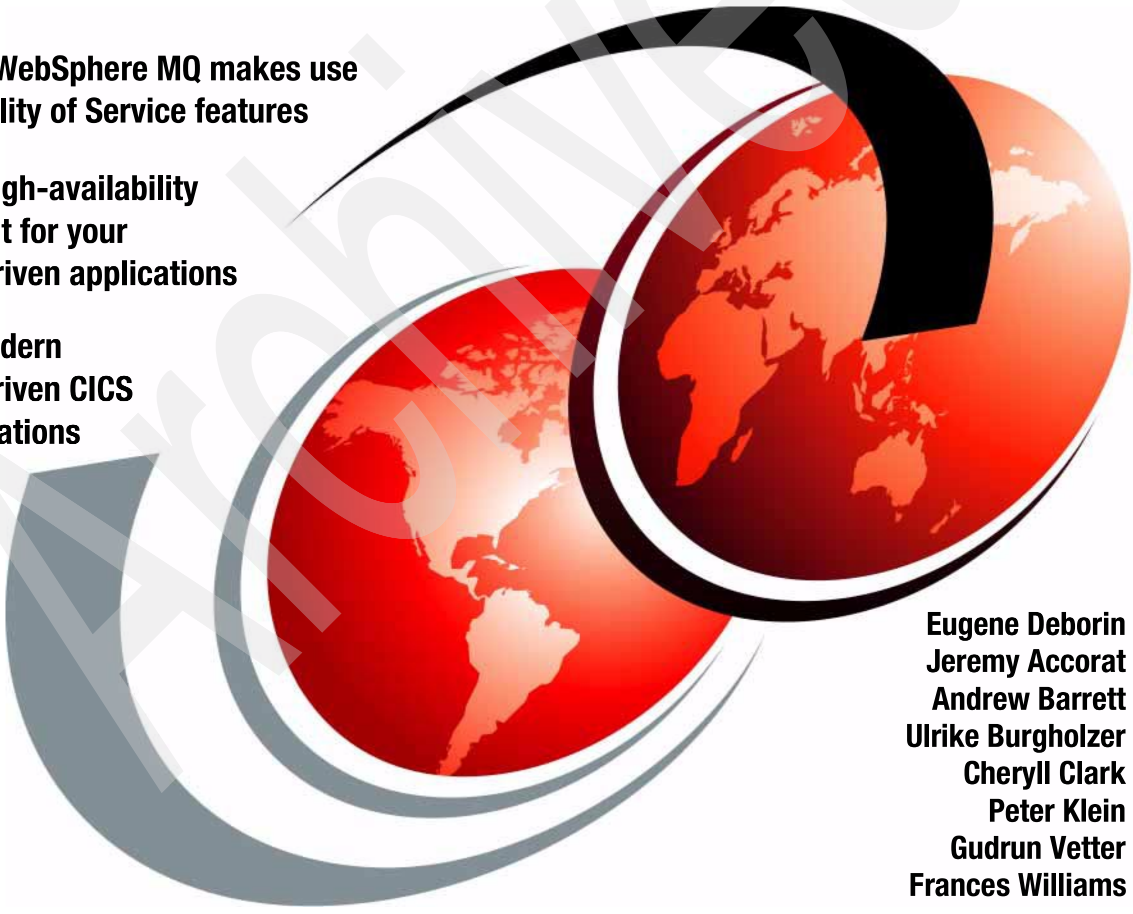
WebSphere. software

IBM

WebSphere MQ in a z/OS Parallel Sysplex Environment

- Learn how WebSphere MQ makes use of z/OS Quality of Service features
- Provide a high-availability environment for your message-driven applications
- Develop modern message-driven CICS Java applications



Eugene Deborin
Jeremy Accorat
Andrew Barrett
Ulrike Burgholzer
Cheryll Clark
Peter Klein
Gudrun Vetter
Frances Williams

Redbooks

ibm.com/redbooks

IBM

International Technical Support Organization

WebSphere MQ in a z/OS Parallel Sysplex Environment

October 2002

SG24-6864-00

Note: Before using this information and the product it supports, read the information in “Notices” on page xi.

First Edition (October 2002)

This edition applies to Version 5, Release 3 of WebSphere MQ for z/OS (product number 5655-F10).

Contents

Notices

This information was developed for products and services offered in the U.S.A.

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

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

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

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.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates 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.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

Redbooks(logo)™
AIX®
C/MVS™
CICS®
CICSPlex®
Database 2™
DB2®
DB2 Universal Database™
DFS™
DXT™
ESCON®
Everyplace™
FICON™
GDPS™
IBM®
IMS™
IMS/ESA®
Language Environment®
MQSeries®
MVS™
MVS/ESA™
OS/2®
OS/390®
PAL®
Parallel Sysplex®
PR/SM™
Processor Resource/Systems Manager™
PR/SM™
RACF®
Redbooks™
RETAIN®
RMF™
S/390®
SecureWay®
Sequent®
SP™
SP1®
SupportPac™
Sysplex Timer®
System/390®
TME®
VisualAge®
VTAM®
WebSphere®
z/OS™
zSeries™

The following terms are trademarks of International Business Machines Corporation and Lotus Development Corporation in the United States, other countries, or both:

Lotus®
Domino™

The following terms are trademarks of other companies:

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation 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.

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

C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.

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

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

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

Preface

This redbook looks at the latest enhancements to WebSphere MQ for z/OS and shows how you can make use of the z/OS Parallel Sysplex to improve throughput and availability of your message-driven applications. It helps you configure and customize your system to use shared queues in a high-availability environment and to migrate from earlier releases.

In the first part of the redbook, we provide an overview of z/OS Parallel Sysplex Technologies that you can integrate into your WebSphere MQ solution. We describe in more detail:

- z/OS Resource Recovery Services (RRS)
- Automatic Restart Manager (ARM)
- Sysplex Distributor

In the second part of the redbook, we introduce advanced features of WebSphere MQ for z/OS V5.3, namely the queue-sharing and clustering technologies.

In the third part of the redbook, we describe our test implementation scenarios and cover the following topics:

- Implementation of a queue-sharing environment
- Migration from MQSeries V5.2 with or without queue sharing
- High availability configurations
- CICS, WebSphere MQ Integrator Broker, and IMS applications in a queue-sharing environment

Finally, we look at the operational impact of a queue-sharing environment and discuss some failure and recovery scenarios.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Raleigh Center.

Eugene Deborin is a Consulting Information Technology Specialist at the International Technical Support Organization, Raleigh Center. He writes extensively and teaches IBM classes worldwide on all areas of transaction and message processing. Eugene has been working for IBM Israel since 1984, first as a Systems Engineer and, more recently, as a Software Technical Sales Specialist. He joined the ITSO San Jose Center in 1997 as a CICS Transaction Server Specialist and is now a member of the WebSphere MQ/ Business Integration ITSO team located in Raleigh and Hursley, UK.

Jeremy Accorat is a member of the Worldwide WebSphere MQ Support Team working for IBM Global Services Australia. His experience includes implementation of WebSphere MQ solutions across distributed, midrange, and 390 platforms.

Andrew Barrett is a WebSphere MQ Software Specialist in Australia. He has 20 years of experience in IBM software products . His current role is an e-business consultant in the strategic outsourcing division in IBM Global Services, advising customers about how to best utilize the WebSphere suite of products. His other areas of expertise include transaction processing systems (IMS). He has also contributed to the redbook *IMS/ESA Version 6 Guide,* SG24-2228-01.

Ulrike Burgholzer is an Advisory IT Specialist at IBM Global Services in Austria. She holds a degree in mechanical engineering (process engineering) from the Technical University in Vienna. She has over 10 years of experience in IBM mainframe software products. Her current role is to provide service for customers in installing and implementing z/OS operating systems and subsystems running on z/OS platform, with a special focus on the WebSphere MQ family.

Cheryll Clark is an Information Technology Service Specialist in IBM Australia. She has 12 years of experience in CICS and WebSphere MQ support. She has written and taught extensively on CICS internals.

Peter Klein is a CICS team leader at IBM Germany's Customer Support Center. He has 18 years of experience with IBM software products working as a technical support specialist. His areas of expertise include WebSphere MQ, CICSPlex System Manager, and distributed transaction systems. Peter has contributed to several other redbooks and ITSO projects sponsored by IBM Learning Services.

Gudrun Vetter is a Software Technical Support Specialist in IBM Germany. She supports IBM customers in solving defect and nondefect problems. She has 16 years of experience in IBM software products. Her areas of expertise include WebSphere MQ on z/OS and OS/390 and on distributed platforms.

Frances Williams is a Senior Consultant with the eServer Services in the United States. She has over 18 years of experience in the information technology field as an application design architect. Her focus is on S/390 platform technologies which incude WebSphere Application Server, WebSphere MQ, CICS, and many development languages including Java.

Thanks to the following people for their contributions to this project:

Alfred Christensen
WebSphere and eServer Networking Solutions, IBM USA

Richard Conway
International Technical Support Organization, Poughkeepsie Center

Robert Haimowitz
International Technical Support Organization, Raleigh Center

Eric Hopfgartner
IBM Austria

Yuji Nagasaki
IBM Japan

Geert Van de Putte
International Technical Support Organization, Raleigh Center

Ruediger Staub
IBM Global Services, Germany

Also, we wish to give special thanks to the review team composed of **Ed Ahern**, **Keith Andrews**, **Gren Bailey**, **Geoff Belding**, **Amardeep Bhattal**, **Alison Brinklow**, **Bob Buxton**, **Dave Fisher**, **Hazel Fix**, **Richard Harran**, **Malcolm Heron**, **Steve Hobson**, **Morag Hughson**, **Neil Johnston**, **Mark Piggott**, **Mayur Raja**, **Pete Sidall,** and **Christina Southwick** of IBM UK, Hursley Laboratories.

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

`ibm.com/redbooks/residencies.html`

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- Use the online **Contact us** review redbook form found at:

  `ibm.com/redbooks`

- Send your comments in an Internet note to:

  `redbook@us.ibm.com`

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HZ8 Building 662
  P.O. Box 12195
  Research Triangle Park, NC 27709-2195

Part 1

WebSphere MQ for z/OS environment

In Part 1 of this redbook, we provide an overview of z/OS Parallel Sysplex Technologies that you can integrate into your WebSphere MQ solution. We describe in more detail:

- Base characteristics of a Parallel Sysplex environment
- z/OS Resource Recovery Services (RRS)
- Automatic Restart Manager (ARM)
- Sysplex Distributor

1

Business integration using WebSphere MQ

In this chapter, we look at a typical business scenario and explore the solution that includes the business integration product WebSphere MQ at its core. WebSphere MQ provides the base messaging functions for servers and clients. The base product plus the new features included in Version 5.3 address the goals and meet the challenges of the business case presented. We explore some of these features in the remainder of this chapter and in greater detail throughout this book.

Features have been added or enhanced in WebSphere MQ Version 5.3, driven by the following business requirements:

- High availability
- Higher capacity and workload balancing
- Easier system administration
- Decreasing the differences in functionality between the z/OS platform and the other Version 5 platforms

To achieve the higher availability and capacity and workload balancing, WebSphere MQ has gradually implemented a number of z/OS Parallel Sysplex technologies.

Throughout the book, you find references to OS/390 and z/OS. For the purposes of this book, these names for the mainframe operating system are interchangeable. WebSphere MQ runs as well on OS/390 as on z/OS.

1.1 Meeting the business challenge

Today's businesses are under pressure as never before to deliver applications with high availability and scalability with workload balancing, while decreasing the time required for system administration. Consider the computing challenges to meet the business demands in the following business case. This case is an actual composite of a real-life business problem. We shall give the company a fictitious name, Purity.

Purity accepts credit applications from a Web site, scores the application and then matches the applicant to an appropriate funding institution. It is essential that the system is available 24x7 and that all messages transfer between the key players in timely fashion. Purity hopes to grow to become a major national source of credit application processing.

Some of the services included are viewing the loan information, viewing the funder's information, selecting a funder, analyzing the loan, requesting additional information, creating invoices for seller/lender transactions, verifying funds, and sending confirmation for fund transfers.

The system requires a browser-based credit application entry. The buyer must be identified and authorized through a trusted source. After the application enters the system, the scoring of the credit application involves a third-party application service provider (ASP) and all data about the application must be shared. In that the shared information contains sensitive data about the applicant, Purity wants to ensure the confidentiality of data being exchanged.

To help meet the challenge of the daily handling of thousands of applications and payments with promptness and accuracy, Purity wants to allow potential customers to access credit information and to make it easier for them to submit applications. Finally a key overall objective is to integrate these Web site enhancements with Purity's existing mainframe-based applications to prevent errors and delays that might be associated with application processing. The back-end applications have served the company well and have met the business requirements. They would be too costly to replace or replicate at this time. Calculation of fees for the seller and lender are handled by the back-end applications. These transactions will interface to the existing general ledger.

Purity now houses its enterprise data and operations on a z/OS Parallel Sysplex and would like to take full advantage of computing in this environment. They also want to integrate other platforms and require use of Microsoft components in their application solution.

Goals

- Allow for single data format
- Integrate internal and external data sources
- Minimize changes to existing applications
- Provide a method of iterative deployment
- Allow for the use of Microsoft components in enterprise integration

Challenges

- Unknown data sources
- Client requires Microsoft components for integration
- Both real-time and batch update requirements
- Leverage the investment in legacy infrastructure
- Web and mainframe applications must be integrated
- Integration to the ASPs. The time frame for connecting to the vendor is essential. How quickly can we get the integration done?
- Security. How can we secure the data? There are concerns about authentication, authorization, and privacy.
- Service level agreements. Can we assure high availability and scalability?

Solution

Figure 1-1 on page 6 shows the complexity of the software component solution. Combined with the service level agreements and the Purity's desire for growth, the solution will require a great deal of integration.

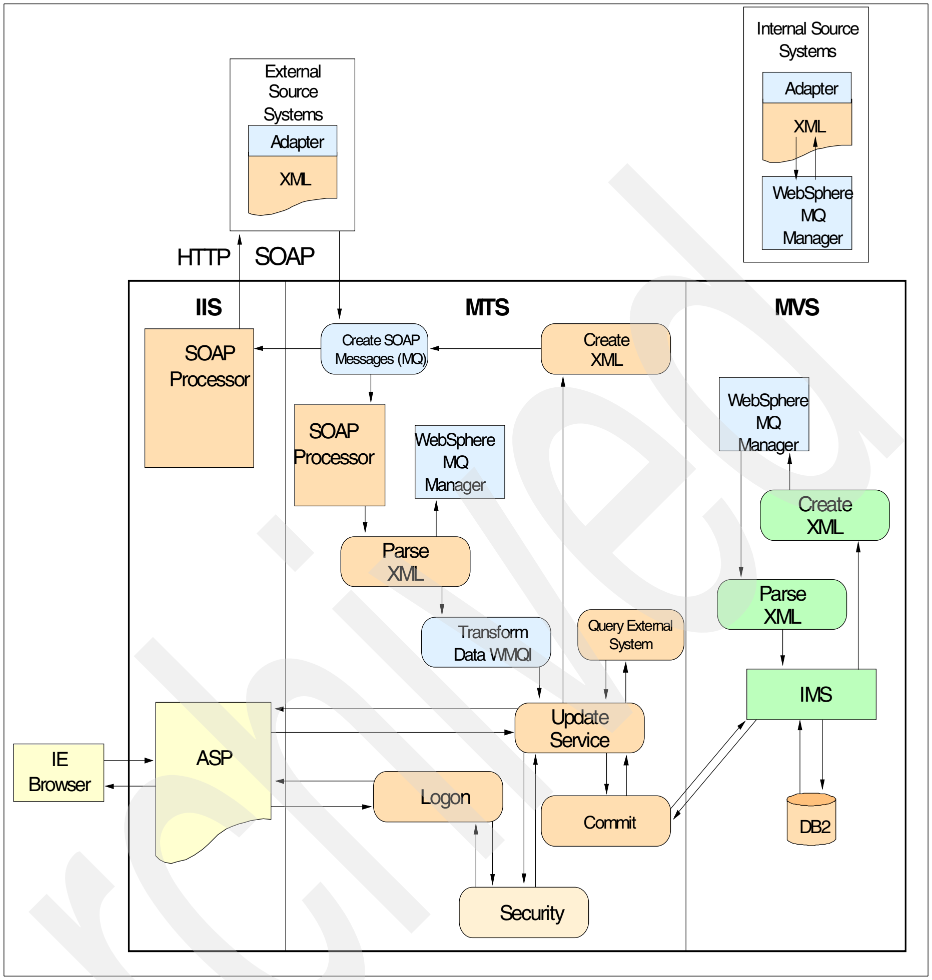


Figure 1-1 Software components

Typically, enterprise application integration is achieved using one of two architectural approaches: point-to-point integration or the hub-and-spoke integration. The choice of approach depends on the number of applications being integrated and the importance of flexibility and extensibility. With the point-to-point architecture, the number of two-way connections grows as the number of applications being integrated grow. Flexibility and extensibility are limited since each connection needs to be uniquely defined, designed, and implemented. With the hub-and-spoke architecture, each application connects to the central hub. Flexibility and extensibility are maximized since only one connection to the hub must be defined. In this solution, we recommend using WebSphere MQ to deliver messages between different computing platforms implementing the hub-and-spoke design.

