set (BSDS).

ACTIVEn
Name of an active log data set you want to print (n=number).

ARCHIVE
Name of an archive log data set you want to print.

If you specify the keyword **EXTRACT***(YES)*, provide one or more of the following DD statements, depending on what types of data you want to extract. Do not specify an LRECL, as it is set internally by the utility.

CSQBACK
This data set will contain persistent messages written to the log by units of work that were rolled back during the log range specified.

CSQCMT
This data set contains persistent messages written to the log by units of work that were committed during the log range specified

CSQBOTH
This data set contains persistent messages written to the log by units of work that were either committed or rolled back during the log range specified.

CSQINFLT
This data set contains persistent messages written to the log by units of work that remained inflight during the log range specified.

CSQOBJS
This data set contains information about object alterations that occurred during the log range specified.

If you are processing active log data sets, the utility runs even if WebSphere MQ is running, provided that the BSDS and active log data sets are defined using at least SHAREOPTIONS(2 3).

```
//PRTLOG EXEC PGM=CSQ1LOGP
//STEPLIB  DD DISP=SHR,DSN=thlqual.SCSQANLE
//         DD DISP=SHR,DSN=thlqual.SCSQLOAD
//BSDS     DD DSN=qmgr.bsds.dsname,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSSUMRY DD SYSOUT=*
//SYSIN    DD *
* extract records for pageset 3. Produce both summary and detail reports
PAGESET(3)
SUMMARY(YES)
/*
```

Figure 70. Sample JCL to invoke the CSQ1LOGP utility using a BSDS

```
//PRTLOG   EXEC PGM=CSQ1LOGP
//STEPLIB  DD DISP=SHR,DSN=thlqual.SCSQANLE
//         DD DISP=SHR,DSN=thlqual.SCSQLOAD
//ACTIVE1  DD DSN=qmgr.logcopy1.ds01,DISP=SHR
//ACTIVE2  DD DSN=qmgr.logcopy1.ds02,DISP=SHR
//ACTIVE3  DD DSN=qmgr.logcopy1.ds03,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSSUMRY DD SYSOUT=*
//SYSIN DD *
  insert your input control statements here
/*
```

Figure 71. Sample JCL to invoke the CSQ1LOGP utility using active log data sets

```
//PRTLOG   EXEC PGM=CSQ1LOGP
//STEPLIB  DD DISP=SHR,DSN=thlqual.SCSQANLE
//         DD DISP=SHR,DSN=thlqual.SCSQLOAD
//ARCHIVE  DD DSN=qmgr.archive1.ds01,DISP=SHR
//         DD DSN=qmgr.archive1.ds02,DISP=SHR
//         DD DSN=qmgr.archive1.ds03,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSSUMRY DD SYSOUT=*
//SYSIN    DD *
  insert your input control statements here
/*
```

Figure 72. Sample JCL to invoke the CSQ1LOGP utility using archive log data sets

```
//PRTLOG   EXEC PGM=CSQ1LOGP
...
//CSQBACK   DD DSN=backout.dataset,DISP=(NEW,CATLG)
//CSQCMT    DD DSN=commit.dataset,DISP=(NEW,CATLG)
//CSQBOTH   DD DSN=both.dataset,DISP=(NEW,CATLG)
//CSQINFLT  DD DSN=inflight.dataset,DISP=(NEW,CATLG)
//CSQOBJS   DD DSN=objects.dataset,DISP=(NEW,CATLG)
```

Figure 73. Sample JCL showing additional statements for the EXTRACT keyword

Input control parameters

The keywords that you can use in the SYSIN data set are described below.

You can specify various selection criteria to limit the log records that are processed. These are:
- log range, using RBASTART-RBAEND or LRSNSTART-LRSNEND
- page sets, using PAGESET
- units of recovery, using URID
- record contents, using DATA
- resource manager, using RM

Different types of selection criteria can be combined; only records meeting all the criteria are processed.

LRSNSTART*(hexadecimal-constant)*
Specifies the logical record sequence number (LRSN) from which to begin processing. You cannot use this keyword together with RBASTART. Use this keyword only if your queue manager is in a queue-sharing group.

LRSN values are always greater than A00000000000; this value is used as the start value if a lower value is specified.

You can also use the forms STARTLRSN or STRTLRSN or LRSNSTRT. Specify this keyword only once.

LRSNEND*(hexadecimal-constant)*
Specifies the logical record sequence number (LRSN) of the last record to be scanned. The default is FFFFFFFFFFFF (the end of the data sets). You can use this keyword only with LRSNSTART.

You can also use the form ENDLRSN.

Specify this keyword only once.

RBASTART*(hexadecimal-constant)*
Specifies the log RBA from which to begin processing. You cannot use this keyword together with LRSNSTART.

You can also use the forms STARTRBA or ST. Specify this keyword only once.

RBAEND*(hexadecimal-constant)*
Specifies the last valid log RBA that is to be processed. If this keyword is omitted, processing continues to the end of the log (FFFFFFFFFFFF). You can use this keyword only with RBASTART.

You can also use the forms ENDRBA or EN. Specify this keyword only once.

PAGESET*(decimal-integer)*
Specifies a page set identifier. The number should be in the range 00 through 99. You can specify a maximum of 10 PAGESET keywords. If PAGESET keywords are specified, only log records associated with the page sets you specify are processed.

URID *(hexadecimal-constant)*
Specifies a hexadecimal unit of recovery identifier. Changes to data occur in the context of a WebSphere MQ unit of recovery. A unit of recovery is identified on the log by a BEGIN UR record. The log RBA of that BEGIN UR record is the URID value you must use. If you know the URID for a given UR that you are interested in, you can limit the extraction of information from the log to that URID.

The hexadecimal constant can consist of 1 through 12 characters (6 bytes), and leading zeros are not required.

You can specify a maximum of 10 URID keywords.

DATA *(hexadecimal-string)*
Specifies a data string in hexadecimal.

The string can consist of 2 through 48 characters (24 bytes), and must have an even number of characters.

You can specify a maximum of 10 DATA keywords.

If DATA keywords are specified, only log records that contain at least one of the strings are processed.

RM *(resource_manager)*
Specifies a particular resource manager. Only records associated with this resource manager are processed. Valid values for this keyword are:

RECOVERY
Recovery log manager

DATA Data manager

BUFFER
> Buffer manager

IMSBRIDGE
> IMS bridge

SUMMARY*(YES|NO|ONLY)*
> Specifies whether a summary report is to be produced or not:
>
> **YES**   Produce a summary report in addition to the detail report.
>
> **NO**   Do not produce a summary report.
>
> **ONLY** Produce only a summary report (no detail report).
>
> The default is NO.

EXTRACT*(YES|NO)*
> Specifying EXTRACT(YES) will cause each log record that meets the input selection criteria to be written to the appropriate output file, as explained on page "The EXTRACT function." The default is NO.

Usage notes

1. If your queue manager is in a queue-sharing group, you can specify the log range required by either LRSNSTART (optionally with LRSNEND) or RBASTART (optionally with RBAEND). You cannot mix LRSN and RBA specifications.

   If you need to coordinate the log information from the different queue managers in the queue-sharing group, you should use LRSN specifications.

2. If your queue manager is not in a queue-sharing group, you cannot use LRSN specifications; you must use RBA specifications.

3. If you are using a BSDS, RBASTART or LRSNSTART must be specified.

4. CSQ1LOGP starts its processing on the first record containing an LRSN or RBA value greater than or equal to the value specified on LRSNSTART or RBASTART.

5. Normally you are only interested in the most recent additions to the log. Take care to choose a suitable value for the start of the log range, and do not use the defaults. Otherwise, you create an enormous amount of data, most of which is of no interest to you.

The EXTRACT function

Details of the EXTRACT function of CSQ1LOGP.

Typical uses of the **EXTRACT** parameter are to:

- Review which persistent messages were put to or got from a queue and whether the request was committed. This allows messages to be replayed.
- Review persistent messages that were put or got, but the request was backed out.
- Display which applications backed out rather than committed.
- Discover the volume of persistent data processed by queues, to identify the high use queues.
- Identify which applications set object attributes.
- Recreate object definitions for recovery purposes after a major failure.

When CSQ1LOGP with the **EXTRACT** parameter set is run against a log data set it processes all records in the data set, or all those within a specified range. Processing is as follows:

1. When a commit request is found, if the CSQCMT ddname is present then the data is written to this data set. If the CSQBOTH ddname is present the data is also written to this data set.

2. When a backout request is found, if the CSQBACK ddname is present then the data is written to this data set. If the CSQBOTH ddname is present the data is also written to this data set.

3. When changes to objects are detected, the information is written to the data set identified by the CSQOBJS ddname.

4. When the last record has been processed, information about remaining units of work is written to the data set identified by the CSQINFLT ddname.

If you do not want to collect one or more of these classes of information, then omit the appropriate DD statements.

Processing EXTRACT data

This section gives an example of processing the data returned from CSQ1LOGP with EXTRACT(YES) set, and details of the sample program supplied.

Example 1. Counting the number of bytes put to each queue:

The following job uses DFSORT™ facilities to process the file of committed records to add up the number of bytes put to each queue.

```
//TOOLRUN EXEC PGM=ICETOOL,REGION=1024K
//TOOLMSG  DD SYSOUT=*
//DFSMSG   DD SYSOUT=*
//TOOLIN   DD *
SORT FROM(IN) TO(TEMP1) USING(CTL1)
DISPLAY FROM(TEMP1) LIST(OUT1) ON(5,48,CH) ON(53,4,BI)
/*
//CTL1     DD *
* Select the records which were put
INCLUDE COND=(175,5,CH,EQ,C'MQPUT')
* Sort by queue name
SORT FIELDS=(110,48,CH,A)
* Only copy the queue name and size of user data to output record
OUTREC FIELDS=(1,4,110,48,102,4)
* Add up the number of bytes processed
SUM FIELDS=(102,4,FI)
/*
//IN       DD DISP=SHR,DSN=commit.dataset
//TEMP1    DD DISP=(NEW,DELETE),DSN=&TEMP1,SPACE=(CYL,(10,10))
//OUT1     DD SYSOUT=*
```

Figure 74. Accumulating bytes put to each queue

This produces output in the following format:

BA1   3605616

BA10   3572328

BA2   3612624

BA3   3579336

BA4   3572328

BA5	3491736
BA6	3574080
BA7	3532032
BA8	3577584
BA9	3539040
SYSTEM.ADMIN.CHANNEL.EVENT	186120
SYSTEM.ADMIN.QMGR.EVENT	384
SYSTEM.CHANNEL.SYNCQ	46488312

The following table lists the samples that are provided to allow you to print and interpret the data generated when EXTRACT(YES) is used

Sample	Description
thlqual.SCSQLOAD(CSQ4LOGS)	Sample C program to report on the UOW activity and object manipulation
thlqual.SCSQC37S(CSQ4LOGS)	Source for sample C program
thlqual.SCSQC370(CSQ4LOGD)	C header file to map records generated when using the EXTRACT(YES) function of CSQ1LOGP
thlqual.SCSQPROC(CSQ4LOGJ)	Sample JCL to run program CSQ4LOGS

Output

Detail report

The detail report begins by echoing the input selection criteria specified by SYSIN, and then prints each valid log record encountered. Definitions of keywords in the detail report are as follows:

RM Resource manager that wrote the log record.

TYPE Type of log record.

URID BEGIN UR for this unit of recovery, see the description above.

LRID Logical record identifier in the form: `AAAAAAAA.BBBBBBCC` where:

> **AAAAAAAA**
> Is the page set number.
>
> **BBBBBB**
> Is the relative page number in the page set.
>
> **CC** Is the relative record number on the page.

LRSN Logical record sequence number (LRSN) of the log record scanned.

SUBTYPE
Subtype of the log record type.

CHANGE LENGTH
Length of the logged change.

CHANGE OFFSET
Start position of the change.

BACKWARD CHAIN
Pointer to the previous page.

FORWARD CHAIN
> Pointer to the next page.

RECORD LENGTH
> Length of the inserted record.

Record layouts for the output data sets

The data sets produced when the **EXTRACT** keyword is specified contain information about persistent messages. Messages are identified by their queue name and an eight character key. Once a message has been got, the key can be reused by another message, so it is important to ensure that time sequence is maintained. In the records are times. A time stamp can only be extracted from a Begin-UR record or from an MQPUT request. Thus if there is only a long running transaction which is getting messages, the times when the gets occurred will all be the time the transaction started (the Begin-UR record). If there are many short units of work, or many messages being put, the time is reasonably accurate (within milliseconds), otherwise the times will become less and less accurate.

The information in the data sets has the following layout: (Note there is a 4 byte Record Descriptor Word at the front of each record because the files are Variable Blocked format). The field names correspond to those in the C header file CSQ4LOGD in thlqual.SCSQC370.

Offset		Type	Length	Name	Description
Dec	Hex				
0	0	Character	21	csrecorddate	The approximate time the log was written, in the format yyyy.ddd hh:mm:ss.thm
21	15	Character	7	cstimedelta	Approximate time difference in milliseconds from the start of the unit of work. Right-justified and padded with blanks.
28	1C	64-bit integer	8	dtodout	Estimated time that the log record was created, in STCK format.
36	24	Character	6	csurid	Queue manager specific unique identifier of the unit of work that created the log record.
42	2A	Character	12	cscorrelator	Thread correlation identifier
54	36	Character	8	csauth	Authorization identifier (Userid associated with unit of work)
62	3E	64-bit integer	8	dtime	Time that the unit of work was started, in STCK format
70	46	Character	8	csresource	Resource name
78	4E	Character	8	cscnty	Connection type: one of BATCH, RRSBATCH, IMS, CICS, CHIN or nulls for an internal task
86	56	Character	8	cscnid	Connection ID of thread that created this unit of work
94	5E	Character	3	csstatus	Unit of work type: BUR for begin or CP for checkpoint information
97	61	Integer	4	ldatalen	Length of the message data (if any)
101	65	Character	4	csqmgrname	Name of queue manager

105	69	Character	48	csqueuename	Name of queue, for get, put, or expired messages
153	99	Character	12	cssqdmcp	Shared queue message key. Blank if not a shared queue
165	A5	Character	8	csdmcp	Non-shared queue message key. Blank if a shared queue.
173	AD	Character	8	csverb	Activity: **ALTER** the object was changed **DEFINE** the object was created **MQGET** the message was got **MQPUT** the message was put **EXPIRE** the message expired **ABORT2** the message was backed out **PHASE1** the first phase of two phase commit **PHASE2** the second phase of two phase commit, or the only phase of one phase commit
181	B5	Character	1	cscmitstatus	Status of unit of work: **B** backed out **C** committed **I** inflight
182	B6	Character	1	csshunt	Shunted indicator: **S** shunted record **N** not shunted
183	B7	Character	6	cslogrba	RBA of log record
189	BD	Character	6	csshuntrba	RBA of shunted log record
195	C3	Character	1	csuowscope	UOW scope in hexadecimal: **01** local **02** shared
196	C4	Integer	4	lsegment	The segment number of the data, starting from 1.
200	C8		Variable		Data part
200	C8	Character	1	csbora	If csverb is ALTER, indicates whether the data is the 'before' or 'after' copy of the object. **B** before **A** after

201	C9	Character	Variable	csvardata	Message or object data. Length as given in ldatalen.

The queue-sharing group utility (CSQ5PQSG)

Use the CSQ5PQSG utility program to add queue-sharing group and queue manager definitions to the WebSphere MQ DB2 tables, and to remove them.

The CSQ5PQSG utility can also be used to verify the consistency of DB2 object definitions for queue manager, CF structure, and shared queue objects, within a queue-sharing group.

Invoking the queue-sharing group utility

Figure 75 shows an example of the JCL used to invoke the CSQ5PQSG utility.

```
//S001    EXEC PGM=CSQ5PQSG,REGION=4M,
//      PARM='function,function parameters'
//STEPLIB  DD DSN=thlqual.SCSQANLE,DISP=SHR
//         DD DSN=thlqual.SCSQAUTH,DISP=SHR
//         DD DSN=db2qual.SDSNLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
```

Figure 75. Sample JCL to invoke the CSQ5PQSG utility

Data definition statements

The CSQ5PQSG utility requires data definition statements with the following DDname:

SYSPRINT
> This statement is required; it names the data set for print output. The logical record length (LRECL) is 125.

Keywords and parameters

Queue-sharing group utility

```
▶▶──PARM='──┬─ADD QMGR──,qmgr-name,qsg-name,dsg-name,DB2-ssid──────┬──'──────────▶◀
            ├─ADD QSG──,qsg-name,dsg-name,DB2-ssid─────────────────┤
            ├─REMOVE QMGR──,qmgr-name,qsg-name,dsg-name,DB2-ssid───┤
            ├─REMOVE QSG──,qsg-name,dsg-name,DB2-ssid──────────────┤
            ├─MIGRATE DSG──,dsg-name,DB2-ssid──────────────────────┤
            ├─MIGRATE QSG──,qsg-name,dsg-name,DB2-ssid─────────────┤
            ├─FORCE QMGR──,qmgr-name,qsg-name,dsg-name,DB2-ssid────┤
            └─VERIFY QSG──,qsg-name,dsg-name,DB2-ssid──────────────┘
```

A queue-sharing group name (*qsg-name*) can have up to 4 characters, comprising uppercase A-Z, 0-9, $, #, @. It must not start with a numeric. For implemtation reasons, names of less than 4 characters are padded internally with @ symbols, so do not use names ending in @.

PARM

This field contains the function request followed by the function-specific parameters. These are described below:

ADD QMGR

Add a queue manager record into the CSQ.ADMIN_B_QMGR table. This only completes successfully if a corresponding queue-sharing group record already exists in the CSQ.ADMIN_B_QSG table and the queue manager entry does not already exist in the CSQ.ADMIN_B_QMGR table as the member of a different queue-sharing group.

qmgr-name	The queue manager name
qsg-name	The queue-sharing group name
dsg-name	The DB2 data-sharing group name
DB2-ssid	The DB2 subsystem ID

ADD QSG

Add a queue-sharing group record into the CSQ.ADMIN_B_QSG table.

qsg-name	The queue-sharing group name
dsg-name	The DB2 data-sharing group name
DB2-ssid	The DB2 subsystem ID

REMOVE QMGR

Remove a queue manager record from the CSQ.ADMIN_B_QMGR table. This only completes successfully if the queue manager has either never been started, or terminated normally from its last execution.

qmgr-name	The queue manager name
qsg-name	The queue-sharing group name
dsg-name	The DB2 data-sharing group name
DB2-ssid	The DB2 subsystem ID

REMOVE QSG

Remove a queue-sharing group record from the CSQ.ADMIN_B_QSG table. This only completes successfully if no queue managers are defined to the queue-sharing group.

qsg-name	The queue-sharing group name
dsg-name	The DB2 data-sharing group name
DB2-ssid	The DB2 subsystem ID

MIGRATE DSG

Check that the installation is ready to migrate its DB2 data-sharing group table definitions from WebSphere MQ Version 5.3 or 5.3.1 to WebSphere MQ Version 6.0.