The solution to meet the client's needs employs a rich array of IBM technologies including WebSphere MQ and WebSphere MQ Integrator. The integration to the enterprise applications will use WebSphere MQ. We suggested that they deploy the WebSphere MQ in a Parallel Sysplex environment to take advantage of the redundancy with increased availability.

Implementing the multi-platform messaging capabilities of WebSphere MQ, the company will be able to quickly integrate disparate applications and databases to allow applicants and lenders to connect to data across their computer network. WebSphere MQ Integrator provides middleware that transforms the data in messages to and from the Web site and provides real-time access to comprehensive repositories of information, such as personal accounts. In these ways, the IBM WebSphere MQ solution enables the company to leverage its diverse databases and business processes into a seamless network to exploit its resources and more fully connect to the end users.

Connectivity

WebSphere MQ can be fully leveraged to provide asynchronous, loosely coupled connections between the systems and their applications. Asynchronous, loosely coupled connections allow the program sending a message to continue processing without having to wait for a reply from the receiver. If the receiver or the communication channel to it temporarily goes down, the message can be forwarded at a later time. Separate request and reply queues simplify the design and use of WebSphere MQ by the applications.

WebSphere MQ is the industry-leading messaging solution for reliable data movement between over 40 different platforms. The types of connections include:

- SNA/LU6.2
- TCP/IP

Capacity requirements

Based on our discussions with Purity, we believe that our solution will meet the performance and scalability requirements that they indicated.

The solution must meet the response time and throughput requirements over a large range of volumes. Some determinants for meeting the requirements are as follows:

- WebSphere MQ runs on Windows NT and 2000, AIX, Sun Solaris, HP-UX, z/OS, and S/390.

- Clustering: MQ supports clustering, allowing increased flexibility in workload management and failover capability.
- Reliability: Using the availability and failover protection offered by the Parallel Sysplex on z/OS.

Data exchange security

The solution also encompasses the Secure Socket Layer (SSL), an out-of-the-box standard to authenticate channel partners and to encrypt messages. This is a new feature shipped with WebSphere MQ Version 5.3.

Conclusion to solution

We recommended using WebSphere MQ Integrator for message routing and data transformation. We recommended the use of SOAP-based XML for external source data and XML over WebSphere MQ for internal source data.

In the next sections of this chapter, we explain why WebSphere MQ became a significant part of the solution for this business problem. The improved scalability and availability that it lends to applications are explained in the remainder of this chapter and throughout this book.

1.2 High availability

There are two aspects we need to cover for availability. We need to guarantee that a queue manager is up and running and to make sure that communication with other queue managers remains alive. When an outage occurs, WebSphere MQ needs to do anything it can to restore functionality as quickly as possible.

In some situations, the failure is a problem that cannot be fixed in a reasonable time frame. The queue manager might be down for backup purposes, or may be the whole z/OS image is down for a planned outage, while other images with WebSphere MQ are still active. In this situation, cluster channels can be used to route messages via another channel. Cluster channels and cluster queues are introduced with MQSeries for OS/390 V2.1. We investigate the usage of it in detail in Chapter 7, “Using WebSphere MQ clustering technology” on page 115.

While the clustering technology solved many problems and raised availability, it was not a complete solution. Consider the case where a message has arrived on the target z/OS queue manager and it is waiting to be processed by a CICS transaction. At that moment, a major problem occurs with either that queue manager or the CICS system or even the whole z/OS image. The message remains stuck in the queue until the whole system is available again. New messages will be rerouted to the other images in the sysplex, but for messages that have already arrived, clustering is not the final solution.

The final solution is shared queues, where the messages are stored in a shared resource, namely the coupling facility. Shared queues were implemented with the introduction of MQSeries for OS/390 Version 5.2. Now messages can be sent to any queue manager in the sysplex and be handled by any other or the same queue manager. In Chapter 8, “Shared queue overview” on page 137 and Chapter 11, “High availability setup” on page 209, we explore the setup and usage of shared queues in detail.

With shared queues we have enhanced the availability of a queue manager. Now the availability of WebSphere MQ is at the level of the sysplex, not at the level of one specific queue manager. A failure in one queue manager will not dramatically impact current operations. To assist in the restart of that failing queue manager, WebSphere MQ has supported Automatic Restart Manager (ARM) since Version 2.1. ARM is a recovery function that you can use to configure what the system needs to do when a component is failing. You can, for example, say that a failing queue manager has to be restarted automatically on the same z/OS image. Or, if the z/OS image itself is failing, you can use ARM to restart the queue manager on another image, together with other subsystems such as DB2, CICS, or IMS.

Better availability is also achieved by reporting more information into the job log. This information will help you make the right decisions as part of the administration of your queue manager.

1.3 High capacity and workload balancing

When we look at capacity as the number of processed messages per second, then there are a number of ways to increase capacity. The most direct one is to shorten the code path. While the typical code path for the processing of a message has been decreased over time with each release of WebSphere MQ, there are limits that you cannot pass. A queue manager has to do some basic processing for logging and validation and other housekeeping functions. Another approach is to increase the number of instances where processing can occur, without impacting the single instance view from an application point of view. This has been achieved with the clustering technology and with shared queues.

A number of queue managers, on z/OS or another cluster-enabled platform, can host a cluster queue that has the same name on all queue managers. For an application that wants to send message to this queue to get some processing done, it looks like it is presented with a list of possible equivalent destinations. The WebSphere MQ code on the side of the sending application will choose one of the possible queues based on the availability of a running cluster channel. And if there is more than one running cluster channel, the selection will be done on a round-robin basis. The prerequisite is of course that all destinations have to be functionally equivalent. This means the processing of the message will be the same independent of the chosen destination. This is for example the case when the processing occurs in a Parallel Sysplex where the business data is in a shared database.

While cluster channels help to spread the workload over more than one queue manager, they have their limitations. We explore clustering technology more deeply in Chapter 7, “Using WebSphere MQ clustering technology” on page 115. One obvious limitation is the workload balancing mechanism. You can have two queue managers hosting a queue. The first queue manager runs on a heavy duty machine while the second one runs on an entry-level machine. A round-robin mechanism is not the best workload balancing algorithm for this situation. This can be fixed using a workload balancing exit that is aware of this knowledge. A more flexible solution would be a solution that is based on the actual load and capacity of each machine. On z/OS, Workload Manager (WLM) is providing this support.

With the newest release of WebSphere MQ, you now have the concept of shared channels. The WebSphere MQ listener for TCP/IP in each queue manager is listening on the same (virtual) host name and port combination. When a sender channel sends a start channel request to this virtual host name and port, it is TCP/IP that decides to which listener it will send the start channel request. TCP/IP uses the services of WLM to make its decision. If you are using the WebSphere MQ LU 6.2 listener, you can use VTAM Generic Resources feature to define one connection name to connect to a queue-sharing group.

1.4 Easier system administration

The last three releases of MQSeries for OS/390 and WebSphere MQ for z/OS have introduced a number of new object types. We now have clustered queues and channels and shared queues. If a queue is hosted by a number of queue managers, you can keep the definition of each instance of the queue synchronized because the definitions are shared by the queue manager within a group.

When an object is shared in the cluster, then the definition of that object is distributed to any queue manager that shows or has shown an interest in that object. When an application on queue manager A wants to open queue X and queue manager A does not know about this object, the queue manager will contact the repository of the cluster to which it belongs and it will check if a definition of a queue with the name X is known. If it is, the definition will be sent back to queue manager A and the queue is ready to use. This removes the need to define remote queue objects and even transmission queues. As we show in Chapter 7, “Using WebSphere MQ clustering technology” on page 115, the number of objects in a fully connected WebSphere MQ network is now linear, with the number of queue managers as the parameter. Using the traditional way of defining WebSphere MQ objects and connecting queue managers, the number of objects was quadratic. Thus, using the WebSphere MQ clustering greatly reduces the number of definitions that the administrator needs to make and maintain.

When a queue was shared in a cluster on a number of queue managers, you still had to define this queue on each queue manager. For each instance of the queue, you had one definition to make and maintain. In Version 5.2, WebSphere MQ introduces queue-sharing group objects. You define the queue only one time and you specify that the define command has to be executed in the whole queue-sharing group. A queue-sharing group is the group of queue managers on z/OS that share a common object repository, which is stored in a shared DB2 database.

When you combine both technologies together, you build one define command for a clustered queue and you tell WebSphere MQ to run it in the complete queue-sharing group. The effect of this one command is that you will now see as many instances of the object as you have member queue managers in the queue-sharing group and at the same time that the object is visible and usable in the whole cluster, which can span many hundreds or thousands of queue managers on many different platforms. It’s clear that this is a big improvement.

When having multiple instances of an object on different queue managers in the sysplex, then there is another administration aspect for which you need to maintain the same values for all instances: security. However, with MQSeries for OS/390 V5.2 and WebSphere MQ for z/OS V5.3, you can now define security profiles that are valid for all queue managers in the queue-sharing group. Instead of prefixing the profile with the name of the queue manager, you can now prefix the profile with the name of the queue-sharing group. The net result is again that you have only one definition to make that is valid throughout the queue-sharing group.

While the above improvements are important for setting up your system, there are also a number of new functions that greatly improve your ability to monitor a running system. A new command has been introduced to investigate the status of a queue. It shows what application has opened a queue and much more. The command is very powerful and contains lots of interesting information. We analyze the output of the MQSC command DISPLAY QSTATUS in Chapter 14, “Operational impact of shared queue environment” on page 265.

2

Overview of Parallel Sysplex Technologies

Before we start to investigate in detail the use of WebSphere MQ shared queues in a Parallel Sysplex environment, we first provide an overview of hardware and software technologies that are used in a Parallel Sysplex. The intention is to provide a basic level of insight for WebSphere MQ administrators who want to decide on an architecture for WebSphere MQ setup in a Parallel Sysplex environment.

This chapter concentrates on sysplex environment descriptions and definitions. To get a sense of the evolution from a single system uniprocessor through tightly coupled multiprocessors and loosely coupled configuration to the sysplex, refer to Chapter 1, “Introduction to a Sysplex”, in *z/OS Parallel Sysplex Overview: Introducing Data Sharing and Parallelism in a Sysplex,* SA22-7661.

MVS provides several system commands to manage and operate a sysplex, of which we use some in this chapter. *z/OS MVS System Commands,* SA22-7627 contains a complete description of all system commands and their parameters.

2.1 Parallel Sysplex definition

A sysplex is a collection of MVS systems that cooperate, using certain hardware and software products, to process work. All systems in a sysplex share the same sysplex name and a sysplex couple data set, which holds control information and records status information for the sysplex.

If the systems in a sysplex are all using CTC connections for communication and no coupling facility, the sysplex is called a base sysplex.

In contrast, a Parallel Sysplex is a collection of MVS systems, all of which have access to the same one or more coupling facilities. A coupling facility enables parallel processing and improved data sharing for authorized programs running in the sysplex.

If you want to enhance the capability of your enterprise to recover from disasters and other failures, you also can use a Geographically Dispersed Parallel Sysplex (GDPS). A GDPS is a multi-site application-availability solution that provides the capability to manage remote copy configuration and storage subsystem(s), automates Parallel Sysplex operational tasks, and performs failure recovery from a single point of control, thereby improving application availability. You can find information regarding GDPS in *Geographically Dispersed Parallel Sysplex: The Ultimate e-business Availability Solution* at the following Web site:

http://www.ibm.com/servers/eserver/zseries/announce/april2002/gdps.html

2.1.1 Hardware

Certain hardware is necessary to run a Parallel Sysplex. In this section we introduce some of the hardware used in a Parallel Sysplex.

Coupling facility

To run a Parallel Sysplex, you need a coupling facility. The coupling facility provides high-speed caching, list processing, and a locking function, and enables high-performance multisystem data sharing.

A coupling facility is defined as a special logical partition on either a 9674 Coupling Facility, a 9672 Parallel Sysplex Enterprise Server, a zSeries 900 server, or a zSeries 800 server. More information on coupling facility configuration options can be found in *A Positioning Paper March 2002 Coupling Facility Configuration Options,* GF22-5042.

WebSphere MQ in a queue-sharing environment requires a coupling facility at a level 9 (CFLEVEL=9) or later. With CFLEVEL 9, XES coupling facility list structure architecture extensions are introduced. The level of CFCC code that is loaded into the coupling facility determines what functionality is available. A complete and up-to-date list of coupling facility details and enhancements introduced with each CFLEVEL can be found at the following Web site:

http://www.ibm.com/servers/eserver/zseries/pso/cftable.html

Note: Higher CFLEVELs might require higher versions of your operating system. For example CFLEVEL 12 requires z/OS V1.4 to take advantage of all enhancements. Also note that when migrating to higher CFLEVELs, list, lock, and cache structure sizes might need to be increased to support new function.

To display all the coupling facilities in a sysplex, z/OS provides the D CF command. The same command used with parameter CFNAME= allows you to list only specific coupling facilities. Figure 2-1 on page 16 shows the output of the command D CF,CFNAME=CF05.

The output contains the coupling facility name, the CFLEVEL and CFCC release, and the path and device information. The space utilization shows allocated storage for structures and dump space.

```
D CF,CFNAME=CF05
IXL150I  17.14.35  DISPLAY CF 583
COUPLING FACILITY 009672.IBM.02.000000050822
                  PARTITION: F  CPCID: 00
                  CONTROL UNIT ID: FFFE
NAMED CF05
COUPLING FACILITY SPACE UTILIZATION
 ALLOCATED SPACE                 DUMP SPACE UTILIZATION
  STRUCTURES:     339968 K        STRUCTURE DUMP TABLES:        0 K
  DUMP SPACE:       2048 K                  TABLE COUNT:        0
 FREE SPACE:      691456 K       FREE DUMP SPACE:            2048 K
TOTAL SPACE:     1033472 K      TOTAL DUMP SPACE:            2048 K
                               MAX REQUESTED DUMP SPACE:        0 K
   VOLATILE:         YES         STORAGE INCREMENT SIZE:      256 K
    CFLEVEL:           9
    CFCC RELEASE 09.00, SERVICE LEVEL 01.14
    BUILT ON 05/10/2002 AT 15:32:00

COUPLING FACILITY SPACE CONFIGURATION
                            IN USE          FREE         TOTAL
  CONTROL SPACE:          342016 K      691456 K     1033472 K
  NON-CONTROL SPACE:           0 K           0 K           0 K

  SENDER PATH       PHYSICAL            LOGICAL       CHANNEL TYPE
          00         ONLINE              ONLINE        CBS
          02         ONLINE              ONLINE        CBS

  COUPLING FACILITY DEVICE     SUBCHANNEL     STATUS
                   FFFC          229D        OPERATIONAL/IN USE
                   FFFD          229E        OPERATIONAL/IN USE
                   FFFE          229F        OPERATIONAL/IN USE
                   FFFF          22A0        OPERATIONAL/IN USE
```

Figure 2-1 Sample output of command D CF,CFNAME=CF05

Coupling facility channels

There must be at least two operational signaling paths (one inbound and one outbound path) between each of the systems in the sysplex. These connection links can be:

- ESCON or FICON channels between the CPCs or
- Coupling facility channels, which provide high-speed connectivity between the coupling facility and the CPCs that use the coupling facility.

Processors

Selected models of S/390, z900, or z800 processors can take advantage of a sysplex. A single system is referred to as a Central Electronic Complex (CEC) or a Central Processor Complex (CPC).

Sysplex Timer

If systems in a sysplex run on different processors, a Sysplex Timer is required to synchronize the TOD clocks.

Additional hardware

ESCON or FICON channels enhance data access and communication in the sysplex. Directors add dynamic switching capability for both types, ESCON and FICON channels. ESCON or FICON-attached control units and I/O devices in a sysplex provide the increased connectivity necessary among a greater number of systems.

Figure 2-2 shows a hardware configuration of a Parallel Sysplex using an external coupling facility as well as an LPAR in one of the CPCs acting as a coupling facility.