It verifies that:

- All the queue managers in the data-sharing group have applied the migration & coexistence PTF and have since been started.
- The data-sharing group has not already been migrated.

dsg-name	The DB2 data-sharing group name
DB2-ssid	The DB2 subsystem ID

This function does not actually do the migration, which involves several steps. It is included as part of the sample migration job CSQ45ATB in thlqual SCSQPROC.

Migration requires that a migration PTF is installed on **all** queue managers in the data-sharing group.

For more details about migration to Version 6.0 and the migration & coexistence PTF, see the WebSphere MQ for z/OS System Setup Guide.

MIGRATE QSG

Check that the installation is ready to migrate its DB2 queue-sharing group table definitions from WebSphere MQ Version 5.3 or Version 5.3.1 to WebSphere MQ Version 6.0.

The MIGRATE QSG and MIGRATE DSG functions perform exactly the same function. The only difference is in the scope of the processing. MIGRATE QSG works on a single queue-sharing group only, MIGRATE DSG works on all queue-sharing groups that are defined within the data-sharing group.

It verifies that:
- All the queue managers in the queue-sharing group have applied the migration & coexistence PTF and have since been started.

qsg-name	The queue-sharing group name
dsg-name	The DB2 data-sharing group name
DB2-ssid	The DB2 subsystem ID

This function does not actually do the migration, which involves several steps. It is included as part of the sample migration job CSQ45ATB in thlqual SCSQPROC.

Migration requires that a migration PTF is installed on **all** queue managers in the queue-sharing group.

For more details about migration to Version 6.0 and the migration & coexistence PTF, see the WebSphere MQ for z/OS System Setup Guide.

FORCE QMGR

Remove a queue manager record from the CSQ.ADMIN_B_QMGR table, even if the queue manager has terminated abnormally.

Attention: This can override WebSphere MQ efforts to maintain data in a consistent state. Only use this function when you cannot carry out the procedure for removing a queue manager from a queue sharing group on page "Removing a queue manager from a queue-sharing group" on page 101.

qmgr-name	The queue manager name
qsg-name	The queue-sharing group name
dsg-name	The DB2 data-sharing group name

DB2-ssid	The DB2 subsystem ID

VERIFY QSG

Validate the consistency of the DB2 object definitions for queue manager, CF structure, and shared queue objects, within the queue-sharing group.

qsg-name	The queue-sharing group name
dsg-name	The DB2 data-sharing group name
DB2-ssid	The DB2 subsystem ID

Example

The following sample JCL adds an entry for queue manager QM01 into queue-sharing group QSG1. It specifies a connection to DB2 subsystem DB2A, which is a member of DB2 data-sharing group DSN510PG.

```
//S001   EXEC PGM=CSQ5PQSG,REGION=4M,
//     PARM='ADD QMGR,QM01,QSG1,DSN510PG,DB2A'
//STEPLIB  DD DSN=thlqual.SCSQANLE,DISP=SHR
//         DD DSN=thlqual.SCSQAUTH,DISP=SHR
//         DD DSN=db2qual.SDSNLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
```

Figure 76. Using the queue-sharing group utility to add a queue manager into a queue-sharing group

The active log preformat utility (CSQJUFMT)

Use the CSQJUFMT utility to format active log data sets before they are used by a queue manager. If the active log data sets are preformatted by the utility, log write performance is improved on the queue manager's first pass through the active logs. If the utility is not used, the queue manager must format each log control interval at log write time before it is used. On the second and subsequent passes through the active log data sets, the log control intervals already contain data, so need no further formatting, and no performance benefit accrues.

Invoking the CSQJUFMT utility

You can only run the CSQJUFMT program before starting the queue manager that will use the logs.

Note: Do not use this utility to format a log data set after the queue manager has started, or data will be lost.

These DD statements should be provided:

SYSPRINT

This statement is required to specify a data set or print spool class for print output.

SYSUT1
This statement identifies the log data set to be preformatted.

```
//JOBLIB    DD   DISP=SHR,DSN=thlqual.SCSQANLE
//          DD   DISP=SHR,DSN=thlqual.SCSQAUTH
//*
//JUFMT11   EXEC PGM=CSQJUFMT
//SYSPRINT  DD   SYSOUT=*
//SYSUT1    DD   DISP=OLD,DSN=hlq.LOGCOPY1.DS01
//*
//JUFMT21   EXEC PGM=CSQJUFMT
//SYSPRINT  DD   SYSOUT=*
//SYSUT1    DD   DISP=OLD,DSN=hlq.LOGCOPY2.DS01
```

Figure 77. Example of the JCL used to invoke the CSQJUFMT utility

Sample JCL is supplied in thlqual.SCSQPROC (CSQ4LFMT) for preformatting a newly-defined dual log data set. It contains two steps, one step to format each of the copies of the log data set.

The dead-letter queue handler utility (CSQUDLQH)

A *dead-letter queue* (DLQ) is a holding queue for messages that cannot be delivered to their destination queues. Every queue manager in a network should have an associated DLQ.

Queue managers, message channel agents, and applications can put messages on the DLQ. All messages on the DLQ should be prefixed with a *dead-letter header* structure, MQDLH. Messages put on the DLQ by a queue manager or by a message channel agent always have a dead-letter header; ensure that applications putting messages on the DLQ supply a dead-letter header as well. The *Reason* field of the MQDLH structure contains a reason code that identifies why the message is on the DLQ.

There should be a routine that runs regularly to process messages on the DLQ. WebSphere MQ supplies a default *dead-letter queue handler* (DLQ handler) called CSQUDLQH. A user-written *rules table* supplies instructions to the DLQ handler, for processing messages on the DLQ. That is, the DLQ handler matches messages on the DLQ against entries in the rules table. When a DLQ message matches an entry in the rules table, the DLQ handler performs the action associated with that entry.

Invoking the DLQ handler

The CSQUDLQH utility program runs as a z/OS batch program. Specify the name of the dead-letter queue that you want to process and the queue manager on which it resides. You can do this in one of the following two ways (in these examples, the dead-letter queue is called CSQ1.DEAD.QUEUE and the queue manager is called CSQ1):

1. The names can be specified as positional parameters in the PARM parameter of the EXEC statement within the submitted JCL, for example:

```
//READQ EXEC PGM=CSQUDLQH,
//  PARM='CSQ1.DEAD.QUEUE CSQ1'
```

Figure 78. Specifying the queue manager and dead-letter queue names for the dead-letter queue handler in the JCL

> 2. The names can be specified in the rules table, for example:

```
INPUTQ(CSQ1.DEAD.QUEUE) INPUTQM(CSQ1)
```

Figure 79. Specifying the queue manager and dead-letter queue names for the dead-letter queue handler in the rules table

Any parameters that you specify in the PARM parameter override those in the rules table. If you specify only one parameter in the PARM statement, this is used as the name of the dead-letter queue. The rules table is taken from the SYSIN data set.

Data definition statements

CSQUDLQH requires DD statements with these DDnames:

SYSOUT
> This statement is required; it names the data set for print output. You can specify the logical record length (LRECL) and block size (BLKSIZE) for this output data set.

SYSIN
> This statement is required; it names the input data set containing the rules table that specifies what the utility is to do. The logical record length (LRECL) is 80.

Sample JCL

```
//READQ EXEC PGM=CSQUDLQH,
//      PARM='CSQ1.DEAD.QUEUE CSQ1'
//STEPLIB  DD DSN=thlqual.SCSQAUTH,DISP=SHR
//         DD DSN=thlqual.SCSQLOAD,DISP=SHR
//         DD DSN=thlqual.SCSQANLE,DISP=SHR
//SYSOUT DD SYSOUT=*
//SYSIN    DD *
INPUTQM(CSQ2) INPUTQ('CSQ2.DEAD.QUEUE')
ACTION(RETRY)
/*
```

Figure 80. Sample JCL to invoke the CSQUDLQH utility. In this example, queue manager CSQ1 and dead-letter queue CSQ1.DEAD.QUEUE are used because the values specified in the PARM statement override the values specified in the SYSIN data set.

The DLQ handler rules table

The DLQ handler rules table defines how the DLQ handler is to process messages that arrive on the DLQ. There are two types of entry in a rules table:

- The first entry in the table, which is optional, contains *control data*.
- All other entries in the table are *rules* for the DLQ handler to follow. Each rule consists of a *pattern* (a set of message characteristics) that a message is matched

against, and an *action* to be taken when a message on the DLQ matches the
specified pattern. There must be at least one rule in a rules table.

Each entry in the rules table comprises one or more keywords.

See "Rules table conventions" on page 220 for information about the syntax of the
rules table.

Control data

This section describes the keywords that you can include in a control-data entry in
a DLQ handler rules table.
- All keywords are optional.
- If a control-data entry is included in the rules table, it *must* be the first entry in
  the table.
- The default value for a keyword, if any, is underlined.
- The vertical line (|) separates alternatives. You can specify only one of these.

INPUTQ (*QueueName***|' ')**
    Specifies the name of the DLQ that you want to process:
1. If you specify a queue name in the PARM parameter of the EXEC
   statement, this overrides any INPUTQ value in the rules table.
2. If you do not specify a queue name in the PARM parameter of the EXEC
   statement, the INPUTQ value in the rules table is used.
3. If you do not specify a queue name in the PARM parameter of the EXEC
   statement or the rules table, the dead-letter queue named
   qmgr–name.DEAD.QUEUE is used if it has been defined. If this queue does
   not exist, the program fails and returns error message CSQU224E, giving
   the reason code for the error.