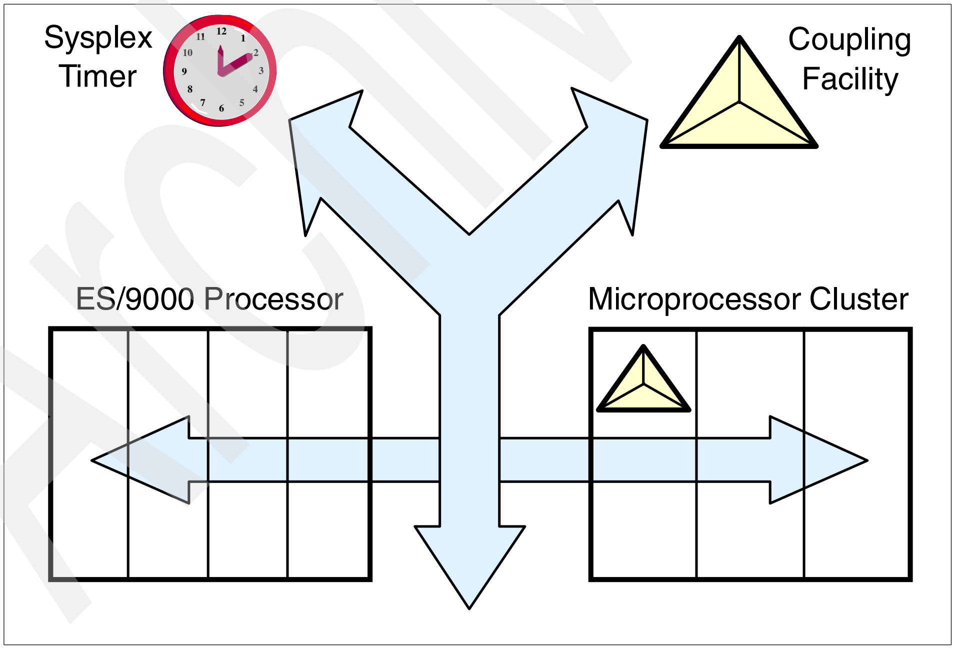


Figure 2-2 Sample configuration of hardware components used by a Parallel Sysplex

2.1.2 Software

At the base operating system level, OS/390 or z/OS operating systems provide the resource management of the coupling facility. MVS is the platform for simplified systems management of a sysplex, including configuration management, availability management, workload management, and single-image operations. The minimum operating system level required for WebSphere MQ shared-queued environment is OS/390 V2.9. You find more information on the z/OS operating system software at the following Web site:

http://www.ibm.com/servers/eserver/zseries/zos/

Communication in a Parallel Sysplex uses two important components of z/OS:

- The XCF component of MVS provides coupling services for simplified multisystem management. Applications on one member of a sysplex use the XCF services of MVS to communicate with applications on the same system or on other systems.
- The XES component of MVS enables applications and subsystems to take advantage of the coupling facility.

Other components of the base operating system and several products running on these platforms are capable of using sysplex services. Among them are:

- JES2 or JES3 controlling job queues and dispatching work in a sysplex
- DFSMS enabling placement and storage management of data for MVS
- Networking software, which includes VTAM and TCP/IP
- z/OS SecureWay Security Server RACF
- Database managers DB2 and IMS DB
- Transaction managers CICS TS and IMS TM
- WebSphere MQ for messaging and queuing

We give an overview of system software components that are used or can be used by WebSphere MQ in 2.1.10, "Workload Manager (WLM)" on page 31 through 2.1.12, "Global Resource Serialization (GRS)" on page 32 and in more detail for the two components RRS and ARM in Chapter 3, "z/OS Resource Recovery Services (RRS)" on page 43 and Chapter 4, "Automatic Restart Manager (ARM)" on page 69.

A sample configuration of a Parallel Sysplex using various software is shown in Figure 2-3.

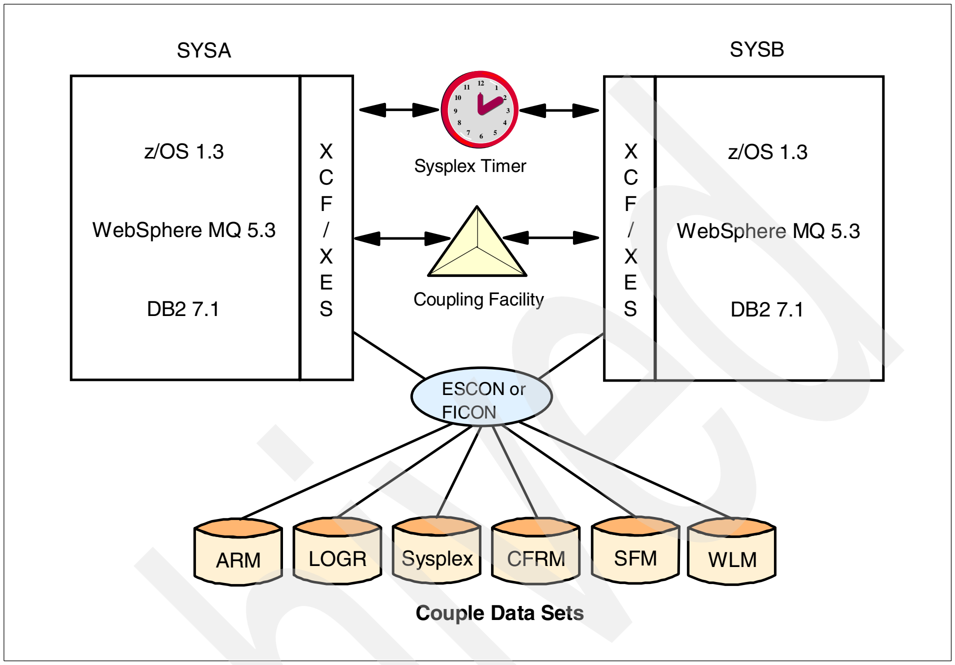


Figure 2-3 Parallel Sysplex configuration with software using sysplex functionality

2.1.3 SYS1.PARMLIB members used for sysplex setup

SYS1.PARMLIB members contain parameter values that MVS uses as input during system initialization to define the characteristics of the system. There are several SYS1.PARMLIB members containing sysplex relevant parameters.

For an MVS system to run in a sysplex you have to update the following SYS1.PARMLIB members:

- IEASYSxx contains parameter values that control the initialization of an MVS system.
- COUPLExx contains values that are used to set up a sysplex. These values include, among other definitions, the name of the sysplex and the sysplex couple data sets, the name and type of other couple data sets, and the path definitions for sysplex communication.

Certain systems running in a PR/SM partition can specify values for the XCFPOLxx parmlib member.

Depending on the way you set up your sysplex, you might also need to change values in the following SYS1.PARMLIB members:

- IEASYMxx, where you can specify system symbols
- GRSCNFxx and GRSRNLxx, which contain definitions regarding the GRS
- CLOCKxx, which holds information on how to initialize the TOD clock during system initialization
- CONSOLxx, which contains console configuration values

You find planning information for these members in Chapter 2, “Planning Parmlib Members for a Sysplex”, in *z/OS MVS Setting Up a Sysplex,* SA22-7625 and the corresponding reference information in *z/OS MVS Initialization and Tuning Reference,* SA22-7592.

2.1.4 Couple data sets

A sysplex requires at least one sysplex couple data set to store information and general status information about its members. In addition the sysplex couple data set contains information about the XCF groups and members running in a sysplex.

The sysplex couple data set does not contain any policy specification. Depending on the policies you want to define, you might need to create additional couple data sets for CFRM, SFM, ARM, LOGR and WLM policies.

To avoid a single point of failure, you should always use a primary and an alternate couple data set, which should be defined on a different device and control unit. Besides preventing a single point of failure, you can manage your couple data sets in a nondisruptive way in order to expand the size of the primary couple data set, or to move the couple data set to a different volume. The command SETXCF COUPLE allows you to make these changes dynamically.

Depending on the designated type of a couple data set, the couple data set must reside on a volume that is shared by some or all of the MVS systems in the sysplex. All sysplex couple data sets must be stored on DASDs shared by all the systems in the sysplex.

The couple data set format utility IXCL1DSU allows you to create couple data sets. The control statements for the utility program follow standard conventions for JCL statements. The utility has two levels of control statements:

- With DEFINEDS, the primary control statement, you identify the couple data set you want to format.

- With DATA TYPE, you specify the type of data to be supported in the couple data set. Data types can be sysplex data, ARM data, CFRM data, SFM data, WLM data, LOGR data or z/OS UNIX System Services data.

Defining a sysplex couple data set

Figure 2-4 shows a sample of the IXCL1DSU utility to format a sysplex couple data set.

```
//IXCSYSDF JOB
//SYSDF001 EXEC PGM=IXCL1DSU
//SYSPRINT DD  SYSOUT=A
//SYSIN    DD  *
  DEFINEDS SYSPLEX(WTSCPLX1)
           DSN(SYS1.XCF.CDS00) VOLSER(TOTDS0)
           MAXSYSTEM(16)
           CATALOG
  DATA TYPE(SYSPLEX)
           ITEM NAME(GROUP) NUMBER(200)
           ITEM NAME(MEMBER) NUMBER(203)
           ITEM NAME(GRS) NUMBER(1)
  DEFINEDS SYSPLEX(WTSCPLX1)
           DSN(SYS1.XCF.CDS04) VOLSER(TOTDS6)
           MAXSYSTEM(16)
           CATALOG
  DATA TYPE(SYSPLEX)
           ITEM NAME(GROUP) NUMBER(200)
           ITEM NAME(MEMBER) NUMBER(203)
           ITEM NAME(GRS) NUMBER(1)
/*
```

Figure 2-4 Sysplex couple data set definition using IXCL1DSU

The data type for the sysplex couple data set must be `SYSPLEX`. For a sysplex couple data set, the following parameters are valid:

- `SYSPLEX` names the sysplex for which this couple data set is formatted
- `DSN` specifies the data set name of the couple data set
- `VOLUME` specifies the volume where the couple data set should reside
- `CATALOG` enforces the data set to be cataloged at allocation time
- `MAXSYSTEM` is the maximum number of systems in the specified sysplex

For the `DATA TYPE(SYSPLEX)`, valid item names are `GROUP` and `MEMBER`. These parameters specify the maximum number of XCF groups and XCF group members for each group.

An XCF group is a set of related members that a multisystem application defines to XCF. A member is a specific function, or instance, of the application. A member resides on one system and can communicate with other members of the same group across the sysplex. Communication between group members on different systems occurs over the signaling paths that connect the systems; on the same system, communication between group members occurs through a local signaling service. To prevent multisystem applications from interfering with one another, each XCF group name in the sysplex must be unique.

In addition, the `ITEM NAME(GRS)` defines that this couple data set supports global resource serialization.

Defining couple data sets that contain policy data

The rules and actions that a sysplex is to follow are defined in policies. These allow MVS to manage specific resources in line with the requirements of a specific Parallel Sysplex setup. For a policy to be active in a sysplex, the couple data set containing this policy must be accessible to all systems that need to use it.

Except for the System Logger, the couple data sets can contain more than one administrative policy of more than one type. We recommend you use different couple data sets for different policy types.

Only one policy for each type can be active at any one time. The active couple data set contains both the administrative and the active policies. It also holds status data about the defined policies, and real-time information about the sysplex and the resources being managed by the particular policy.

Before you can define and activate a policy, you first have to format a couple data set with the utility IXCL1DSU, which is also used to format a sysplex couple data set. To get a list of valid parameters of each type of couple data set, refer to Appendix B, “Format Utility for Couple Data Sets”, in *z/OS MVS Setting Up a Sysplex,* SA22-7625.

To define administrative policies, MVS provides the administrative data utility IXCMIAPU. IXCMIAPU allows you to create or delete policies, and to obtain a report of the contents of the policy data for all administrative policies defined in the couple data set. Figure 2-5 on page 23 shows a sample JCL to define an administrative policy in the couple data set for CFRM and two coupling facilities.

```
//IXCCFRM  JOB
//CFRM0001 EXEC PGM=IXCMIAPU
//SYSPRINT DD  SYSOUT=A
//SYSABEND DD  SYSOUT=A
//SYSIN    DD  *
  DATA TYPE(CFRM) REPORT(YES)
  DEFINE POLICY NAME(CFRM087) REPLACE(YES)
  CF NAME(CF05)
     TYPE(009672)
     MFG(IBM)
     PLANT(02)
     SEQUENCE(000000050822)
     PARTITION(F)
     CPCID(00)
     SIDE(0)
     DUMPSPACE(2048)
  CF NAME(CF06)
     TYPE(002064)
     MFG(IBM)
     PLANT(02)
     SEQUENCE(00000010ECB)
     PARTITION(F)
     CPCID(00)
     SIDE(0)
     DUMPSPACE(2048)
```

Figure 2-5   CFRM policy and coupling facilities definition using IXMIAPU

Different types of policies and the sample jobs (which you can find in SYS1.SAMPLIB) to define and format the associated couple data sets are:

- CFRM policy, with sample jobs IXCCFRMF to define the couple data sets and IXCCFRMP to define the policy.
- Sysplex failure management (SFM) policy (sample jobs IXCSMFF and IXCSMFP).
- WLM policy. To define the couple data sets, you can use sample job IWMFTCDS. For the definition of policies, WLM provides an ISPF administration application.
- Automatic Restart Manager (ARM) policy (sample jobs IXCARMF and IXCARMP).
- System Logger (LOGR) policy. You can use sample job IFBLSJCL to define System Logger policies.
- UNIX System Services. You need a UNIX System Services couple data set when you implement shared HFS. To format the couple data set, you can use

the sample job BPXISCDS. Since UNIX System Services use only XCF signaling services; you do not need any policy definitions.

Most of the above technologies are needed to use the queue-sharing group of WebSphere MQ or are required if you want to exploit the new features of WebSphere MQ. In addition, WebSphere MQ needs a DB2 data-sharing group for the implementation of a queue-sharing group.

2.1.5 Signaling

Signaling is the mechanism through which XCF group members communicate in a sysplex. In a multisystem sysplex, signaling can be achieved through:

- CTC communication connections
- A coupling facility (using coupling facility list structures)
- A combination of CTC connections and list structures

There must be an outbound and an inbound path between each pair of systems in a sysplex in order to achieve full signaling connectivity.

Signaling paths defined through CTC connections must be exclusively defined as either inbound or outbound. In a sysplex configuration using CTC communication connections with four members in the sysplex, each system needs at least two devices to connect to all other systems.

Figure 2-6 shows a sysplex with four systems, which are all using CTC connections with ESCON channels for signaling paths. Figure 2-7 shows the same scenario for a setup with FICON channels.

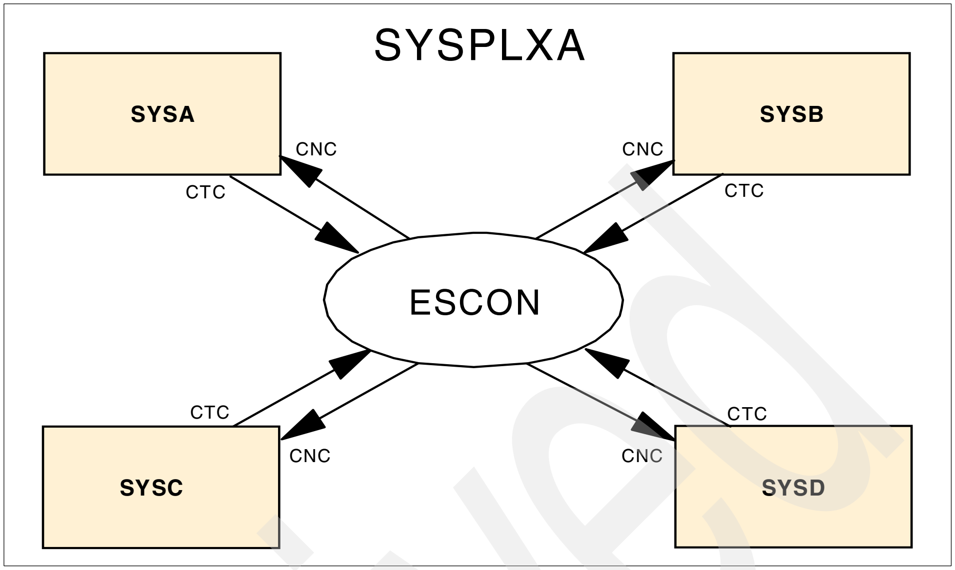


Figure 2-6 Signaling paths in a base sysplex with ESCON channel connections

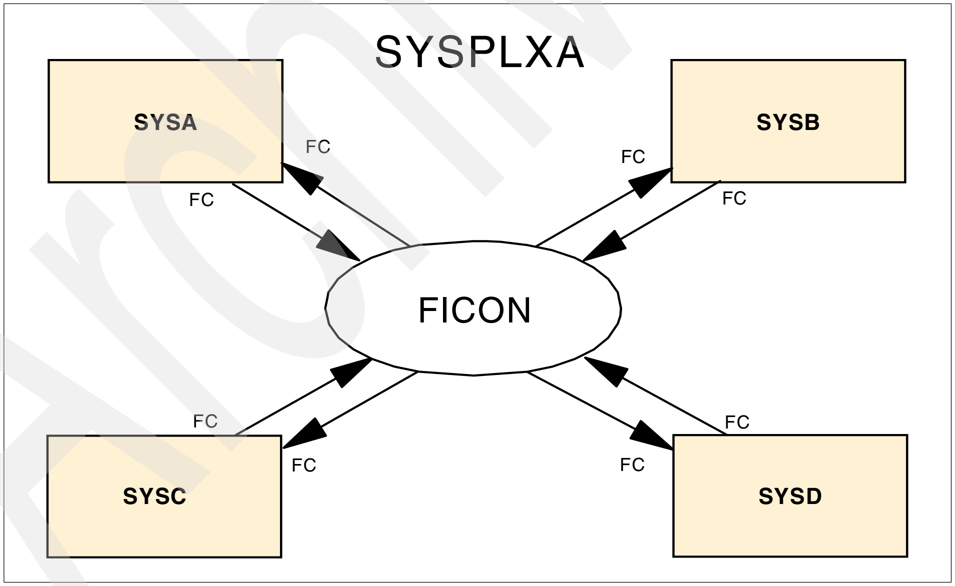


Figure 2-7 Signaling paths in a base sysplex with FICON channel connections

If you use ESCON channels, one channel has to be configured with type CNC, which means that the channel operates in native mode, and the associated channel at the other side has to be configured in channel-to-channel (CTC) mode.