INPUTQM (*QueueManagerName***|' ')**
    Specifies the name of the queue manager that owns the DLQ named on the
    INPUTQ keyword.
1. If you specify a queue manager name in the PARM parameter of the EXEC
   statement, this overrides any INPUTQM value in the rules table.
2. If you do not specify a queue manager name in the PARM parameter of the
   EXEC statement, the INPUTQM value in the rules table is used.
3. If you do not specify a queue manager name in the PARM parameter of the
   EXEC statement or the rules table, the default queue manager is used (if
   one has been defined using CSQBDEFV). If not, the program fails and
   returns error message CSQU220E, giving the reason code for the error.

RETRYINT (*Interval***|60)**
    Specifies the interval, in seconds, at which the DLQ handler should attempt to
    reprocess messages on the DLQ that could not be processed at the first
    attempt, and for which repeated attempts have been requested. The DLQ
    handler reprocesses messages after it has first browsed to the end of the queue.

    The default is 60 seconds.

WAIT (YES|NO|*nnn***)**
    Specifies whether the DLQ handler should wait for further messages to arrive
    on the DLQ when it detects that there are no further messages that it can
    process.

    YES   The DLQ handler waits indefinitely.

NO The DLQ handler terminates when it detects that the DLQ is either empty or contains no messages that it can process.

nnn The DLQ handler waits for *nnn* seconds for new work to arrive after it detects that the queue is either empty or contains no messages that it can process, before terminating.

Specify a value in the range 1 through 999 999.

Specify WAIT (YES) for busy DLQs, and WAIT (NO) or WAIT (*nnn*) for DLQs that have a low level of activity. If the DLQ handler is allowed to terminate, you can use triggering to invoke it when needed.

Rules (patterns and actions)

Figure 81 shows an example rule from a DLQ handler rules table.

```
PERSIST(MQPER_PERSISTENT) REASON (MQRC_PUT_INHIBITED) +
  ACTION (RETRY) RETRY (3)
```

Figure 81. An example rule from a DLQ handler rules table. This rule instructs the DLQ handler to make three attempts to deliver to its destination queue any persistent message that was put on the DLQ because MQPUT and MQPUT1 were inhibited.

This section describes the keywords that you can include in a rules table. It begins with a description of the pattern-matching keywords (those against which messages on the DLQ are matched). It then describes the action keywords (those that determine how the DLQ handler is to process a matching message).

- All keywords except ACTION are optional.
- The default value for a keyword, if any, is underlined. For most keywords, the default value is asterisk (*), which matches any value.
- The vertical line (|) separates alternatives. You can specify only one of these.

The pattern-matching keywords:

The pattern-matching keywords, are described below. You use these to specify values against which messages on the DLQ are matched. All pattern-matching keywords are optional.

APPLIDAT (*ApplIdentityData***|*)**
The *ApplIdentityData* value of the message on the DLQ, specified in the message descriptor, MQMD.

APPLNAME (*PutApplName***|*)**
The name of the application that issued the **MQPUT** or **MQPUT1** call, as specified in the *PutApplName* field of the message descriptor, MQMD, of the message on the DLQ.

APPLTYPE (*PutApplType***|*)**
The *PutApplType* value specified in the message descriptor, MQMD, of the message on the DLQ.

DESTQ (*QueueName***|*)**
The name of the message queue for which the message is destined.

DESTQM (*QueueManagerName***|*)**
The queue manager name for the message queue for which the message is destined.

FEEDBACK (*Feedback*|*)

Describes the nature of the report when the *MsgType* value is MQMT_REPORT.

You can use symbolic names. For example, you can use the symbolic name MQFB_COA to identify those messages on the DLQ that require confirmation of their arrival on their destination queues. A few symbolic names are not accepted by the utility and lead to a syntax error. In these cases, you can use the corresponding numeric value.

FORMAT (*Format*|*)

The name that the sender of the message uses to describe the format of the message data.

MSGTYPE (*MsgType*|*)

The message type of the message on the DLQ.

You can use symbolic names. For example, you can use the symbolic name MQMT_REQUEST to identify those messages on the DLQ that require replies.

PERSIST (*Persistence*|*)

The persistence value of the message. (The persistence of a message determines whether it survives restarts of the queue manager.)

You can use symbolic names. For example, you can use the symbolic name MQPER_PERSISTENT to identify those messages on the DLQ that are persistent.

REASON (*ReasonCode*|*)

The reason code that describes why the message was put to the DLQ.

You can use symbolic names. For example, you can use the symbolic name MQRC_Q_FULL to identify those messages placed on the DLQ because their destination queues were full. A few symbolic names are not accepted by the utility and lead to a syntax error. In these cases, you can use the corresponding numeric value.

REPLYQ (*QueueName*|*)

The reply-to queue name specified in the message descriptor, MQMD, of the message on the DLQ.

REPLYQM (*QueueManagerName*|*)

The queue manager name of the reply-to queue specified in the REPLYQ keyword.

USERID (*UserIdentifier*|*)

The user ID of the user who originated the message on the DLQ, as specified in the message descriptor, MQMD.

The action keywords:

The action keywords are described below. You use these to describe how a matching message is processed.

ACTION (DISCARD|IGNORE|RETRY|FWD)

The action taken for any message on the DLQ that matches the pattern defined in this rule.

DISCARD

Causes the message to be deleted from the DLQ.

IGNORE

Causes the message to be left on the DLQ.

RETRY

Causes the DLQ handler to try again to put the message on its destination queue.

FWD Causes the DLQ handler to forward the message to the queue named on the FWDQ keyword.

You must specify the ACTION keyword. The number of attempts made to implement an action is governed by the RETRY keyword. The RETRYINT keyword of the control data controls the interval between attempts.

FWDQ (_QueueName_**|&DESTQ|&REPLYQ)**

The name of the message queue to which the message is forwarded when you select the ACTION keyword.

QueueName

This parameter is the name of a message queue. FWDQ(' ') is not valid.

&DESTQ

Takes the queue name from the _DestQName_ field in the MQDLH structure.

&REPLYQ

Takes the name from the _ReplyToQ_ field in the message descriptor, MQMD. You can specify REPLYQ (?*) in the message pattern to avoid error messages, when a rule specifying FWDQ (&REPLYQ), matches a message with a blank _ReplyToQ_ field.

FWDQM (_QueueManagerName_**|&DESTQM|&REPLYQM|' ')**

The queue manager of the queue to which a message is forwarded.

QueueManagerName

This parameter defines the queue manager name for the queue to which the message is forwarded when you select the ACTION (FWD) keyword.

&DESTQM

Takes the queue manager name from the _DestQMgrName_ field in the MQDLH structure.

&REPLYQM

Takes the name from the _ReplyToQMgr_ field in the message descriptor, MQMD.

' ' The local queue manager.

HEADER (YES|NO)

Whether the MQDLH should remain on a message for which ACTION (FWD) is requested. By default, the MQDLH remains on the message. The HEADER keyword is not valid for actions other than FWD.

PUTAUT (DEF|CTX)

The authority with which messages should be put by the DLQ handler:

DEF Puts messages with the authority of the DLQ handler itself.

CTX Causes the messages to be put with the authority of the user ID in the message context. You must be authorized to assume the identity of other users, if you specify PUTAUT (CTX).

RETRY (*RetryCount***|1)**

The number of times that an action should be attempted (at the interval specified on the RETRYINT keyword of the control data). Specify a value in the range 1 through 999 999 999.

Note: The count of attempts made by the DLQ handler to implement any particular rule is specific to the current instance of the DLQ handler; the count does not persist across restarts. If you restart the DLQ handler, the count of attempts made to apply a rule is reset to zero.

Rules table conventions

The rules table must adhere to the following conventions regarding its syntax, structure, and contents:

- A rules table must contain at least one rule.
- Keywords can occur in any order.
- A keyword can be included once only in any rule.
- Keywords are not case sensitive.
- A keyword and its parameter value can be separated from other keywords by at least one blank or comma.
- Any number of blanks can occur at the beginning or end of a rule, and between keywords, punctuation, and values.
- Each rule must begin on a new line.
- For reasons of portability, the significant length of a line should not be greater than 72 characters.
- Use the plus sign (+) as the last nonblank character on a line to indicate that the rule continues from the first nonblank character in the next line. Use the minus sign (-) as the last nonblank character on a line to indicate that the rule continues from the start of the next line. Continuation characters can occur within keywords and parameters.

  For example:
  ```
  APPLNAME('ABC+
    D')
  ```
  results in 'ABCD'.
  ```
  APPLNAME('ABC-
    D')
  ```
  results in 'ABC D'.
- Comment lines, which begin with an asterisk (*), can occur anywhere in the rules table.
- Blank lines are ignored.

Each entry in the DLQ handler rules table comprises one or more keywords and their associated parameters. The parameters must follow these syntax rules:

- Each parameter value must include at least one significant character. The delimiting quotation marks in quoted values are not considered significant. For example, these parameters are valid:

  FORMAT('ABC')

  3 significant characters

  FORMAT(ABC)

  3 significant characters

FORMAT('A')
> 1 significant character

FORMAT(A)
> 1 significant character

FORMAT(' ')
> 1 significant character

These parameters are not valid because they contain no significant characters:

- – FORMAT('')
- – FORMAT( )
- – FORMAT()
- – FORMAT

- Wildcard characters are supported. You can use the question mark (?) instead of any single character, except a trailing blank. You can use the asterisk (*) instead of zero or more adjacent characters. The asterisk (*) and the question mark (?) are *always* interpreted as wildcard characters in parameter values.
- You cannot include wildcard characters in the parameters of these keywords: ACTION, HEADER, RETRY, FWDQ, FWDQM, and PUTAUT.
- Trailing blanks in parameter values, and in the corresponding fields in the message on the DLQ, are not significant when performing wildcard matches. However, leading and embedded blanks within strings in quotation marks are significant to wildcard matches.
- Numeric parameters cannot include the question mark (?) wildcard character. You can include the asterisk (*) instead of an entire numeric parameter, but the asterisk cannot be included as part of a numeric parameter. For example, these are valid numeric parameters:

MSGTYPE(2)
> Only reply messages are eligible

MSGTYPE(*)
> Any message type is eligible

MSGTYPE('*')
> Any message type is eligible

However, MSGTYPE('2*') is not valid, because it includes an asterisk (*) as part of a numeric parameter.