If you use FICON channels, both channels are of type FC.

For both types, ESCON and FICON, the signaling paths are defined equal and have to be in the parmlib COUPLExx member. Example 2-1 shows the PATHIN and PATHOUT statements required for this configuration.

Example 2-1 Signaling path definitions for a sysplex using CTC connections

```
/* * signaling paths for connection in sysplex SYSPLXA ************* */

   PATHIN  DEVICE(4013,4023,4033,4043)
   PATHOUT DEVICE(5013,5023,5033,5043)
```

In a Parallel Sysplex, signaling may be implemented through coupling facility list structures. Figure 2-8 shows a configuration using coupling facility list structures with the minimum required signaling paths.

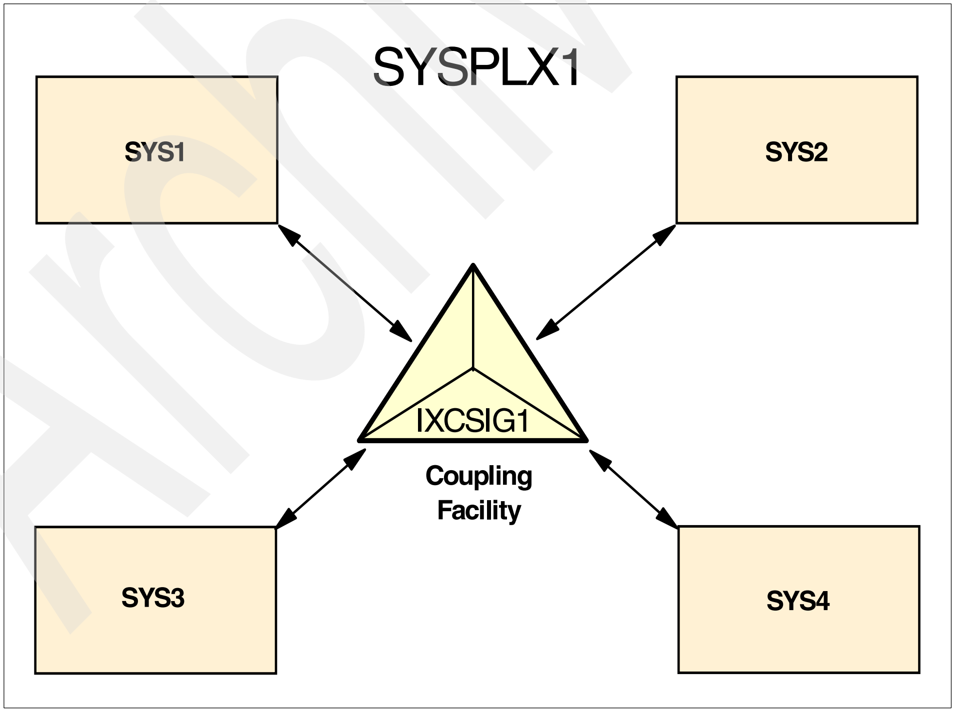


Figure 2-8 Signaling paths in a Parallel Sysplex using coupling facility list structures

With coupling facility list structures, you can use the same structure for inbound and outbound paths, and MVS automatically establishes the paths. For example, if you define a list structure for inbound traffic, MVS automatically establishes a signaling path with every other system that has the structure defined for outbound traffic. Example 2-2 shows the definitions of the necessary PATHIN and PATHOUT in COUPLExx parmlib member.

Example 2-2 Signaling path definitions using coupling facility list structure

```
/* * signaling paths to all systems in sysplex SYSPLX1 ************* */

   PATHIN  STRNAME(IXCSIG1)
   PATHOUT STRNAME(IXCSIG1)
```

2.1.6 Structures within the coupling facility

Storage within the coupling facility is divided into distinct objects called structures. The CFRM policy contains the structure definition. Structures are used by authorized programs to implement data sharing and serialization.

Figure 2-9 shows the three different structure types within a coupling facility.

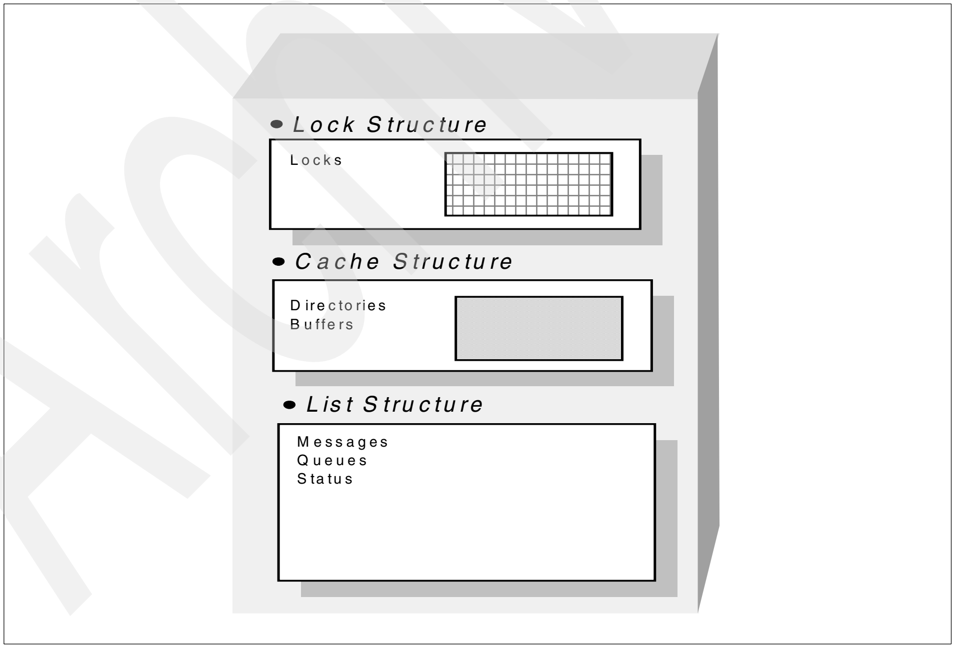


Figure 2-9 The three different structures in a coupling facility

Some storage within the coupling facility can also be allocated as dedicated dump storage for capturing structure information for diagnostic purposes. You can define only one dump space per coupling facility.

The characteristics and services associated with each structure support certain types of uses and offer certain unique functions.

Cache structure

Cache structures allow high-performance sharing of frequently referenced data. Cache structure services allow you to store and access data in the cache structure and to automatically notify affected users when you change shared data. You can also determine whether your copy of shared data is valid.

List structure

List structures enable users to share information organized as entries on a set of lists or queues. A list structure contains a set of lists. Each list contains a number of list entries. Each entry contains control information and data. List structure services allow you to read, write, move, and delete list entries. WebSphere MQ uses list structures to implement shared queues.

Figure 2-10 shows the format of a list structure. MVS processes each IXLLIST request atomically, that is, the request cannot be interrupted. This ensures that the list structure cannot be viewed or accessed while it is being updated.

Lock structure

Lock structures allow users to serialize user-defined resources, including list or cache structure data, with creating a set of locks and locking protocols. Lock structure services allow you to associate user-specified data with each lock, implement locking protocols, resolve lock contention according to your own protocol, and to recover locks as part of an overall recovery mechanism.

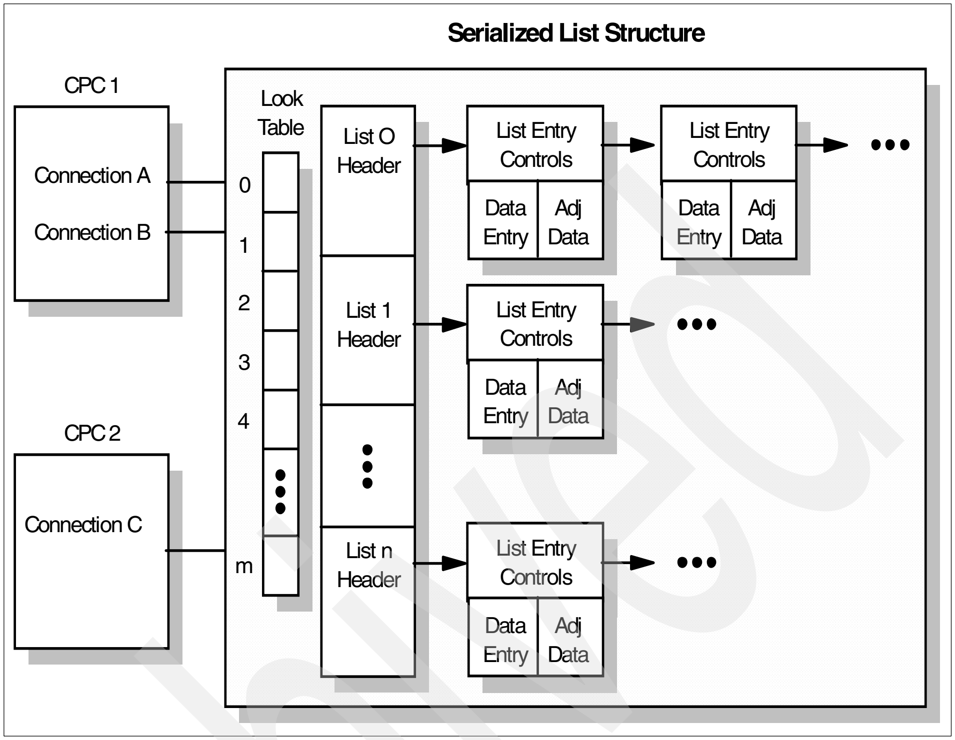


Figure 2-10 Serialized list structure

Each application specifies the structure types it requires. A coupling facility can support multiple coupling facility structures, but any one structure must reside entirely within a single coupling facility.

The size of each structure is specified in 1 KB blocks, and it is specified by the installation in its CFRM policy.

In order to size your structures appropriately, you can use the coupling facility structure sizer tool, which can be found at the Web site:

`http://www.ibm.com/servers/eserver/zseries/cfsizer`

To get a list of structures defined in your coupling facility, you can use the command

```
D XCF,STRUCTURE,STRNAME=strname
```

where *strname* represents a structure name, or `ALL`. Wildcard (*) suffixes are allowed.

2.1.7 Coupling Facility Resource Management (CFRM)

CFRM provides the services to manage coupling facility resources in a sysplex. This management includes the enforcement of CFRM policies to ensure that the coupling facility and structure requirements as defined in each policy are satisfied.

CFRM uses the CFRM couple data set to maintain status data about the use of coupling facility resources in the sysplex and to hold its copy of the active CFRM policy. Whenever a CFRM couple data set is made available in the sysplex, either for the first time or when a switch is made to a different CFRM couple data set, CFRM becomes operational. Once operational, CFRM performs the following services for the sysplex:

- Cleans up status data
- Performs system level initialization
- Reconciles active policy data with data in the coupling facility.

Before you start to set up your CFRM policy, you need to know which subsystems and applications in your sysplex use the coupling facility and what the applications structure requirements are. The CFRM policy defines the amount of coupling facility storage used for each structure; it does not identify the type of the structure.

For more information on setting up the CFRM policy, see *z/OS MVS Setting Up a Sysplex,* SA22-7625.

2.1.8 Sysplex Failure Management (SFM)

SFM allows you to define how system failures, signaling connectivity failures, and PR/SM (Processor Resource/Systems Manager) reconfiguration actions are to be handled. The role of SFM is to isolate the failed component and reconfigure the sysplex so that units of work can continue. The goal of SFM is to allow these reconfiguration decisions to be made and carried out with little or no operator involvement.

The SFM data set contains all the information SFM will need to manage system and signaling connectivity failures.This includes:

- Policy statement
- Systems statement(s)
- Reconfiguration statement(s)

See Chapter 7, “Planning Sysplex Availability and Recovery”, in *z/OS MVS Setting Up a Sysplex,* SA22-7625, for more information on how to control system availability and recovery through the SFM policy.

2.1.9 Automatic Restart Manager (ARM)

ARM is an MVS recovery function that can improve the availability of specific batch jobs or started tasks. When a job or task fails, or the system on which it is running fails, Automatic Restart Manager can restart the job or task without operator intervention. The ARM policy specifies how batch jobs and started tasks that are registered with ARM, should be restarted.

If ARM is available, WebSphere MQ will automatically register at startup. To enable ARM to restart your WebSphere MQ, you must define an ARM couple data set and restart definitions in your ARM policy.

For more information on the ARM refer to Chapter 4, “Automatic Restart Manager (ARM)” on page 69.

2.1.10 Workload Manager (WLM)

MVS WLM provides dynamic sysplex-wide management of system resources. Each installation today processes different types of work with different response time requirements. With Workload Manager, you define a performance goal to each unit of work and assign a business importance to each goal. It is the job of WLM to attain these goals through the management and distribution of resources. Figure 2-11 shows a high-level overview of the workload management philosophy.

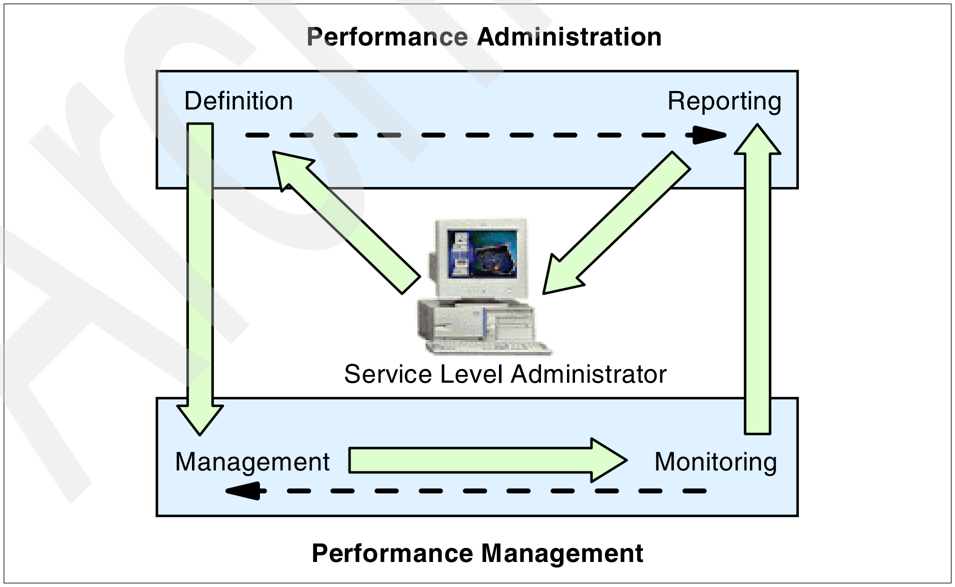


Figure 2-11 High-level overview of the workload management philosophy

For more information on WLM, see *z/OS MVS Planning: Workload Management,* SA22-7602.

2.1.11 MVS System Logger

The System Logger is a system component that allows applications to log data from a sysplex. WebSphere MQ does not use this facility to log its data. However, WebSphere MQ can participate in units of recovery managed by RRS, which uses the System Logger.