- Numeric parameters must be in the range zero through 999 999 999 unless otherwise stated. If the parameter value is in this range, it is accepted, even if it is not currently valid in the field to which the keyword relates. You can use symbolic names for numeric parameters.
- If a string value is shorter than the field in the MQDLH or MQMD to which the keyword relates, the value is padded with blanks to the length of the field. If the value, excluding asterisks, is longer than the field, an error is diagnosed. For example, these are all valid string values for an 8-character field:

'ABCDEFGH'
> 8 characters

'A*C*E*G*I'
> 5 characters excluding asterisks

'*A*C*E*G*I*K*M*O*'
> 8 characters excluding asterisks

- Strings that contain blanks, lowercase characters, or special characters other than period (.), forward slash (/), underscore (_), and percent sign (%) must be

enclosed in single quotation marks. Lowercase characters not enclosed in quotation marks are folded to uppercase. If the string includes a quotation, two single quotation marks must be used to denote both the beginning and the end of the quotation. When the length of the string is calculated, each occurrence of double quotation marks is counted as a single character.

Processing the rules table

The DLQ handler searches the rules table for a rule whose pattern matches a message on the DLQ. The search begins with the first rule in the table, and continues sequentially through the table. When a rule with a matching pattern is found, the rules table attempts the action from that rule. The DLQ handler increments the retry count for a rule by 1 whenever it attempts to apply that rule. If the first attempt fails, the attempt is repeated until the count of attempts made matches the number specified on the RETRY keyword. If all attempts fail, the DLQ handler searches for the next matching rule in the table.

This process is repeated for subsequent matching rules until an action is successful. When each matching rule has been attempted the number of times specified on its RETRY keyword, and all attempts have failed, ACTION (IGNORE) is assumed. ACTION (IGNORE) is also assumed if no matching rule is found.

Note:
1. Matching rule patterns are sought only for messages on the DLQ that begin with an MQDLH. If the dead-letter queue handler encounters one or more messages that are not prefixed by an MQDLH, it issues an information message to report this. Messages that do not contain an MQDLH are not processed by the DLQ handler and remain on the dead-letter queue until dealt with by another method.
2. All pattern keywords can default, so that a rule can consist of an action only. Note, however, that action-only rules are applied to all messages on the queue that have MQDLHs and that have not already been processed in accordance with other rules in the table.
3. The rules table is validated when the DLQ handler starts, and errors flagged at that time. You can make changes to the rules table at any time, but those changes do not come into effect until the DLQ handler is restarted.
4. The DLQ handler does not alter the content of messages, of the MQDLH, or of the message descriptor. The DLQ handler always puts messages to other queues with the message option MQPMO_PASS_ALL_CONTEXT.
5. Consecutive syntax errors in the rules table might not be recognized because the validation of the rules table is designed to eliminate the generation of repetitive errors.
6. The DLQ handler opens the DLQ with the MQOO_INPUT_AS_Q_DEF option.
7. Do not run applications that perform **MQGET** calls against the queue at the same time as the DLQ handler. This includes multiple instances of the DLQ handler. There is usually a one-to-one relationship between the dead-letter queue and the DLQ handler.

Ensuring that all DLQ messages are processed

The DLQ handler keeps a record of all messages on the DLQ that have been seen but not removed. If you use the DLQ handler as a filter to extract a small subset of the messages from the DLQ, the DLQ handler still keeps a record of those messages on the DLQ that it did not process. Also, the DLQ handler cannot

guarantee that new messages arriving on the DLQ will be seen, even if the DLQ is defined as first-in first-out (FIFO). Therefore, if the queue is not empty, the DLQ is periodically rescanned to check all messages. For these reasons, ensure that the DLQ contains as few messages as possible. If messages that cannot be discarded or forwarded to other queues (for whatever reason) are allowed to accumulate on the queue, the workload of the DLQ handler increases and the DLQ itself is in danger of filling up.

You can take specific measures to enable the DLQ handler to empty the DLQ. For example, do not use ACTION (IGNORE), which simply leaves messages on the DLQ. (Remember that ACTION (IGNORE) is assumed for messages that are not explicitly addressed by other rules in the table.) Instead, for those messages that you would otherwise ignore, use an action that moves the messages to another queue. For example:

```
ACTION (FWD) FWDQ (IGNORED.DEAD.QUEUE) HEADER (YES)
```

Similarly, the final rule in the table should be a catchall to process messages that have not been addressed by earlier rules in the table. For example, the final rule in the table could be something like this:

```
ACTION (FWD) FWDQ (REALLY.DEAD.QUEUE) HEADER (YES)
```

This forwards messages that fall through to the final rule in the table to the queue REALLY.DEAD.QUEUE, where they can be processed manually. If you do not have such a rule, messages are likely to remain on the DLQ indefinitely.

An example DLQ handler rules table

Here is an example rules table that contains a single control-data entry and several rules:

```
*******************************************************************************
*               An example rules table for the CSQUDLQH utility              *
*******************************************************************************
* Control data entry
* ------------------
* If no queue manager name is supplied as an explicit parameter to CSQUDLQH,
* use the default queue manager.
* If no queue name is supplied as an explicit parameter to CSQUDLQH, use the
* DLQ defined for the queue manager.
*
inputqm(' ')  inputq(' ')

* Rules
* -----

* The first check deals with attempted security violations.
* If a message was placed on the DLQ because the putter did not have the
* appropriate authority for the target queue, forward the message to a queue
* for manual inspection.

REASON(MQRC_NOT_AUTHORIZED) ACTION(FWD) +
FWDQ(DEADQ.MANUAL.SECURITY)

* The next set of rules with ACTION (RETRY) try to deliver the message to the
* intended destination.

* If a message is placed on the DLQ because its destination queue is full,
* attempt to forward the message to its destination queue.  Make 5 attempts at
* approximately 60-second intervals (the default value for RETRYINT).
```

```
                REASON(MQRC_Q_FULL) ACTION(RETRY) RETRY(5)

                * If a message is placed on the DLQ because there has been a problem starting the
                * application by triggering, forward the message to another queue for manual
                * inspection.

                REASON(MQFB_APPL_CANNOT_BE_STARTED) ACTION(FWD) +
                FWDQ(DEADQ.MANUAL.TRIGGER)

                * If a message is placed on the DLQ because of a put inhibited condition, attempt
                * to forward the message to its destination queue.  Make 5 attempts at
                * approximately 60-second intervals (the default value for RETRYINT).

                REASON(MQRC_PUT_INHIBITED) ACTION(RETRY) RETRY(5)

                * The AAAA corporation often send messages with incorrect addresses. When we find
                * a request from the AAAA corporation, we return it to the DLQ (DEADQ) of the
                * reply-to queue manager (&REPLYQM). The AAAA DLQ handler attempts to
                * redirect the message.

                MSGTYPE(MQMT_REQUEST) REPLYQM(AAAA.*) +
                  ACTION(FWD) FWDQ(DEADQ) FWDQM(&REPLYQM)

                * The BBBB corporation requests that we try sending messages to queue manager
                * BBB2 if queue manager BBB1 is unavailable.

                DESTQM(BBB1) +
                  ACTION(FWD) FWDQ(&DESTQ) FWDQM(BBB2) HEADER(NO)

                * The CCCC corporation is very security conscious, and believes that none of its
                * messages will ever end up on one of our DLQs.  If we do see a message from a
                * CCCC queue manager on our DLQ, we send it to a special destination in the CCCC
                * organization where the problem is investigated.

                REPLYQM(CCCC.*) +
                  ACTION(FWD) FWDQ(ALARM) FWDQM(CCCC.SYSTEM)

                * Messages that are not persistent risk being lost when a queue manager terminates.
                * If an application is sending nonpersistent messages, it will be able to cope with
                * the message being lost, so we can afford to discard the message.

                PERSIST(MQPER_NOT_PERSISTENT) ACTION(DISCARD)

                * For performance and efficiency reasons, we like to keep the number of messages on
                * the DLQ small. If we receive a message that has not been processed by an earlier
                * rule in the table, we assume that it requires manual intervention to resolve the
                * problem.

                * Some problems are best solved at the node where the problem was detected, and
                * others are best solved where the message originated. We do not have the message
                * origin, but we can use the REPLYQM to identify a node that has some interest
                * in this message. Attempt to put the message onto a manual intervention queue
                * at the appropriate node. If this fails, put the message on the manual
                * intervention queue at this node.

                REPLYQM('?*') +
                  ACTION(FWD) FWDQ(DEADQ.MANUAL.INTERVENTION) FWDQM(&REPLYQM)

                ACTION(FWD) FWDQ(DEADQ.MANUAL.INTERVENTION)
```

Chapter 7. User messages on startup

User messages are output when WebSphere MQ, or a channel initiator, are started successfully.