A System Logger application can write log data into a log stream, which is simply a collection of data. There are two types of log streams:

1. Coupling facility log stream, which is initially written to the coupling facility. When the structure becomes full, the data is moved to the DASD log data sets.
2. DASD-only log stream, which is first written to local storage buffers. When these buffers become full, log stream data is moved to the DASD log data sets. A DASD-only log stream can only be connected to one system at a time.

The System Logger component resides in its own address space, IXGLOGR, on each system in a sysplex. It maintains information about the current use of the log streams and coupling facility list structures in the LOGR policy. It is responsible for offloading the data from the coupling facility to DASD and automatically allocating new DASD logger data sets, when necessary.

The MVS System Logger manages the log streams based on the policy you defined in the LOGR couple data set. The LOGR policy includes log stream definitions.

For more information on the System Logger, refer to Chapter 9, “Planning for System Logger Applications”, in *z/OS MVS Setting Up a Sysplex,* SA22-7625.

2.1.12 Global Resource Serialization (GRS)

In a multi-tasking, multi-processing environment, resource serialization is the technique used to coordinate access to resources that are used by more than one program. When multiple users share data, a way to control access to that data is needed.

GRS offers the control needed to ensure the integrity of resources in a multisystem environment, and provides the serialization mechanism for shared DASD across multiple MVS images.

Depending on the MVS system operating system level and XCF environment the systems are running in, sysplex or non-sysplex, the complex consists of one or more systems:

- Connected to each other in a RING configuration through:
  - GRS managed CTC adapters
  - XCF communication paths (CTCs), or
  - Signaling paths through a coupling facility
- Connected to a coupling facility lock structure in a STAR configuration

The two types of GRS complexes are:

- A star: in this type of GRS complex, all of the systems in the sysplex must match the systems in the GRS complex. This is also sometimes called a GRS-plex.
- A ring: in this type of complex either:
  - The systems in the sysplex are the same as the systems in the complex (sysplex matches GRS complex).
  - At least one non-sysplex system in the GRS complex is outside of the sysplex (mixed complex).

GRS is required in a sysplex because components and products that use sysplex services need to access a sysplex-wide serialization mechanism. GRS must be active in each system within the Parallel Sysplex. Every system in a sysplex must be a member of the same GRS complex.

For more information on how to set up a GRS, see *MVS Planning: Global Resource Serialization,* SA22-7600.

2.1.13 Shared HFS

UNIX System Services provide mechanisms to make HFS data shareable for all sysplex users. With shared HFS, all file systems that are mounted by a system participating in shared HFS group are available to all participating systems. In other words, once a file system is mounted by a participating system, that file system is accessible by any other participating system.

It is not possible to mount a file system so that it is restricted to just one of those systems. Consider a sysplex that consists of two systems, SYSA and SYSB:

- A user logged onto any system can make changes to file systems mounted on /u, and those changes are visible to all systems.
- The system programmer who manages maintenance for the sysplex can change entries in both /etc file systems from either system.

The UNIX System Services couple data set contains the sysplex-wide mount table and information about all participating systems, and all mounted file systems in the sysplex.

Shared HFS uses a sysplex root HFS to provide a sysplex-wide root HFS. No files or code reside in the sysplex root data set. It consists of directories and symbolic links only.

Each system in a shared HFS environment still has its own common HFS data sets for /etc, /tmp, /var, and /dev. The system-specific HFS data set contains the directory entries for the /etc, /tmp, /var, and /dev mount points.

The version HFS is the IBM-supplied root HFS data set. UNIX System Services creates a directory with the value *nnnn* specified on VERSION parameter in the BPXPRMxx member and uses this directory as mount point for the specified version data set.

Figure 2-12 shows a sample setup for two systems using the same version HFS data set.

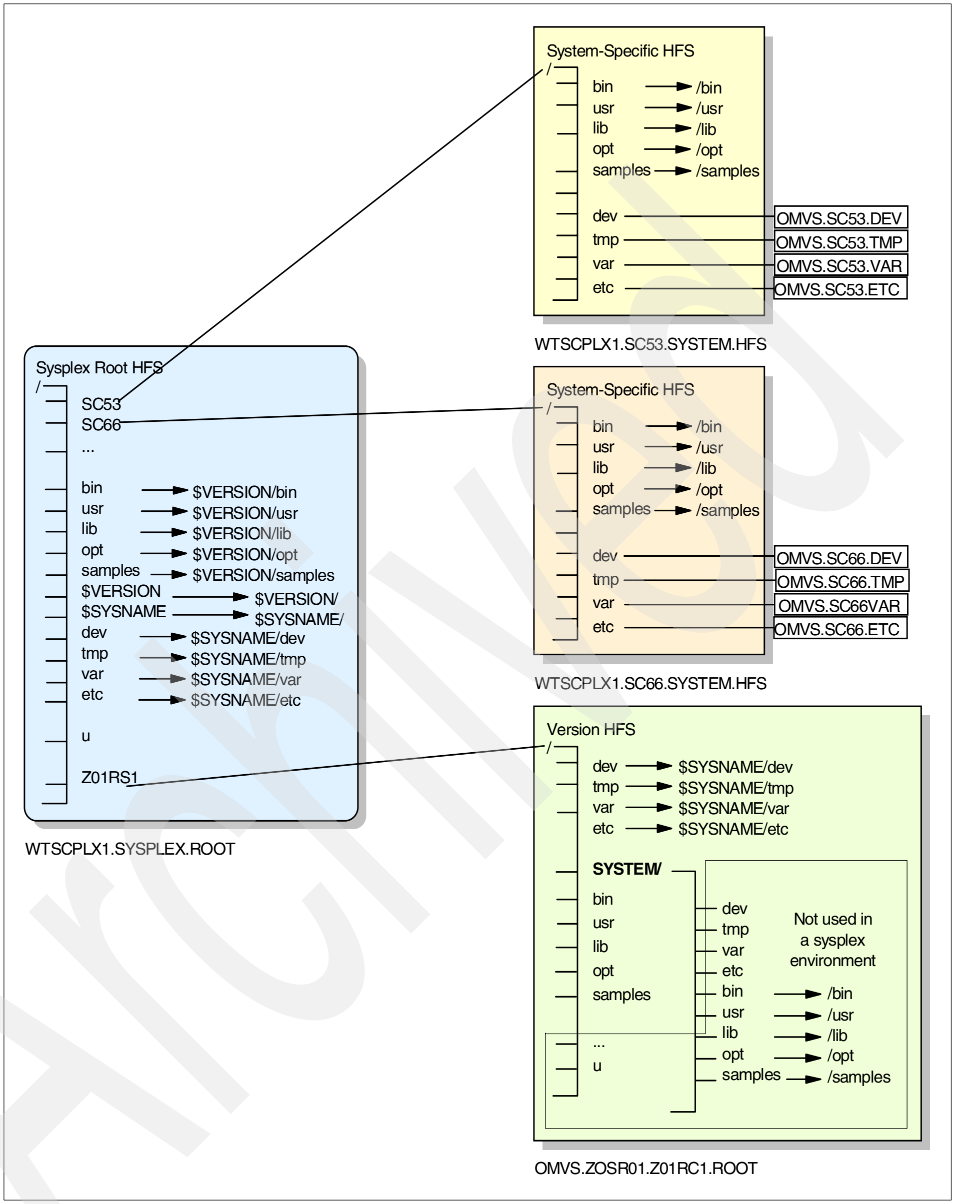


Figure 2-12   Shared HFS setup in a sysplex for two systems using the same release level of the version HFS

Figure 2-13 show the changes to BPXPRMxx member.

```
.
.

FILESYSTYPE TYPE(HFS)
            ENTRYPOINT(GFUAINIT)
            PARM(' ')

VERSION('&SYSR1.')
SYSPLEX(YES)

.
.

ROOT  FILESYSTEM('&SYSPLEX..SYSPLEX.ROOT')
      TYPE(HFS) MODE(RDWR)

MOUNT FILESYSTEM('&SYSPLEX..&SYSNAME..SYSTEM.HFS')
      TYPE(HFS) MODE(RDWR)
      NOAUTOMOVE
      MOUNTPOINT('/&SYSNAME.')

MOUNT FILESYSTEM('OMVS.ZOSR01.&SYSR1..HFS')
      TYPE(HFS) MODE(READ)
      MOUNTPOINT('/$VERSION')

MOUNT FILESYSTEM('OMVS.&SYSNAME..DEV')
      TYPE(HFS) MODE(RDWR)
      NOAUTOMOVE
      MOUNTPOINT('&SYSNAME./dev')

MOUNT FILESYSTEM('OMVS.&SYSNAME..ETC')
      TYPE(HFS) MODE(RDWR)
      NOAUTOMOVE
      MOUNTPOINT('&SYSNAME./etc')

.
.
```

Figure 2-13 Shared HFS setup in BPXPRMxx member with one version HFS for the entire sysplex

Refer to *z/OS UNIX System Services Planning,* GA22-7800 for more information on how to set up UNIX System Services and shared HFS.

2.2 Advantages of a Parallel Sysplex

You must have a Parallel Sysplex environment to use shared queues with WebSphere MQ. WebSphere MQ exploits several advantages of Parallel Sysplex providing persistent shared queues.

Continuous availability of application

A single image, hardware or software, exposes you to system outages due to either planned or unplanned outages. Within a sysplex, all systems can have concurrent access to all critical applications and data. This enables systems to take over the work of a failing system by either restarting the applications of the failing system on other systems, or by redirecting work requests to other data-sharing instances of the subsystem on other systems.

Reduction in planned outages

A Parallel Sysplex can make a planned outage less disruptive. One system can be removed and the work running will be routed to a remaining system *on the fly*. While the system is removed, new software or maintenance levels can be applied to this system. When reintroduced into the sysplex, the new software levels can coexist with any older levels on other systems within the sysplex.

Dynamic workload balancing

Dynamic workload balancing can be defined as the capacity to dynamically direct unit of work requests to run on any or all of the systems within a sysplex. The entire Parallel Sysplex can be viewed as a single logical resource to end users and business applications.

Single system image

The collection of systems in a Parallel Sysplex appear as a single entity to operators, end users, and administrators. A single system image ensures reduced complexity from both operational and definition perspectives.

Scalability

You can increase the capacity of your system and the throughput by adding additional z/OS images to your existing configuration.

2.3 DB2 data-sharing group

Since WebSphere MQ object definitions and status information is shared among the members in a WebSphere MQ queue-sharing group, WebSphere MQ needs a possibility to keep these shared data consistent. WebSphere MQ takes advantage of the DB2 data-sharing group, which actually supports data sharing. Therefore implementation of queue-sharing group also requires a DB2 data-sharing group.

If you are not familiar with DB2, refer to *DB2 UDB for OS/390 and z/OS: An Introduction to DB2 for OS/390*, SC26-9937, or go to the following Web site:

`http://www.ibm.com/software/data/db2/os390/`

A data-sharing group is a collection of one or more DB2 subsystems that access shared DB2 data. Using the data-sharing function of DB2 for OS/390 and z/OS, applications that run on more than one DB2 subsystem can now share DB2 data, that is they are able to read from and write to the same set of data concurrently.

A DB2 data-sharing group requires a Parallel Sysplex. Each subsystem that belongs to a data-sharing group is a member of that group. All members of a data-sharing group use the same shared DB2 catalog and directory. Currently, up to 32 members can participate in a data-sharing group.

2.3.1 DB2 data-sharing overview

DB2 uses special data-sharing locking and caching mechanisms to ensure data consistency. To exploit these functions, DB2 uses all three types of coupling facility structures, which are defined in CFRM policies.

Applications can access the same data from any DB2 subsystem in the data-sharing group, and therefore a request to read and write the same data can be issued by many subsystems at the same time, which is called inter-DB2 read/write interest, and also sometimes inter-DB2 interest.

DB2 uses storage areas called *group buffer pools* to cache data of inter-DB2 interest, whenever this data is changed. You define these group buffer pools as coupling facility structures in a CFRM policy.

DB2 maps these group buffer pools to buffer pools of the group members. In order to use data sharing, you must define a group buffer pool 0. Each DB2 subsystem in the data-sharing group must define a mapping BP0. GBP0 is used for caching the DB2 catalog and directory table spaces and indexes, and any other DB2 data, that use buffer pool 0. Figure 2-14 shows a sample of configuring group buffer pools, and how the group buffer pools are mapped to buffer pools in each DB2 subsystem.

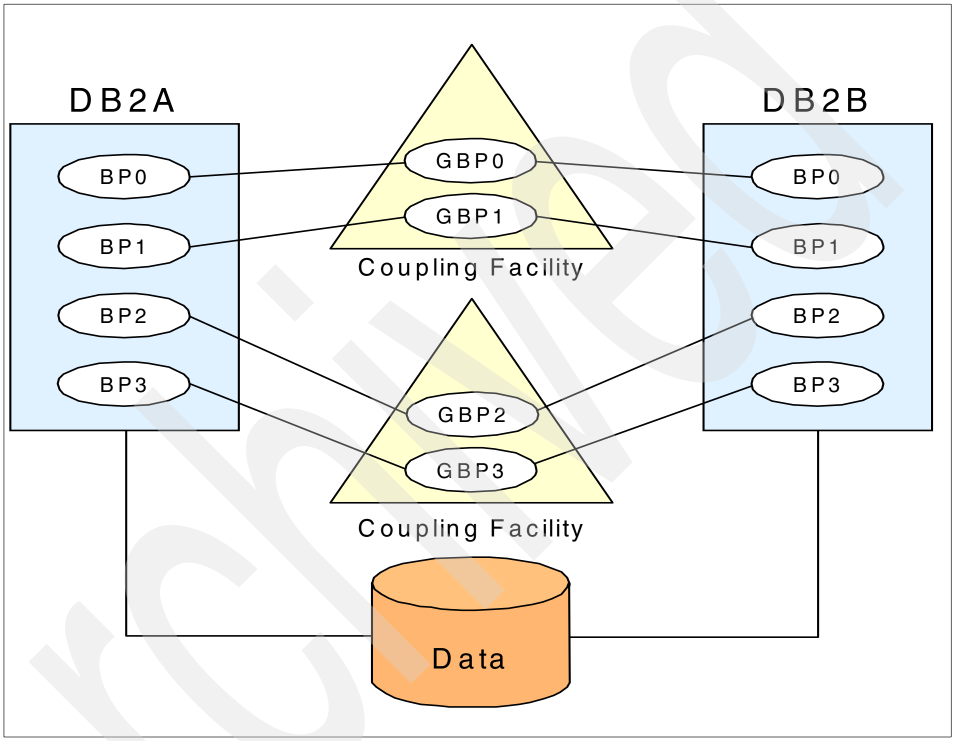


Figure 2-14 Relationship of buffer pools of DB2 subsystems and group buffer pools

You can use more than one coupling facility to put your buffer pools in, but a single buffer pool can reside only in one single coupling facility. An exception to this are duplexed buffer pools, where the secondary group buffer pool can reside in a different coupling facility from the primary group buffer pool.

2.3.2 Using a DB2 data-sharing group

If you want to change your DB2 subsystems from single non-sharing subsystems to a data-sharing group, you first must plan your naming conventions, because names in the sysplex and in the group must be unique. In a data-sharing group shared data objects as well as every group resource (such as a name for the group) must have unique names.

The DB2 catalog of the first DB2 subsystem that you change to be a member of a data-sharing group is also the catalog of all additional members that you add in future. Therefore, if you want to migrate a second DB2 subsystem to the same data-sharing group, you have to merge existing data and catalog definitions to the catalog of your data-sharing group.

You should also consider merging your connection and sign-on exit routines, because you need to share users across all possible systems and therefore you have to merge all secondary user IDs in all DB2s of the data-sharing group.

Because the DB2 catalog is shared, data definition, authorization, and control is the same as for non-data sharing. You can administer your data-sharing group from each member in the group. Any authorization and granted privilege given to an authorization ID is the same in every member of the group.

DB2 provides information on how to set up a data-sharing group in *DB2 Universal Database for OS/390 and z/OS Data Sharing: Planning and Administration,* SC26-9935.

2.4 Advantages of DB2 data sharing

DB2 data sharing provides many advantages that you can exploit in your environment:

- Improves the availability of DB2. Since all DB2 subsystems share the same catalog, directories and user data, users can access their DB2 data from another subsystem, if one subsystem goes down. Transaction managers are informed that DB2 is down and can switch new user work to another DB2 subsystem in the group.

  e-business applications with 7x24x365 availability requirements are the main users of data sharing with DB2 as an enterprise server. Refer to the redbook *DB2 UDB for OS/390 and Continuous Availability*, SG24-5486, for more information about a high-availability setup.
- With DB2 data sharing, you are now able to scale your systems very easily. Instead of splitting DB2 data in an environment without DB2 data sharing, you

can now add an additional DB2 subsystem and access the same data through the new DB2.