When you start WebSphere MQ successfully, it produces a set of startup messages similar to the ones for subsystem MQ86, shown here:

```
$HASP373 MQ86MSTR STARTED
IEF403I MQ86MSTR - STARTED
CSQY000I MQ86 IBM WebSphere MQ for z/OS V7
CSQY001I MQ86 QUEUE MANAGER STARTING, USING PARAMETER MODULE CSQZMQ86
CSQ3111I MQ86 CSQYSCMD - EARLY PROCESSING PROGRAM IS V7 LEVEL 004-000
CSQY100I MQ86 SYSTEM parameters ...
CSQY101I MQ86 IDBACK=1000, IDFORE=1000, LOGLOAD=500000
CSQY102I MQ86 CMDUSER=CSQOPR, QMCCSID=0, ROUTCDE=( 1)
CSQY103I MQ86 SMFACCT=NO (00000000), SMFSTAT=NO (00000000), STATIME=30
CSQY104I MQ86 OTMACON=  303
(         ,                 ,DFSYDRU0,2147483647,CSQ)
CSQY105I MQ86 TRACSTR=( 1), TRACTBL=999
CSQY106I MQ86 EXITTCB=8, EXITLIM=5, WLMTIME=60, WLMTIMU=MINS
CSQY107I MQ86 QSGDATA=(QH03,DSN910P7,DH48,4,8)
CSQY108I MQ86 RESAUDIT=YES, QINDXBLD=WAIT, CLCACHE=STATIC
CSQY110I MQ86 LOG parameters ...
CSQY111I MQ86 INBUFF=60, OUTBUFF=4000, MAXRTU=2, MAXARCH=500
CSQY112I MQ86 TWOACTV=NO, TWOARCH=NO, TWOBSDS=NO
CSQY113I MQ86 OFFLOAD=YES, WRTHRSH=256, DEALLCT=0
CSQY120I MQ86 ARCHIVE parameters ...
CSQY121I MQ86 UNIT=3390, UNIT2=, ALCUNIT=CYL,  314
PRIQTY=56, SECQTY=5, BLKSIZE=24576
CSQY122I MQ86 ARCPFX1=MQTST.SUBSYS.MQ86.ARC1,  315
ARCPFX2=MQTST.SUBSYS.MQ86.ARC2, TSTAMP=NO
CSQY123I MQ86 ARCRETN=0, ARCWTOR=YES, ARCWRTC=( 1 ,3 ,4)
CSQY124I MQ86 CATALOG=YES, COMPACT=YES, PROTECT=NO, QUIESCE=5
CSQY201I MQ86  CSQYSTRT ARM REGISTER for element  324
SYSMQMGRMQ86 type SYSMQMGR successful
IEC161I 056-084,MQ86MSTR,MQ86MSTR,BSDS1,,,MQTST.SUBSYS.MQ86.BSDS01,  326
IEC161I MQTST.SUBSYS.MQ86.BSDS01.DATA,ICFCAT.PLEX7.CATALOGA
IEC161I 056-084,MQ86MSTR,MQ86MSTR,BSDS1,,,MQTST.SUBSYS.MQ86.BSDS01,  327
IEC161I MQTST.SUBSYS.MQ86.BSDS01.INDEX,ICFCAT.PLEX7.CATALOGA
IEC161I 062-086,MQ86MSTR,MQ86MSTR,BSDS1,,,MQTST.SUBSYS.MQ86.BSDS01,  328
IEC161I MQTST.SUBSYS.MQ86.BSDS01.DATA,ICFCAT.PLEX7.CATALOGA
CSQJ127I MQ86 SYSTEM TIME STAMP FOR BSDS=2008-01-29 10:58:32.83
CSQJ001I MQ86 CURRENT COPY 1 ACTIVE LOG DATA SET IS  330
DSNAME=MQTST.SUBSYS.MQ86.LOGCOPY1.DS01, STARTRBA=000000000000
ENDRBA=00000275FFFF
CSQJ099I MQ86 LOG RECORDING TO COMMENCE WITH  331
STARTRBA=00000108B000
CSQW130I MQ86 'GLOBAL' TRACE STARTED, ASSIGNED TRACE NUMBER 01
CSQ5001I MQ86 CSQ5CONN Connected to DB2 DH48
CSQP007I MQ86 Page set 1 uses buffer pool 1
CSQP007I MQ86 Page set 2 uses buffer pool 2
CSQP007I MQ86 Page set 3 uses buffer pool 3
CSQP007I MQ86 Page set 4 uses buffer pool 0
CSQP007I MQ86 Page set 5 uses buffer pool 1
CSQP007I MQ86 Page set 6 uses buffer pool 2
CSQP007I MQ86 Page set 7 uses buffer pool 3
CSQP007I MQ86 Page set 8 uses buffer pool 0
CSQP007I MQ86 Page set 9 uses buffer pool 1
CSQP007I MQ86 Page set 99 uses buffer pool 2
CSQP007I MQ86 Page set 0 uses buffer pool 3
CSQY220I MQ86 Queue manager is using 26 MB of local  362
storage, 1413 MB are free
```

```
CSQV452I MQ86 CSQVXLDR Cluster workload exits not available
CSQR001I MQ86 RESTART INITIATED
CSQR003I MQ86 RESTART - PRIOR CHECKPOINT RBA=000000002158
CSQR004I MQ86 RESTART - UR COUNTS -  379
IN COMMIT=1, INDOUBT=0, INFLIGHT=0, IN BACKOUT=0
CSQR007I MQ86 UR STATUS  380
T  CON-ID        THREAD-XREF        S    URID             TIME
- --------- ---------------------- ------------- -------------------
C IYRMQ86  281A2790C1D7F0F20000738C C00000108120E 2008-01-29 10:58:44
CSQI049I MQ86 Page set 0 has media recovery  381
RBA=000000002158, checkpoint RBA=000000002158
CSQI049I MQ86 Page set 1 has media recovery  382
RBA=000000002158, checkpoint RBA=000000002158
CSQI049I MQ86 Page set 2 has media recovery  383
RBA=000000002158, checkpoint RBA=000000002158
CSQI049I MQ86 Page set 3 has media recovery  384
RBA=000000002158, checkpoint RBA=000000002158
CSQI049I MQ86 Page set 4 has media recovery  385
RBA=000000002158, checkpoint RBA=000000002158
CSQI049I MQ86 Page set 5 has media recovery  386
RBA=000000002158, checkpoint RBA=000000002158
CSQI049I MQ86 Page set 6 has media recovery  387
RBA=000000002158, checkpoint RBA=000000002158
CSQI049I MQ86 Page set 7 has media recovery  388
RBA=000000002158, checkpoint RBA=000000002158
CSQI049I MQ86 Page set 8 has media recovery  389
RBA=000000002158, checkpoint RBA=000000002158
CSQI049I MQ86 Page set 9 has media recovery  390
RBA=000000002158, checkpoint RBA=000000002158
CSQI049I MQ86 Page set 99 has media recovery  391
RBA=000000002158, checkpoint RBA=000000002158
CSQE005I MQ86 Structure CSQ_ADMIN connected as  393
IXL014I IXLCONN REQUEST FOR STRUCTURE QH03CSQ_ADMIN  392
CSQEQH03MQ8604, version=C1DEA7C6AB88EB45 00040002
WAS SUCCESSFUL.  JOBNAME: MQ86MSTR ASID: 03A6
CSQE021I MQ86 Structure CSQ_ADMIN connection as  394
CSQEQH03MQ8604 warning, RC=00000004 reason=02010407 codes=00000000
CONNECTOR NAME: CSQEQH03MQ8604 CFNAME: P7CF05
00000000 00000000
IXL014I IXLCONN REQUEST FOR STRUCTURE QH03APPL1  396
CSQE005I MQ86 Structure APPL1 connected as  397
CSQEQH03MQ8604, version=C1DEA8DB0311932E 00020002
WAS SUCCESSFUL.  JOBNAME: MQ86MSTR ASID: 03A6
CONNECTOR NAME: CSQEQH03MQ8604 CFNAME: P7CF05
CSQE007I MQ86 EEPLDISCFAILCONNECTION event received  398
for structure APPL1 connection name CSQEQH03MQ8604
CSQE011I MQ86 Recovery phase 1 started for structure  399
APPL1 connection name CSQEQH03MQ8604
CSQE013I MQ86 Recovery phase 1 completed for  400
structure APPL1 connection name CSQEQH03MQ8604
CSQE012I MQ86 Recovery phase 2 started for structure  401
APPL1 connection name CSQEQH03MQ8604
CSQE014I MQ86 Recovery phase 2 completed for  402
structure APPL1 connection name CSQEQH03MQ8604
CSQE006I MQ86 Structure APPL1 connection name  407
CSQEQH03MQ8604 disconnected
IXL014I IXLCONN REQUEST FOR STRUCTURE QH03APPL2  427
CSQE005I MQ86 Structure APPL2 connected as  428
CSQEQH03MQ8604, version=C1DEA8E6BB67A14F 00040002
WAS SUCCESSFUL.  JOBNAME: MQ86MSTR ASID: 03A6
CONNECTOR NAME: CSQEQH03MQ8604 CFNAME: P7CF05
CSQE007I MQ86 EEPLDISCFAILCONNECTION event received  429
for structure APPL2 connection name CSQEQH03MQ8604
CSQE011I MQ86 Recovery phase 1 started for structure  430
APPL2 connection name CSQEQH03MQ8604
CSQE013I MQ86 Recovery phase 1 completed for  431
structure APPL2 connection name CSQEQH03MQ8604
```

CSQE012I MQ86 Recovery phase 2 started for structure 432
APPL2 connection name CSQEQH03MQ8604
CSQE014I MQ86 Recovery phase 2 completed for 433
structure APPL2 connection name CSQEQH03MQ8604
CSQE006I MQ86 Structure APPL2 connection name 436
CSQEQH03MQ8604 disconnected
CSQE005I MQ86 Structure APPLSYS connected as 440
IXL014I IXLCONN REQUEST FOR STRUCTURE QH03APPLSYS 439
CSQEQH03MQ8604, version=C1DEA7CC633AC140 00040003
WAS SUCCESSFUL. JOBNAME: MQ86MSTR ASID: 03A6
CONNECTOR NAME: CSQEQH03MQ8604 CFNAME: P7CF05
CSQE007I MQ86 EEPLDISCFAILCONNECTION event received 441
for structure APPLSYS connection name CSQEQH03MQ8604
CSQE011I MQ86 Recovery phase 1 started for structure 442
APPLSYS connection name CSQEQH03MQ8604
CSQE013I MQ86 Recovery phase 1 completed for 443
structure APPLSYS connection name CSQEQH03MQ8604
CSQE012I MQ86 Recovery phase 2 started for structure 444
APPLSYS connection name CSQEQH03MQ8604
CSQE014I MQ86 Recovery phase 2 completed for 445
structure APPLSYS connection name CSQEQH03MQ8604
CSQE006I MQ86 Structure APPLSYS connection name 448
CSQEQH03MQ8604 disconnected
CSQR030I MQ86 Forward recovery log range 449
from RBA=000000002158 to RBA=00000108A1A2
CSQR005I MQ86 RESTART - FORWARD RECOVERY COMPLETE - 450
IN COMMIT=0, INDOUBT=0
CSQR032I MQ86 Backward recovery log range 451
from RBA=00000108A1A2 to RBA=00000108A1A2
CSQR006I MQ86 RESTART - BACKWARD RECOVERY COMPLETE - 452
INFLIGHT=0, IN BACKOUT=0
CSQH001I MQ86 CSQHINSQ Security using uppercase classes
CSQH021I MQ86 CSQHINSQ SUBSYSTEM security switch set 456
OFF, profile 'MQ86.NO.SUBSYS.SECURITY' found
CSQR002I MQ86 RESTART COMPLETED
CSQP018I MQ86 CSQPBCKW CHECKPOINT STARTED FOR ALL BUFFER POOLS
CSQP019I MQ86 CSQP1DWP CHECKPOINT COMPLETED FOR 461
MQ86 DISPLAY CONN(*) TYPE(CONN) ALL WHERE(UOWSTATE EQ UNRESOLVED)
BUFFER POOL 3, 10 PAGES WRITTEN
CSQP019I MQ86 CSQP1DWP CHECKPOINT COMPLETED FOR 463
BUFFER POOL 1, 17 PAGES WRITTEN
CSQP019I MQ86 CSQP1DWP CHECKPOINT COMPLETED FOR 464
BUFFER POOL 2, 14 PAGES WRITTEN
CSQP019I MQ86 CSQP1DWP CHECKPOINT COMPLETED FOR 465
BUFFER POOL 0, 47 PAGES WRITTEN
CSQP021I MQ86 Page set 0 new media recovery 466
RBA=00000108D142, checkpoint RBA=00000108D142
CSQP021I MQ86 Page set 1 new media recovery 467
RBA=00000108D142, checkpoint RBA=00000108D142
CSQP021I MQ86 Page set 2 new media recovery 468
RBA=00000108D142, checkpoint RBA=00000108D142
CSQP021I MQ86 Page set 3 new media recovery 469
RBA=00000108D142, checkpoint RBA=00000108D142
CSQP021I MQ86 Page set 4 new media recovery 470
RBA=00000108D142, checkpoint RBA=00000108D142
CSQP021I MQ86 Page set 5 new media recovery 471
RBA=00000108D142, checkpoint RBA=00000108D142
CSQP021I MQ86 Page set 6 new media recovery 472
RBA=00000108D142, checkpoint RBA=00000108D142
CSQP021I MQ86 Page set 7 new media recovery 473
RBA=00000108D142, checkpoint RBA=00000108D142
CSQP021I MQ86 Page set 8 new media recovery 474
RBA=00000108D142, checkpoint RBA=00000108D142
CSQP021I MQ86 Page set 9 new media recovery 475
RBA=00000108D142, checkpoint RBA=00000108D142
CSQP021I MQ86 Page set 99 new media recovery 476
RBA=00000108D142, checkpoint RBA=00000108D142

```
CSQI007I MQ86 CSQIRBLD BUILDING IN-STORAGE INDEX FOR  477
QUEUE SYSTEM.DURABLE.SUBSCRIBER.QUEUE
CSQI006I MQ86 CSQIRBLD COMPLETED IN-STORAGE INDEX FOR  478
QUEUE SYSTEM.DURABLE.SUBSCRIBER.QUEUE
CSQN011I MQ86 COMMAND SERVER STATUS IS ENABLED
CSQM297I MQ86 CSQMDRTC NO CONN FOUND MATCHING REQUEST CRITERIA
CSQ9022I MQ86 CSQMDRTC ' DISPLAY CONN' NORMAL COMPLETION
MQ86 DISPLAY SYSTEM
MQ86 DISPLAY LOG
CSQJ322I MQ86 DISPLAY SYSTEM report ...  484
Parameter    Initial value          SET value
-----------  ---------------------- ----------------------
IDBACK      1000
IDFORE      1000
LOGLOAD     500000
CMDUSER     CSQOPR
QMCCSID     0
ROUTCDE     1
SMFACCT     NO
SMFSTAT     NO
STATIME     30
OTMACON
  GROUP
  MEMBER
  DRUEXIT    DFSYDRU0
  AGE        2147483647
  TPIPEPFX   CSQ
TRACSTR     1
TRACTBL     999
EXITTCB     8
EXITLIM     5
WLMTIME     60
MQ86 DISPLAY ARCHIVE
WLMTIMU     MINS
QSGDATA
  QSGNAME    QH03
  DSGNAME    DSN910P7
  DB2NAME    DH48
  DB2SERV    4
  DB2BLOB    8
RESAUDIT    YES
QINDXBLD    WAIT
CLCACHE     STATIC
End of SYSTEM report
CSQ9022I MQ86 CSQJC001 ' DISPLAY SYSTEM' NORMAL COMPLETION
CSQJ322I MQ86 DISPLAY LOG report ...  487
Parameter    Initial value          SET value
-----------  ---------------------- ----------------------
MQ86 DISPLAY USAGE
INBUFF      60
OUTBUFF     4000
MAXRTU      2
MAXARCH     500
TWOACTV     NO
TWOARCH     NO
TWOBSDS     NO
OFFLOAD     YES
WRTHRSH     256
DEALLCT     0
End of LOG report
CSQJ370I MQ86 LOG status report ...  489
Copy %Full DSName
 1    43  MQTST.SUBSYS.MQ86.LOGCOPY1.DS01
 2        Inactive
Restarted at 2008-01-29 11:00:23 using RBA=00000108B000
Latest RBA=0000010914B7
Offload task is AVAILABLE
```

```
Full logs to offload - 0 of 4
CSQ9022I MQ86 CSQJC001 ' DISPLAY LOG' NORMAL COMPLETION
CSQM050I MQ86 CSQMIGQA Intra-group queuing agent starting, TCB=0092AAD0
CSQJ322I MQ86 DISPLAY ARCHIVE report ...  492
Parameter   Initial value        SET value
CSQM073I MQ86 CSQMDURR Loading of durable subscribers started
----------- ---------------------- ----------------------
UNIT        3390
UNIT2
CSQM074I MQ86 CSQMDURR Loading of durable subscribers finished
ALCUNIT     CYL
PRIQTY      56
SECQTY      5
CSQM075I MQ86 CSQMDURR Consolidation of durable subscribers started
BLKSIZE     24576
ARCPFX1     MQTST.SUBSYS.MQ86.ARC1
ARCPFX2     MQTST.SUBSYS.MQ86.ARC2
TSTAMP      NO
ARCRETN     0
ARCWTOR     YES
ARCWRTC     1 ,3 ,4
CATALOG     YES
CSQM076I MQ86 CSQMDURR Consolidation of durable subscribers finished
COMPACT     YES
PROTECT     NO
QUIESCE     5
End of ARCHIVE report
CSQJ325I MQ86 ARCHIVE tape unit report ...  498
Addr St CorrelID VolSer DSName
---- -- -------- ------ -------------------------------------------
No tape archive reading activity
End of tape unit report
CSQ9022I MQ86 CSQJC001 ' DISPLAY ARCHIVE' NORMAL COMPLETION
CSQI010I MQ86 Page set usage ...  500
  Page Buffer     Total    Unused  Persistent  NonPersist Expansion
   set   pool     pages     pages  data pages  data pages         count
_    0      0      5038      5010          28           0 USER      0
_    1      1     50035     50033           2           0 USER      0
_    2      2     50035     50032           3           0 USER      0
_    3      3     50035     50035           0           0 USER      0
_    4      0     50035     50033           2           0 USER      0
CSQY022I MQ86 QUEUE MANAGER INITIALIZATION COMPLETE
_    5      1     50035     50035           0           0 USER      0
_    6      2     50035     50035           0           0 USER      0
CSQ9022I MQ86 CSQYASCP 'START QMGR' NORMAL COMPLETION
_    7      3     50035     50035           0           0 USER      0
_    8      0     50035     50035           0           0 USER      0
_    9      1     50035     50035           0           0 USER      0
_   99      0     50035     50035           0           0 USER      0
 End of page set report
CSQP001I MQ86 Buffer pool 0 has 5000 buffers
CSQP001I MQ86 Buffer pool 1 has 5000 buffers
CSQP001I MQ86 Buffer pool 2 has 5000 buffers
CSQP001I MQ86 Buffer pool 3 has 5000 buffers
CSQI024I MQ86 CSQIDUSE Restart RBA for system as  508
configured=000000000000
CSQ9022I MQ86 CSQIDUSE ' DISPLAY USAGE' NORMAL COMPLETION
IXL014I IXLCONN REQUEST FOR STRUCTURE QH03APPLSYS  517
CSQE005I MQ86 Structure APPLSYS connected as  518
CSQEQH03MQ8604, version=C1DEA7CC633AC140 00040004
WAS SUCCESSFUL.  JOBNAME: MQ86MSTR ASID: 03A6
CONNECTOR NAME: CSQEQH03MQ8604 CFNAME: P7CF05
```

1. The actual messages you get depend on the state of your system, the objects
   that you have defined, the parameters you specify, and the page sets and other
   data sets that you are using.