- Since DB2 works closely with WLM, you can ensure that incoming work is optimally balanced across the systems in your sysplex environment. For example, you can take advantage of WLM when using DDF connections in a data-sharing environment, or to exploit intra-DB2 query parallelism.

  DB2 does not use WLM for data-sharing purposes only. You can also use it within a single system. Using WLM management for your stored procedures address spaces is highly recommended.

- With DB2 data sharing, you are able to increase your transaction rates. The same application can run on more than one DB2 subsystem. You do not need to change your application to do this.
- You are now able to configure your environment to be more flexible. It is possible to run more than one data-sharing group in the same sysplex, and in addition to run single non-data sharing DB2 subsystems as well.
- You can also take advantage of data sharing, by sharing common customer tables, which were perhaps split in the past for capacity and availability reasons.

3

z/OS Resource Recovery Services (RRS)

Programs that update dependent data of more than one kind of resources, for example WebSphere MQ messages and DB2 tables, need a way to synchronize their updates so that, whether a program ends successfully or not, data updates are still consistent, and the integrity of these resources can be guaranteed. The updated resources are called *protected resources* or, sometimes, *recoverable resources*.

Resource recovery is the protection of the resources, and consists of the protocols and program interfaces that allow an application program to make consistent changes to multiple protected resources. Some resource managers (CICS Transaction Server is one) provide resource recovery mechanisms, also called *two-phase commit* processing. For z/OS batch applications, a common method to do two-phase commit is available with the introduction of RRS.

This chapter gives an overview of two-phase commit and of the terminology used for the description of resource recovery processes. It covers terms used for the description of participants in two-phase commit processing, as well as an introduction to RRS. Two sample scenarios show RRS usage with WebSphere MQ.

3.1 Concepts and terminology

Before we start to describe the functionality of RRS, we need to know some terms used later in this chapter.

3.1.1 Transaction

The term *transaction* is often used in different senses. In general, a transaction is a sequence of processing actions, for example database changes, done by a computer program, that must be completed before any of the individual actions performed by the program can be regarded as committed. A transaction has to conform to the following set of rules known as ACID features:

- Atomicity:

  Changes made by a transaction have to be treated as one single change. Either all of them have to be done, or none.

- Consistency:

  Data transformation has to be consistent. The actions done to a group of resources may not violate any of the integrity constraints associated with the state of the data.

- Isolation:

  Even though transactions execute concurrently, they appear to be serialized. So it appears to each transaction that any other transaction is done either before or after it.

- Durability:

  After a transaction completes successfully, the changes done by the transaction survive failure.

In contrast to the general definition of a transaction, the term transaction is often used to cover a set of work done by one or more application programs in response to a terminal request. CICS, for example, defines a transaction as a unit of application data processing, which may consist of one or more programs initiated by a single request.

3.1.2 Unit of work and unit of recovery (UR)

The set of changes that are to be either made or backed out as a single unit are called a *unit of work*. A unit of work represents the changes made by an application to resources since the last commit or backout operation.

The term unit of work is mostly used in application development-related documents. For example you find this term in *WebSphere MQ Application Programming Reference,* SC34-6062. In systems administration manuals, you find the term *unit of recovery* (UR), which relates to the same sequence of operations performed within a single resource manager.

Figure 3-1 shows an application flow over time and when during this application flow a UR exists.

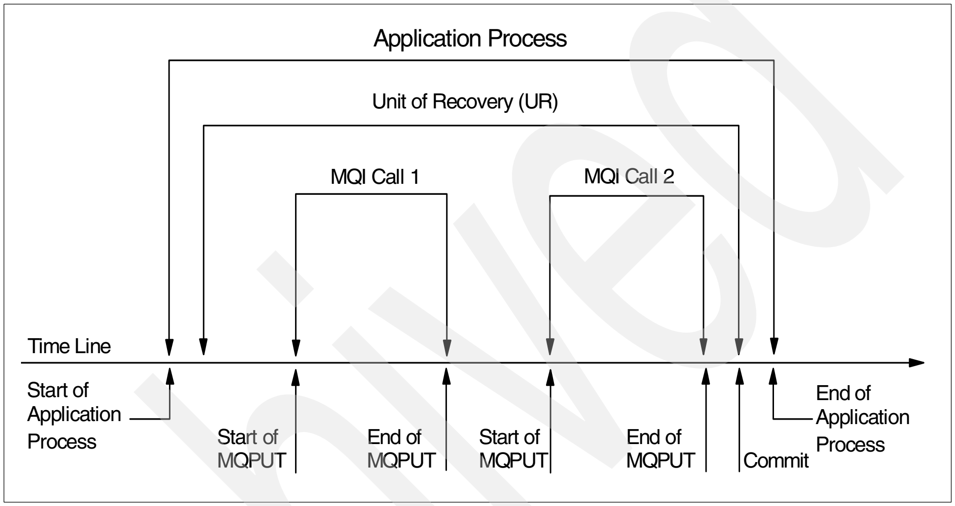


Figure 3-1 Application flow and UR

The application flow also shows that the UR can contain a set of coordinated changes.

3.1.3 Commit and backout

A program has two possibilities to finish a UR: commit or backout. The way to finish a UR depends on whether all updates are done successfully or not.

Commit

If, at the end of a UR, all updates are completed successfully, an application program can commit the UR. During the commit phase, all changes are made permanent.

A WebSphere MQ application program, for example, puts a message to a queue within a UR. This message is not visible to other programs until the UR is committed. After the commit the message is available to other programs for further processing.

Backout

On the other hand, if an application program detects an error during a UR, the application program can back out all changes done since the start of the UR. These changes, also known as pending changes, are not made, and the state of the data is as it was before the UR.

If a WebSphere MQ application program gets a message from a queue, this message remains on the queue until the program commits the unit of work, but the message is not available to be retrieved by other programs. If the program backs out the UR, WebSphere MQ restores the queue by making the message available to be retrieved by other programs.

3.1.4 Resource managers

To ensure that updates to resources are consistent, the following three programs work together:

- Application program

  The application program accesses protected resources and requests changes to the resources.

- Resource manager

  A resource manager controls and manages access to a resource. A resource manager provides an application programming interface (API) that allows the application to read and change the resources protected by the resource manager.

- Syncpoint manager

  The syncpoint manager coordinates changes to protected resources with all involved resource managers. To coordinate these changes it uses a two-phase commit protocol, described in 3.1.5, “Syncpoint coordination” on page 47.

There are three types of resource managers.

- Data resource manager

  A data resource manager is a resource manager, that allows the application to read and change data. DB2, for example, is a data resource manager. To process a syncpoint event, a data resource manager would either commit or back out the data it manages.

- Communications resource manager

  A communication resource manager controls access to distributed resources. It allows an application to communicate with other applications and resource managers, either located on the same system or on a different system. A communication resource manager acts as an extension to the syncpoint manager by allowing the local syncpoint manager to communicate with other remote syncpoint managers. APPC/MVS, for example, is a communication resource manager.

- Work manager

  A work manager controls application access to system resources by determining when and in what environment an application runs. A work manager ensures that the application is in the correct environment to allow the processing of a syncpoint event. For example, CICS Transaction Server and IMS Transaction Monitor are work managers.

A single resource manager can be more than one type of resource manager; for example, IMS is both a data resource manager and a work manager.

For more information on planning resource managers, refer to z/*OS MVS Programming: Resource Recovery,* SA22-7616.

3.1.5 Syncpoint coordination

An application can decide to synchronize data changes either at the end of processing or at certain logical points, which are called synchronization points. Other names for synchronization points are syncpoints or points of consistency.

The whole process by which units of work are either committed or backed out is called syncpoint coordination. The difference between single-phase commit and two-phase commit depends on whether the application program coordinates its changes with only one resource manager or with more than one resource manager.

Single-phase commit

In a single-phase commit process a program can make syncpoint-coordinated updates to only one kind of resource. Figure 3-1 on page 45 shows a WebSphere MQ application flow with a single-phase commit. Only updates to the queue are committed at the end of a UR.

Two-phase commit

Applications that need to coordinate updates with more than one resource manager can use the two-phase commit process to coordinate their updates with more than one resource manager. One of these resource managers has to be the coordinator (syncpoint manager); the others are participants.

For each UR there is one syncpoint manager and a number of resource managers. The syncpoint manager determines the outcome, either commit or backout for each UR.

In a batch z/OS application environment, RRS is the syncpoint manager and coordinates the two-phase commit process. When the application is ready to commit or back out its changes, it invokes RRS to begin the two-phase commit protocol.

The two-phase commit protocol is a set of actions used to make sure an application program makes all the changes to the collection of resources represented by a UR or makes no changes to the collection. Figure 3-2 shows a two-phase commit protocol with WebSphere MQ.

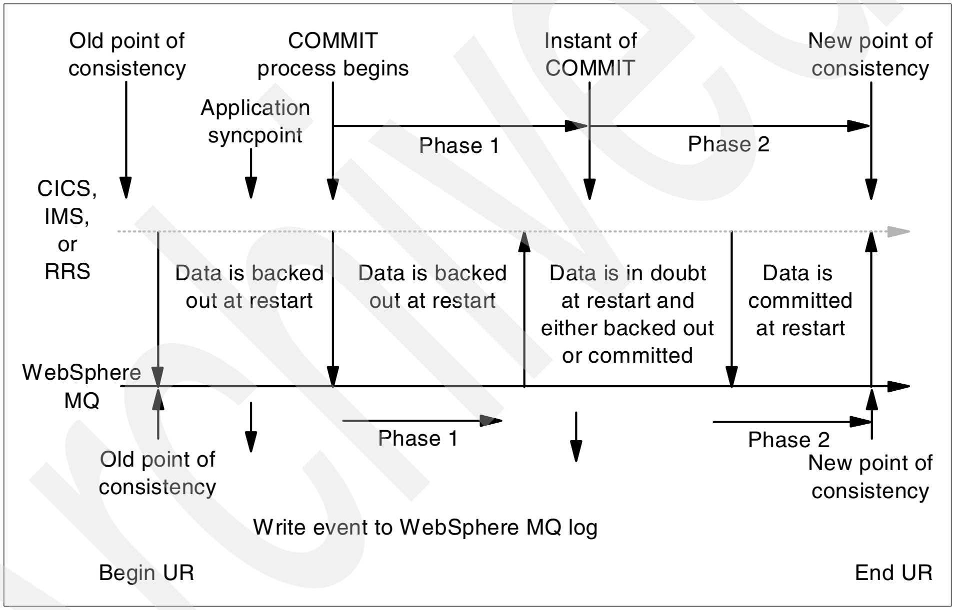


Figure 3-2 Two-phase commit protocol

The phases of the protocol are:

- Phase 1

  Each resource manager prepares to commit the changes by telling RRS that it agrees to allow the commit to continue. If the resource manager cannot commit the changes, it tells RRS to back out the changes. If all the resource managers agree to commit, RRS stores the decision in an RRS log, which means the decision is hardened, and phase 2 begins.

Once the commit decision is agreed to, the application changes are considered to be committed. If the application, any of the involved resource managers, RRS, or the system fails after the decision is hardened, the application changes are made during restart.

If the decision is to back out the changes, RRS generally does not harden the decision, and phase 2 begins as soon as the decision is made.

- Phase 2

  The resource managers commit or back out the changes represented by the UR.

For a detailed description of the two-phase commit protocol used by RRS, see Chapter 1, "Introducing Resource Recovery", in *z/OS MVS Programming: Resource Recovery,* SA22-7616.

3.2 RRS overview

Before we can describe the functions RRS covers, we have to introduce another term, the *exit manager*.

An exit manager is an authorized program that controls the flow of a predefined set of events. When such an event occurs, the exit manager gives control to an exit routine owned by a program interested in the event. The exit routine provides the processing for the event.

RRS is a component of z/OS and it functions as an exit manager. RRS provides two-phase commit support across participating resource managers. An application can update recoverable resources managed by various z/OS resource managers, such as WebSphere MQ, and then commit or back out the updates as a single UR. RRS provides the necessary UR status logging during normal execution, coordinates the syncpoint processing, and provides appropriate UR status information during subsystem restart.

Two other operating system components also play key roles in resource recovery:

- Registration services: The registration server coordinates the communication between a resource manager and RRMS. A resource manager must register itself with RRMS before it can request any RRMS service.
- Context services, which is also an exit manager, provides data constructs and primitives that resource managers use to track specific events related to given work requests. A context represents a specific unit of work and the components involved in processing it, such as programs, resource managers, and recoverable resources.

For a detailed description of these two components, refer to z/*OS MVS Programming: Resource Recovery,* SA22-7616.

RRS, context services, and registration services are three separate MVS components. The term *recoverable resource management services* (RRMS) covers all three functions. RRMS, which is also known as z/OS syncpoint manager, provides a systems programming interface (SPI) that enables a resource manager:

- To register with the system as a resource manager
- To express interest in work requests that access its resources
- To take part in resource recovery for those work requests

Since RRS covers much of the resource recovery function, in particular syncpoint processing, you may find the term RRS instead of RRMS, unless the documentation specifically describes context services or registration services.

3.3 Setting up and running RRS

Before you can actually use RRS with WebSphere MQ or other resource managers, for example CICS Transaction Server or DB2, you have to set up the RRS subsystem. Chapter 8, "Managing RRS", in z/*OS MVS Programming: Resource Recovery,* SA22-7616 describes these steps.

3.3.1 Log stream definition

RRS uses five log streams to log all synchronization requests. These log streams might be defined as DASD-only log streams or coupling facility log streams. Table 3-1 shows the five logs, their contents, and data set names.

The RRS images on different systems in a sysplex run independently but share log streams to keep track of the work. If a system fails, RRS on a different system in the sysplex can use the shared logs to take over the failed system's work.

Table 3-1 RRS log names, contents, and log data set names

RRS Log Names	Contents	Data Set Names
Archive log	Completed UR information	ATR.lgname.ARCHIVE
Resource manager data log	RRS resource manager information	ATR.lgname.RM.DATA
UR state log	State of active URs	ATR.lgname.MAIN.UR
Delayed UR state log	State of active URs, when UR completion is delayed	ATR.lgname.DELAYED.UR

RRS Log Names	Contents	Data Set Names
Restart log	Incomplete UR information needed during restart	ATR.lgname.RESTART

Since RRS uses log streams, you also have to set up MVS System Logger. For more information on MVS System Logger setup and definition of log streams, refer to *z/OS MVS Setting Up a Sysplex,* SA22-7625.

In our test environment we use coupling facility log streams and we have to define CFRM structures. Figure 3-3 shows the definitions for the CFRM structures.

```
STRUCTURE NAME(RRS_ARCHIVE_1)
          INITSIZE(16000)
          SIZE(72000)
          PREFLIST(CF05,CF06)
          REBUILDPERCENT(5)

STRUCTURE NAME(RRS_RMDATA_1)
         INITSIZE(8000)
         SIZE(16000)
         PREFLIST(CF06,CF05)
         REBUILDPERCENT(5)

STRUCTURE NAME(RRS_MAINUR_1)
          INITSIZE(8000)
          SIZE(48000)
          PREFLIST(CF05,CF06)
          REBUILDPERCENT(5)

STRUCTURE NAME(RRS_DELAYEDUR_1)
          INITSIZE(8000)
          SIZE(16000)
          PREFLIST(CF06,CF05)
          REBUILDPERCENT(5)

STRUCTURE NAME(RRS_RESTART_1)
          INITSIZE(8000)
          SIZE(48000)
          PREFLIST(CF05,CF06)
          REBUILDPERCENT(5)
```

Figure 3-3   CFRM structure definitions for RRS setup

You can define a new CFRM policy including the structure definitions for RRS and activate this policy with the following command:

```
SETXCF START POLICY,TYPE=CFRM,POLNAME=policy-name
```

where *policy-name* is the name you choose for your CFRM policy.

To define the RRS structures and log streams, use the utility IXCMIAPU. Figure 3-4 on page 53 shows the definitions for log streams that we defined for our RRS setup.

```
//ATRLOGR  JOB
//LOGR0001 EXEC PGM=IXCMIAPU
//SYSPRINT DD  SYSOUT=*
//SYSABEND DD  SYSOUT=*
//SYSIN    DD  *
  DATA TYPE(LOGR) REPORT(YES)

  DEFINE STRUCTURE NAME(RRS_ARCHIVE_1) LOGSNUM(1)
         MAXBUFSIZE(64000) AVGBUFSIZE(262)
  DEFINE STRUCTURE NAME(RRS_RMDATA_1) LOGSNUM(1)
         MAXBUFSIZE(1024) AVGBUFSIZE(252)
  DEFINE STRUCTURE NAME(RRS_MAINUR_1) LOGSNUM(1)
         MAXBUFSIZE(64000) AVGBUFSIZE(158)
  DEFINE STRUCTURE NAME(RRS_DELAYEDUR_1) LOGSNUM(1)
         MAXBUFSIZE(64000) AVGBUFSIZE(158)
  DEFINE STRUCTURE NAME(RRS_RESTART_1) LOGSNUM(1)
         MAXBUFSIZE(64000) AVGBUFSIZE(158)

  DEFINE LOGSTREAM NAME(ATR.SANDBOX.ARCHIVE)
         STRUCTNAME(RRS_ARCHIVE_1)
         LS_DATACLAS(SHARE33) HLQ(LOGR) LS_SIZE(1024)
         STG_DUPLEX(NO) RETPD(15) AUTODELETE(YES)
  DEFINE LOGSTREAM NAME(ATR.SANDBOX.RM.DATA)
         STRUCTNAME(RRS_RMDATA_1)
         LS_DATACLAS(SHARE33) HLQ(LOGR) LS_SIZE(1024)
         STG_DUPLEX(NO) RETPD(15) AUTODELETE(YES)
  DEFINE LOGSTREAM NAME(ATR.SANDBOX.MAIN.UR)
         STRUCTNAME(RRS_MAINUR_1)
         LS_DATACLAS(SHARE33) HLQ(LOGR) LS_SIZE(1024)
         STG_DUPLEX(NO) RETPD(15) AUTODELETE(YES)
  DEFINE LOGSTREAM NAME(ATR.SANDBOX.DELAYEDUR)
         STRUCTNAME(RRS_DELAYEDUR_1)
         LS_DATACLAS(SHARE33) HLQ(LOGR) LS_SIZE(1024)
         STG_DUPLEX(NO) RETPD(15) AUTODELETE(YES)
  DEFINE LOGSTREAM NAME(ATR.SANDBOX.RESTART)
         STRUCTNAME(RRS_RESTART_1)
         LS_DATACLAS(SHARE33) HLQ(LOGR) LS_SIZE(1024)
         STG_DUPLEX(NO) RETPD(15) AUTODELETE(YES)
```

Figure 3-4   JCL to define RRS log streams

To minimize any risk of losing data, you can use the unconditional duplexing System Logger option for the RRS logs, but be aware that this might cause performance problems. Setting unconditional duplexing ensures that all data is backed up to DASD staging data sets.

Although our test environment does not use staging data sets, we recommend that you use unconditional duplexing for the resource manager data log, since losing resource manager log data will force an RRS cold start. Example 3-1 shows the additional parameters for unconditional duplexing.

Example 3-1 RRS log stream parameters for duplexing

```
STG_DUPLEX(YES)
DUPLEXMODE(UNCOND)
```

To display information about the System Logger. MVS provides the command:

```
D LOGGER
```

If you use the parameter `STATUS` or no parameter, you get status information. To display Logger status, all information about the RRS structures, and log streams, issue the commands shown in Example 3-2. The generic names for the parameters `STRN` and `LSN` refer to the sample job provided in Figure 3-4 on page 53.

Example 3-2 Display commands to see RRS log stream and structure information

```
D LOGGER,STATUS
D LOGGER,STRUCTURE,STRN=RRS*
D LOGGER,LOGSTREAM,LSN=ATR*
```

3.3.2 RRS subsystem definition

You need to update the IEFSSNxx SYS1.PARMLIB member with the subsystem definition of RRS, as shown in Example 3-3.

Example 3-3 RRS subsystem definition

```
SUBSYS SUBNAME(RRS)
```

Ensure that the RRS subsystem definition is placed after the primary subsystem, and do not specify any other parameters. But you can change the subsystem name to a name of your choice. In our environment, the subsystem name is RRS, and we refer to this subsystem name in the following sections.

> **Note:** RRS does not support dynamic subsystem definition. Although you can use the command SETSSI ADD,SUBNAME=RRS, you cannot start RRS before you re-IPL the system.

RRS procedure

RRS provides a sample procedure ATRRRS in SYS1.SAMPLIB. Copy this procedure to your PROCLIB and change the procedure name to the subsystem name.

Example 3-4 RRS procedure

```
//RRS       PROC GNAME='',CTMEM=''
//RRS       EXEC PGM=ATRIMIKE,REGION=0M,TIME=NOLIMIT,
//               PARM='GNAME=&GNAME CTMEM=&CTMEM'
```

As shown in Example 3-4, the RRS procedure has two parameters:

- `CTMEM=` specifies the CTnRRSxx SYS1.PARMLIB member that RRS component trace is to use.
- `GNAME=` specifies the log group name. A log group is a group of systems that share RRS workload. If you do not specify a value for group name, the name defaults to the sysplex name.

3.3.3 RRS access authorization

In general, access to RRS administration functions should not be made available to all users. To control access to RRS resources and RRS system management functions, you can use the RACF component of SecureWay for z/OS. For a detailed description of how to control access, refer to Chapter 8, “Managing RRS”, in *z/OS MVS Programming: Resource Recovery,* SA22-7616.

3.3.4 Starting and stopping RRS

Once you have completed your setup, you can start RRS with the operator command START RRS. This start can be:

- A warm start.

  In a warm start, RRS can complete work that was in progress when a previous instance failed or was intentionally stopped. All RRS logs must be intact and available to RRS.

- A cold start.

  In contrast to a warm start, a cold start occurs when the RRS resource manager data log is empty.

You can use the operator command SETRRS CANCEL to stop RRS. If there are any non-resource manager programs in syncpoint, you might receive a X’058’ ABEND. If this ABEND occurs, transactions that were in progress will be resolved when RRS restarts. If SETRRS CANCEL does not stop RRS, you can use the FORCE RRS,ARM command.

3.3.5 ISPF screens for RRS administration

RRS provides an ISPF administration application to allow system administrators to work with RRS. Using this application you can:

- View RRS logs
- View UR information
- View resource manager information
- Determine, where a resource manager can restart after a system failure
- Resolve an in-doubt state for a UR
- Remove a resource manager's interest in a UR

In a Parallel Sysplex, you can configure RRS to allow a user to manage all the RRS images in the sysplex from a single screen.

Before you can use the RRS administration, you have to allocate the RRS libraries containing the screens, add the RRS application to the ISPF primary options menu or to a different menu of your choice, and set up access authorization.

Figure 3-5 shows the RRS main screen, where you can choose the action you want to perform. We show all RRS screens without the available PF keys. If you want to see, which PF keys are available, use the ISPF command PFSHOW ON.

```
                                        RRS
Option ===>

Select an option and press ENTER:

1 Browse an RRS log stream
2 Display/Update RRS related Resource Manager information
3 Display/Update RRS Unit of Recovery information
4 Display/Update RRS related Work Manager information
5 Display/Update RRS UR selection criteria profiles
6 Display RRS-related system information
```

Figure 3-5 RRS main screen

To see if a particular resource manager is registered to RRS, select option **2 Display/Update RRS related Resource Manager information**. For example, in our environment, we expect to see a WebSphere MQ queue manager MQGV and a DB2 subsystem DBG1.

Figure 3-6 on page 57 shows the RRS resource manager selection screen.

```
                        RRS Resource Manager Selection
Command ===>

Optionally provide resource manager name, system name, and logging group
patterns and press Enter:

RM name  . . . . . .
System name  . . . .
Logging Group  . . .

If left blank, system name and logging group will default to the current
system's name and group.
```

Figure 3-6   RRS resource manager selection screen

In order to get a list of all resource managers in the current system and logging group, we leave all input fields blank and just press the Enter key. Figure 3-7 shows the list of resource managers running on system SC63 in the logging group SANDBOX, which is also our sysplex name.

```
                          RRS Resource Manager List           Row 1 to 6 of 6
Command ===>                                                 Scroll ===> PAGE

Commands: v-View Details u-View URs r-Remove Interest

S   RM Name                         State             System   Logging Group
    CSQ.RRSATF.IBM.MQGV             Reset             SC63     SANDBOX
    CSQ.RRSATF.IBM.MQSA             Run               SC63     SANDBOX
    DSN.RRSATF.IBM.DBG1             Run               SC63     SANDBOX
    DSN.RRSATF.IBM.DB2G             Run               SC63     SANDBOX
    DSN.RRSPAS.IBM.DBG1             Run               SC63     SANDBOX
    DSN.RRSPAS.IBM.DB2G             Run               SC63     SANDBOX
******************************* Bottom of data *******************************
```

Figure 3-7   RRS list of resource managers

The output list shows the resource manager names and their current status. In Figure 3-7 queue manager MQGV with the resource manager name CSQ.RRSATF.IBM.MQGV is not running, and therefore its status is reflected as RESET. The third entry DSN.RRSATF.IBM.DBG1 refers to the DB2 subsystem we specified in CSQZPARM for our queue manager to connect to.

For more information on using the ISPF application and how to make the RRS ISPF application available, refer to z/*OS MVS Programming: Resource Recovery,* SA22-7616.

3.4 Programming with RRS

Now that we know the general concepts of resource recovery and RRS, we have to know how to communicate with RRS and how to use RRS in an application program.

3.4.1 RRS services and API

In order to allow resource managers to communicate with RRS, RRS provides:

- Callable registration services. These services are used by resource managers to register to RRS.
- Callable context services. A resource manager can use callable context services to request work context service.
- Callable resource recovery services can be used to request resource recovery services.

In order to allow application programs to communicate commit and/or backout requests to RRS, the callable resource recovery services include the following API functions:

- SRRCMIT: Application_Commit_UR
- SRRBACK: Application_Backout_UR

SRRCMIT and SRRBACK are the RRS-supplied verbs used for commit and backout in applications. They are found in SYS1.CSSLIB(ATRSCSS).

3.4.2 WebSphere MQ RRS adapters

WebSphere MQ provides two RRS adapters that allow batch applications that connect to WebSphere MQ, as well as to DB2 stored procedures, to use RRS coordinated commit processing. The adapters allow WebSphere MQ to be a full participant in RRS coordination of two-phase commit syncpoints.

The WebSphere MQ RRS adapters can be used in the same way as the Batch/TSO adapter. The WebSphere MQ RRS adapters support simultaneous connections to multiple WebSphere MQ queue managers running on a single z/OS instance from a single task, and they support the ability to switch a WebSphere MQ batch thread between TCBs.

The WebSphere MQ provided RRS adapters are:

1. CSQBRSTB

   This stub allows you to use two-phase commit and backout for applications using RRS callable resource recovery services instead of the MQI calls MQCMIT and MQBACK. CSQBRSTB requires you to use SRRCMIT and SRRBACK. If your program still uses MQCMIT or MQBACK, when linked with CSQBRSTB, you will receive MQRC_ENVIRONMENT_ERROR return code.

2. CSQBRSSI

   This stub allows you to use either MQI calls MQCMIT and MQBACK or SRRCMIT and SRRBACK. If you use MQCMIT and MQBACK, WebSphere MQ actually implements these calls as the SRRCMIT and SRRBACK RSS calls.

The bind step for an application that wants to use RRS must include one of these stubs. There exists no dynamic support for these stubs. Both stubs are shipped with linkage attributes AMODE(31) and RMODE(ANY).

3.4.3 WebSphere MQ and DB2 stored procedures using RRS

If you use DB2 stored procedures with RRS, you should be aware of the following:

- DB2 stored procedures that use RRS must run under control of WLM.
- If a DB2 stored procedure using WebSphere MQ calls is linked with either RRS stub, the MQCONN or MQCONNX call returns MQRC_ENVIRONMENT_ERROR.
- If a WLM-managed DB2 stored procedure contains WebSphere MQ calls and is linked with a non-RRS stub, the MQCONN or MQCONNX call returns MQRC_ENVIRONMENT_ERROR, unless it is the first WebSphere MQ call executed since the DB2 stored procedure address space started.
- If your DB2 stored procedure contains WebSphere MQ calls and is linked with a non-RRS stub, WebSphere MQ resources updated in that stored procedure are not committed until the stored procedure address space ends, or until a subsequent stored procedure does an MQCMIT.
- Multiple copies of the same DB2 stored procedure can execute concurrently in the same address space. You should ensure that your program is coded in a re-entrant manner if you want DB2 to use a single copy of your stored procedure.
- You must not code MQCMIT and MQBACK in a WLM-managed DB2 stored procedure.
- All programs must be designed to run in Language Environment (LE).

3.4.4 Migrating existing applications

It is possible to migrate existing batch/TSO WebSphere MQ applications to use RRS with very few changes. The two possible ways are:

1. If you link-edit your applications with the CSQBRRSI adapter, you can still use MQCMIT and MQBACK to syncpoint your unit of work across WebSphere MQ and all other RRS-enabled resource managers.
2. If you link-edit your applications with the CSQBRSTB adapter you have to ensure that all MQCMIT and MQBACK statements are changed to SRRCMIT and SRRBACK.

> **Note:** If RRS is not active on your system, any WebSphere MQ call in an application using CSQBRSTB or CSQBRSSI returns MQRC_ENVIRONMENT_ERROR.

Example 3-5 and Example 3-6 show the necessary code change in a C application to migrate from MQCMIT to SRRCMIT. Example 3-5 shows the necessary parts for MQCMIT.

Example 3-5 Code for MQCMIT in a C language application

```
/* *** program start, variable definitions ******************************** */
.
MQLONG     CompCode, Reason;
.
/* *** actual program code ********************************************** */
.
MQCMIT (Connhandle, &CompCode, &Reason);
if (CompCode != MQCC_OK)
   {
    printf ("MQCMIT  failed, reason: %d\n",Reason);
    free (malloc_addr);
    return(1);
   }
.
/* *** end of program *************************************************** */
```

To change to the SRRCMIT function, you have to add variables for the return code of SRRCMIT and change the commit statement, and of course you have to change your error processing to refer to the new return codes.

Example 3-6 Code for SRRCMIT in a C language application

```
/* *** program start, variable definitions ******************************** */
.
MQLONG     CompCode, Reason;
MQLONG     SRRcode;
```

```
int        rc;
.
/* *** actual program code ********************************************** */
.
MQCMIT (Connhandle, &CompCode, &Reason);
rc = SRRCMIT (&SRRcode);
if (rc != 0)
   {
    printf ("SRRCMIT  failed, reason: %d\n",SRRcode);
    free (malloc_addr);
    return(1);
   }
.
/* *** end of program **************************************************** */
```

3.5 Test scenarios for MQ applications using RRS

To test the RRS adapter with WebSphere MQ and to show the synchronization updates performed by RRS, we have two test programs:

1. The first program uses one queue manager and does only WebSphere MQ updates.
2. The second program updates WebSphere MQ resources and DB2 resources.

As described in 3.2, “RRS overview” on page 49 we know that the exit manager gives control to an exit routine of the resource manager when a certain event occurs. For our two test programs, we find that different exits are called:

1. If WebSphere MQ is the only participating resource manager in the syncpoint process, then RRS invokes the ONLYAGENT exit.
2. If a transaction makes requests to more than one resource manager, in our case WebSphere MQ and DB2, RRS invokes two exits:
   - PREPARE exit
   - COMMIT exit

If only one resource manager or if more than one resource managers are involved in syncpoint processing, and a backout request occurs, RRS calls the BACKOUT exit routine.

You can see which exits have been called by checking the RRS logs via the RRS ISPF application, which we introduced in 3.3.5, “ISPF screens for RRS administration” on page 56.

3.5.1 Scenario 1

The first example program uses one queue manager and does two MQPUTs. The program is linked with the RRS adapter CSQBRRSI, which allows the usage of MQCMIT. WebSphere MQ actually implements this call as the SRRCMIT call.