If you are starting the queue manager for the first time, or if the queue manager is not in a queue-sharing group, the messages are somewhat different.

2. If any of the values in message CSQR004I is not zero, message CSQR007I is issued to provide the restart status table.

3. Messages CSQP018I and CSQP019I are issued every time a checkpoint is taken. At checkpoint time, all pages that have not been changed for the two checkpoints are written out to DASD. Message CSQP019I is issued for each buffer pool, giving the number of pages written. You can use this information when balancing page sets in buffer pools.

   If you want to suppress these messages, see the WebSphere MQ for z/OS System Setup Guide.

4. There might be periods during startup when no messages are produced; for example, if you are using indexed queues, no messages are produced while the queue indexes are being rebuilt.

5. The messages provide a wide-ranging set of information about the state of the queue manager, including:

   - system parameter values from the system parameter module, and as set by commands in the initialization data sets
   - security switch settings
   - page set and buffer pool status
   - unresolved units of work
   - log status

   A number of commands are issued internally during start up (for example DISPLAY CONN and DISPLAY SYSTEM) to provide some of the information, in addition to information that is provided directly.

Messages when a channel initiator is started

When you successfully start a channel initiator, messages similar to those shown here are output:

```
CSQM138I MQ04 CSQMSCHI CHANNEL INITIATOR STARTING
CSQ9022I MQ04 CSQXCRPS ' START CHINIT' NORMAL COMPLETION
CSQM137I MQ04 CSQMDDQM  DISPLAY DQM COMMAND ACCEPTED
CSQX830I MQ04 CSQXRDQM Channel initiator active
CSQX831I MQ04 CSQXRDQM 8 adapter subtasks started, 8 requested
CSQX832I MQ04 CSQXRDQM 5 dispatchers started, 5 requested
CSQX833I MQ04 CSQXRDQM 0 SSL server subtasks started, 0 requested
CSQX840I MQ04 CSQXRDQM 0 channels current, maximum 1520
CSQX841I MQ04 CSQXRDQM 0 channels active, maximum 1520, including 0 paused
CSQX842I MQ04 CSQXRDQM 0 channels starting,0 stopped, 0 retrying
CSQX836I MQ04 CSQXRDQM Maximum channels - TCP/IP 1520, LU 6.2 1520
CSQX845I MQ04 CSQXRDQM TCP/IP system name is TCPIP
CSQX848I MQ04 CSQXRDQM TCP/IP listener INDISP=QMGR not started
CSQX849I MQ04 CSQXRDQM LU 6.2 listener INDISP=QMGR not started
CSQ9022I MQ04 CSQXCRPS ' DISPLAY DQM' NORMAL COMPLETION
CSQM134I MQ04 CSQMSLIS  START LISTENER TRPTYPE(TCP) COMMAND ACCEPTED
CSQ9022I MQ04 CSQXCRPS ' START LISTENER' NORMAL COMPLETION
CSQM134I MQ04 CSQMSLIS  START LISTENER TRPTYPE(LU62) COMMAND ACCEPTED
CSQ9022I MQ04 CSQXCRPS ' START LISTENER' NORMAL COMPLETION
```

Notices

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

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

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

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

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

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

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

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

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

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

IBM United Kingdom Laboratories,
Mail Point 151,
Hursley Park,
Winchester,
Hampshire,
England
SO21 2JN.

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

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

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

The following are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

CICS	CUA	DB2
DB2 Universal Database™	DFSMS/MVS	DFSORT
IBM	IBMLink™	IMS
IMS/ESA®	iSeries™	Lotus®
MQSeries®	MVS/ESA™	NetView
OS/390®	RACF	SecureWay®
WebSphere	Windows	z/OS

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

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

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

Index

A

abend
    application option of SSM entry 66
    CICS transaction disconnecting 58
    starting after 18
    U3042 69
abnormal termination, restarting
  after 109
access method services (AMS)
    commands 86
    deleting damaged BSDS 145
    new active log definition 83
    renaming damaged BSDS 145
    REPRO 86, 96
ACTION keyword, DLQ handler 218
action queue manager 9
active data sets, printing
  (CSQ1LOGP) 201
active log
    adding 138
    data set
        copying 83
        copying with AMS REPRO
          statement 96
        log print utility (CSQ1LOGP) 201
    date 82
    defining in BSDS 83
    delays in off-loading 137, 138
    deleting from BSDS 83
    enlarging 84
    out of space 138
    preformatting (CSQJUFMT) 213
    problem
        both copies are damaged 135
        delays in off-loading 137, 138
        dual logging lost 133
        log stopped 133
        one copy is damaged 134
        out of space 138
        read I/O errors 136
        short of space 137
        write I/O errors 135
    recording existing in BSDS 84
    recovery plan, problems 133
    short of space 137
    status 82
    stopped data set effect 134, 137
    time 82
active threads 124
active units of work 124
adding new active log 138
address space
    abend 16
    canceling for WebSphere MQ 19
administering by writing programs 20
administration programs 20, 21
ALTER BUFFPOOL 99
alternative site recovery 111, 115
AMS (access method services)
    commands 86
    deleting damaged BSDS 145

AMS (access method services) *(continued)*
    new active log definition 83
    renaming damaged BSDS 145
    REPRO 86, 96
AMS REPRO, backing up and recovering
  page sets 96
API-crossing exit, enable or disable 47
application program
    CKQC DISPLAY 56
    command format 40
    issuing commands from 20
APPLIDAT keyword, DLQ handler 217
APPLNAME keyword, DLQ
  handler 217
APPLTYPE keyword, DLQ handler 217
archive data sets, printing
  (CSQ1LOGP) 201
archive log
    adding information to BSDS
      (NEWLOG) 193
    data set
        log print utility (CSQ1LOGP) 201
        password 197
    date 82
    deleting 78
    deleting information from the
      BSDS 84, 196
    discarding records 78
    password, changing 85
    problem
        allocation problems 139
        insufficient DASD for
          off-load 140
        read I/O errors during
          restart 141
        write I/O errors during
          off-load 140
    recording in BSDS 84
    recovery plan 139
    restarting 77
    time 82
ARCHIVE LOG command 75
ARCHIVE, utility function
  (CSQJU003) 197
archiving
    ARCHIVE LOG command 75
    using SET commands 77
ARM (Automatic Restart Manager)
    activating a policy 120
    clusters 121
    couple data sets 119
    defining a policy 119
    introduction 118
    LU 6.2 121
    network considerations 121
    policy sample 119
    queue-sharing groups 121
    registering with 120
    TCP/IP 121
authentication information objects,
  disposition 8

Automatic Restart Manager (ARM)
    activating a policy 120
    clusters 121
    couple data sets 119
    defining a policy 119
    introduction 118
    LU 6.2 121
    network considerations 121
    policy sample 119
    queue-sharing groups 121
    registering with 120
    TCP/IP 121

B

backing up page sets 94, 96
backup, of Coupling Facility
  structures 102
blank fields in operations and control
  panels 4
BMP (batch message program) 66
bootstrap data set (BSDS)
    adding an active log 83, 193
    adding an archive log 84, 193
    both copies are damaged 143
    change log inventory utility
      (CSQJU003) 83, 191
    changing for active logs 83
    changing for archive logs 84
    changing log inventory utility
      (CSQJU003) 85
    deleting active log information 83
    deleting archive log information 84
    determining log inventory
      contents 81
    does not agree with log 143
    error while opening 142
    errors 142
    I/O error 145
    log print utility (CSQ1LOGP) 201
    managing 81, 85
    out of synchronization 144
    print log map utility (CSQJU004) 200
    recover log inventory 79
    recovery 85
    restoring from the archive log 86
    single recovery 85
    time stamps 81
    unequal time stamps 144
BSDS (bootstrap data set)
    adding an active log 83, 193
    adding an archive log 84, 193
    both copies are damaged 143
    change log inventory utility
      (CSQJU003) 83, 191
    changing for active logs 83
    changing for archive logs 84
    changing log inventory utility
      (CSQJU003) 85
    deleting active log information 83
    deleting archive log information 84

235

Sending your comments to IBM

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book.

Please limit your comments to the information in this book and the way in which the information is presented.

To make comments about the functions of IBM products or systems, talk to your IBM representative or to your IBM authorized remarketer.

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

You can send your comments to IBM in any of the following ways:
- By mail, to this address:

  User Technologies Department (MP095)
  IBM United Kingdom Laboratories
  Hursley Park
  WINCHESTER,
  Hampshire
  SO21 2JN
  United Kingdom
- By fax:
  – From outside the U.K., after your international access code use 44-1962-816151
  – From within the U.K., use 01962-816151
- Electronically, use the appropriate network ID:
  – IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  – IBMLink: HURSLEY(IDRCF)
  – Internet: idrcf@hursley.ibm.com

Whichever method you use, ensure that you include:
- The publication title and order number
- The topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.

IBM

Spine information:

IBM

WebSphere MQ for z/OS

System Administration Guide

Version 7.0