Figure 3-8 shows the link-edit step of our sample program. As you can see, you have to include the CSQBRRSI adapter module with:

```
INCLUDE MQMLIB(CSQBRRSI)
```

You should also add a DD statement MQMLIB that points to your WebSphere MQ load library, in our case MQ530.SCSQLOAD. You also have to include the RRS module with:

```
INCLUDE CSSLIB(ATRSCSS)
```

where `CSSLIB` is the reference to the DD statement of the SYS1.CSSLIB data set that contains ATRSCSS.

```
//LKED     EXEC PGM=IEWL,COND=(4,LT),
//              PARM='RENT,LIST,MAP,LET,XREF,AMODE(31),RMODE(ANY)'
//SYSPRINT DD  SYSOUT=*
//SYSLIB   DD  DISP=SHR,DSN=CEE.SCEELKED
//CSSLIB   DD  DISP=SHR,DSN=SYS1.CSSLIB
//MQMLIB   DD  DISP=SHR,DSN=MQ530.SCSQLOAD
//SYSLMOD  DD  DISP=SHR,DSN=MQRES1.LOADLIB
//OBJECT   DD  DISP=(OLD,DELETE),DSN=*.PLKED.SYSMOD
//SYSLIN   DD  DDNAME=SYSIN
//SYSUT1   DD  UNIT=SYSDA,SPACE=(32000,(30,30))
//SYSIN    DD  *
  INCLUDE OBJECT
  INCLUDE CSSLIB(ATRSCSS)
  INCLUDE MQMLIB(CSQBRRSI)
  NAME RRSM01B(R)
/*
```

Figure 3-8 JCL to link-edit a batch WebSphere MQ application with RRS adapter CSQBRRSI

To check the RRS logs, you can select option **1** from the RRS main screen, which is shown in Figure 3-5 on page 56. The next screen allows you to select which logs you want to display and whether you want a summary or a detailed level of report. Figure 3-9 on page 63 shows this selection screen. In our case, we need to select option **1 RRS Archive log** and option **2** for the level of report detail.

```
                        RRS Log Stream Browse Selection
Command ===>

Provide selection criteria and press Enter:


Select a log stream to view:                Level of report detail:
1   1.  RRS Archive log                     2   1.  Summary
    2.  RRS Unit of Recovery State logs         2.  Detailed
    3.  RRS Restart log
    4.  RRS Resource Manager Data log

RRS Group Name . . .          Default Group Name: : SANDBOX
Output data set  . . ATR.REPORT
```

Figure 3-9 RRS Log stream browse selection

In Figure 3-9 we do not show the whole screen, but omit the optional filtering part, because we do not use any filtering. If you decide to use filtering, you can specify date and time filtering as well as UR or resource manager filtering.

Note: To display the RRS archive log information including information regarding the call of exits, you have to select option **2** for a detailed report level, because this is the only option that will show the return codes from the exits and if the exits have been called.

Figure 3-10 on page 64 shows the output from the above selection. In the output you can easily find your information by searching on your job name. In this example we can see the `SYNCPOINT=Commit` parameter. We also can see that this syncpoint has been done by RRS, since the ONLYAGENT exit has been called and completed with RC00000000.

```
SC63     2002/08/15 14:30:22.870751 BLOCKID=000000000000018D
  URID=B81549517E81CDD0000000101020000 JOBNAME=MQRES1D  USERID=MQRES1
  PARENT URID=00000000000000000000000000000000
  SURID=N/A
  WORK MANAGER NAME=SC63.MQRES1D.0019
  SYNCPOINT=Commit  RETURN CODE=00000000
  START=2002/08/15 18:30:22.868314 COMPLETE=2002/08/15 18:30:22.870414
  EXITFLAGS=00800000
  LUWID=                                   TID=             GTID=

  FORMATID=            (decimal)          (hexadecimal)
  GTRID=

  BQUAL=

  RMNAME=CSQ.RRSATF.IBM.MQGV              ROLE=Participant
    FLAGS=10020000 PROTOCOL=PresumeAbort
    StateCheck EXIT RC=Uncalled
    Prepare    EXIT RC=Uncalled
    DistSp     EXIT RC=Uncalled
    Commit     EXIT RC=Uncalled
    Backout    EXIT RC=Uncalled
    EndUr      EXIT RC=Uncalled
    ExitFailed EXIT RC=Uncalled
    Completion EXIT RC=Uncalled
    OnlyAgent  EXIT RC=00000000
```

Figure 3-10   RRS archive log: ONLYAGENT exit invoked

3.5.2 Scenario 2

The second scenario does DB2 and WebSphere MQ updates in one UR. To use RRS with DB2, additional calls to DSNRLI have to be made. The program is link-edited with an RRS stub.

There are seven steps with different commit and backout scenarios in the executing job:

1. Put five messages to a WebSphere MQ queue, commit using SRRCMIT
2. Put five messages to a WebSphere MQ queue, back out using SRRBACK
3. Put five messages to a WebSphere MQ queue, commit using MQCMIT
4. Put five messages to a WebSphere MQ queue, back out using MQBACK
5. Put five messages to a WebSphere MQ queue, commit using SQL COMMIT

6. Put five messages to a WebSphere MQ queue, back out using SQL ROLLBACK
7. Put five messages to a WebSphere MQ queue, ABEND

All seven steps insert a record into a DB2 table, which contains the result of the MQPUT. After running this job, we check the RRS archive log and expect to see the commit and backout. As we are now using DB2 and WebSphere MQ, we do not see the ONLYAGENT exit called. This is because the program is link-edited with the RRS adapter and two resource managers are participating in the commit process. Rather, we see in Figure 3-11 on page 66 both the PREPARE and COMMIT routine called. In both cases we can also see the syncpoint return code of zeros.

```
SC63     2002/08/15 16:57:09.569403 BLOCKID=00000000000007BD
  URID=B8156A207E81CDD00000000501020000 JOBNAME=MQRES16  USERID=MQRES1
  PARENT URID=00000000000000000000000000000000
  SURID=N/A
  WORK MANAGER NAME=SC63.MQRES16.0019
  SYNCPOINT=Commit  RETURN CODE=00000000
  START=2002/08/15 20:57:09.554526 COMPLETE=2002/08/15 20:57:09.569237
  EXITFLAGS=00800000
  LUWID=                                    TID=             GTID=

  FORMATID=            (decimal)          (hexadecimal)
  GTRID=

  BQUAL=

  RMNAME=CSQ.RRSATF.IBM.MQGV              ROLE=Participant
    FLAGS=10001000 PROTOCOL=PresumeAbort
    StateCheck EXIT RC=Uncalled
    Prepare    EXIT RC=00000000
    DistSp     EXIT RC=Uncalled
    Commit     EXIT RC=00000010
    Backout    EXIT RC=Uncalled
    EndUr      EXIT RC=Uncalled
    ExitFailed EXIT RC=Uncalled
    Completion EXIT RC=Uncalled
    OnlyAgent  EXIT RC=Uncalled
  RMNAME=DSN.RRSATF.IBM.DBG1              ROLE=Participant
    FLAGS=10001000 PROTOCOL=PresumeNothing
    StateCheck EXIT RC=Uncalled
    Prepare    EXIT RC=00000000
    DistSp     EXIT RC=Uncalled
    Commit     EXIT RC=00000010
    Backout    EXIT RC=Uncalled
    EndUr      EXIT RC=Uncalled
    ExitFailed EXIT RC=Uncalled
    Completion EXIT RC=Uncalled
    OnlyAgent  EXIT RC=Uncalled
```

Figure 3-11   RRS archive log of a successful commit with DB2 and WebSphere MQ involved

Figure 3-11 also shows that RRS calls the exits for both resource managers, DB2 and WebSphere MQ.

If the programs initiates a backout or an ABEND occurs, the output shows that RRS invokes the backout exit routine, as you can see in Figure 3-12 on page 67.

```
SC63     2002/08/15 14:30:35.584985 BLOCKID=0000000000000319
  URID=B815495E7E81CDD00000000201020000 JOBNAME=MQRES1D  USERID=MQRES1
  PARENT URID=00000000000000000000000000000000
  SURID=N/A
  WORK MANAGER NAME=SC63.MQRES1D.0019
  SYNCPOINT=Backout RETURN CODE=00000000
  START=2002/08/15 18:30:35.579523 COMPLETE=2002/08/15 18:30:35.584803
  EXITFLAGS=02000000
  LUWID=                                   TID=             GTID=

  FORMATID=            (decimal)          (hexadecimal)
  GTRID=

  BQUAL=

  RMNAME=CSQ.RRSATF.IBM.MQGV               ROLE=Participant
    FLAGS=10000000 PROTOCOL=PresumeAbort
    StateCheck EXIT RC=Uncalled
    Prepare    EXIT RC=Uncalled
    DistSp     EXIT RC=Uncalled
    Commit     EXIT RC=Uncalled
    Backout    EXIT RC=00000010
    EndUr      EXIT RC=Uncalled
    ExitFailed EXIT RC=Uncalled
    Completion EXIT RC=Uncalled
    OnlyAgent  EXIT RC=Uncalled
```

Figure 3-12 RRS archive log of a backout scenario

3.5.3 Failing scenarios

If RRS fails during a syncpoint operation, the application terminates abnormally. If RRS fails before the application issues a commit or a backout, RRS ensures the application will receive an OUTCOME_PENDING return code for each incomplete UR. If a failure occurs when WebSphere MQ has completed phase 1 of the commit and is waiting for a decision from RRS, the UR enters an in-doubt state.

For all in-doubt URs, RRS does not issue a return code. Later, if RRS restarts without an IPL, and the application is still running, when the in-doubt UR is resolved, RRS automatically commits or backs out the UR, depending on whether there was a log record marking the beginning of the commit. WebSphere MQ cannot resolve these in-doubt URs until the connection to RRS is reestablished. If RRS is unable to recovery the in-doubt UR, a message is sent to the z/OS console.

3.6 A final word

In order to fulfill business requirements, applications often need to update more than one kind of protected resource in one UR. With RRS you are able to design and run your application as a z/OS batch application with two-phase commit protocol controlled by RRS as the syncpoint manager.

If you are going to use WebSphere MQ with RRS, you should keep in mind the following:

- When starting your z/OS image, be sure that RRS is started before you start your resource managers.
- If you are also in a queue-sharing group, you have to start DB2 before you can start your WebSphere MQ queue manager. For more information on queue-sharing groups, refer to Chapter 8, “Shared queue overview” on page 137 and Chapter 9, “Creating a shared queue environment” on page 159.
- To avoid MQRC_ENVIRONMENT_ERROR situations, be sure that your programs are using the correct commit and backout calls, and that your programs are link-edited with the correct RRS adapters.
- If you are migrating from MQI commit and backout calls to RRS commit and backout, remember that you have to update your error recovery processing.
- CICS or IMS adapters in WebSphere MQ do not use RRS functionality, because CICS and IMS are syncpoint managers.

Note: A WebSphere MQ queue manager running in a queue-sharing group always needs a running RRS.

4

Automatic Restart Manager (ARM)

The Automatic Restart Manager (ARM) is a z/OS recovery function that can improve the availability of your WebSphere MQ queue managers. ARM improves the time required to reinstate a queue manager by automatically restarting the batch job or started task (referred to as an element) when it unexpectedly terminates. ARM does this without operator intervention.

If a queue manager or a channel initiator fails, ARM can restart it on the same LPAR. If z/OS fails, ARM can restart WebSphere MQ and any related subsystem automatically on another LPAR within the sysplex.

4.1 ARM overview

ARM is a function introduced in MVS/ESA 5.2.0. It runs in the extended connection facility (XCF) address space and maintains its own data spaces. It provides an auto restart capability for failed job and started tasks. Transaction and database recovery is the responsibility of the relevant transaction manager or database manager. ARM only takes care of the *restart* of an application, not the initial start. Restart groups can be defined to enable a set of related applications to be restarted together on another LPAR in the sysplex. The sequence that applications are restarted in can also be controlled using the ARM policy.

4.2 Set up the ARM environment

There are five main steps required for ARM to be implemented in a Parallel Sysplex:

1. An ARM couple data set must be defined.
2. An ARM policy must be in place.
3. Your job or started task must be registered with ARM.
4. Ensure XCF authority is at the correct level.
5. Plan and implement any exits required.

It is also important to carefully plan your restart scenarios and which systems you want to include in your ARM policy, prior to implementation.

4.2.1 ARM couple data set

To enable program recovery through ARM, you need to create an ARM couple data set, using the IXCL1DSU utility. Sample JCL can be found in the IXCARMF member of SYS1.SAMPLIB. A system must be connected to an ARM couple data set with an active ARM policy in place.

Figure 4-1 on page 71 contains the JCL needed to format a couple data set for ARM to use. If any of the ITEM NAME(xxx) parameters are not specified, the default value for that item will take effect.

- `ITEM NAME(POLICY)` sets the maximum number of user-defined ARM policies that can be stored in the couple data set at any one time.
- `ITEM NAME(MAXELEM)` sets the maximum number of elements per policy. One started task or job would be an element.

- `ITEM NAME(TOTELEM)` sets the maximum number of elements that can be registered with ARM across the sysplex at any one time.

```
//IXCARMF  JOB
//ARMF0001 EXEC PGM=IXCL1DSU,REGION=2M
//SYSPRINT DD  SYSOUT=A
//SYSABEND DD  SYSOUT=A
//SYSIN    DD  *
  DEFINEDS SYSPLEX(WTSCPLX1)
           DSN(SYS1.XCF.CDS00) VOLSER(TOTDS0)
           MAXSYSTEM(16)
           CATALOG
  DATA TYPE(ARM)
           ITEM NAME(POLICY) NUMBER(5)
           ITEM NAME(MAXELEM) NUMBER(25)
           ITEM NAME(TOTELEM) NUMBER(20)
/*
```

Figure 4-1 Sample job to format couple data set for ARM using IXCL1DSU

After defining the couple data set, you can then define your ARM policies.

4.2.2 ARM policy

The ARM policy indicates how registered jobs or started tasks are to be restarted. The JCL in Figure 4-2 on page 72 has been extracted from SYS1.SAMPLIB(IXCARMP0). It shows the control statements required to define a sample ARM policy.

```
//IXCARMP0 JOB
//STEP1    EXEC PGM=IXCMIAPU
//SYSIN    DD   *
  DATA TYPE(ARM)
       DSN(SYS1.XCF.ARM00) VOLSER(TOTDS0) REPORT(YES)
  DEFINE POLICY NAME(POLICY1) REPLACE(YES)
  RESTART_ORDER
       LEVEL(5)
       ELEMENT_NAME(SUBSYS?0,SUBSYS?1,SUBSYS?2,
                    SUBSYS?3,SUBSYS?4,SUBSYS?5,
                    SUBSYS?6,SUBSYS?7,SUBSYS?8,
                    SUBSYS?9)
       ELEMENT_TYPE(AUTHREQ,
                    CUSTFILE,
                    SAVINGS)
  RESTART_GROUP(*)
       RESTART_PACING(250)
       ELEMENT(*)
            READY_TIMEOUT(250)

  DEFINE POLICY NAME(POLICY2) REPLACE(YES)
  RESTART_GROUP(NEWS@SERVER)
       TARGET_SYSTEM(*)
       ELEMENT(CORPNEWSSERVER)
            RESTART_METHOD(BOTH,STC,'S CICSNEWS')
/*
```

Figure 4-2 Sample JCL from SYS1.SAMPLIB(IXCARMP0) to define an ARM policy

The example shows the usage of a wild card '?' specification in the level assignments. Elements `SUBSYS?0 to SUBSYS?9`, where ? can be A-Z or 0-9, will be associated with LEVEL(5). `LEVEL` refers to the order in which the jobs will be restarted. Elements are restarted from the lowest level to the highest level. `LEVEL` is defined on all tasks that need to be restarted in an explicit order. Once defined, the ARM policy is stored in the couple data set. User-defined policies are optional, since ARM is shipped with default policy parameters that it applies when no policy is provided. Even when an installation-provided policy is active, it may not cover all possible scenarios. If ARM needs to use a parameter that has *not* been defined, it will take a default value.

You have two choices when deciding on which policy to use for ARM:

- Activate the default policy
- Build your own policy and activate it

The types of definitions required to build a user-defined ARM policy are described below. They fall into two categories:

- Groups of ARM elements
- Individual ARM elements

Groups of ARM elements

Prior to defining your ARM policy, you need to identify all elements with interdependencies on each other. These are referred to as *restart groups* and must be given a unique name. ARM is only concerned with restart groups when a system leaves the sysplex.

Individual ARM elements

The definitions for individual ARM elements are subentries of the restart group to which they belong. The parameters that relate to individual elements are:

1. Element name
2. Maximum number of restart attempts
3. When an element is to be restarted, whether after system termination, element termination, or if either are terminated.
4. Time interval allowed between ARM related events.
5. Start commands that ARM is to use.

You can also define the order in which you require the elements to be started by ARM. This is to control the restart sequence of related subsystems.

Refer to *z/OS MVS Setting Up a Sysplex,* SA22-7625, for detailed information on defining the ARM policy.

4.2.3 ARM services

Each task that wishes to use ARM services must register with ARM using the IXCARM macro. ARM is given control from IXCARM macro. The *z/OS MVS Sysplex Services Guide,* SA22-7617, contains a sample of this macro.

WebSphere MQ uses the IXCARM macro to register a queue manager or channel initiator to ARM during queue manager startup. The same macro is used to de-register from ARM during the normal shutdown process. Only one IXCARM request can be in progress from any one ASID at any time.

4.2.4 XCF authority

In order to perform restarts, ARM issues operator commands from the XCF address space. You must ensure XCF has enough authority to issue the commands required to restart a failed element.

4.2.5 ARM exits

There are three exits provided by ARM that can be implemented to influence how a restart is done. These are:

- Workload restart exit

  This exit is called once on the z/OS image that ARM is about to restart a failed element on. You can use this exit if you want to prepare your system for the additional workload ARM is about to start on it. An example of this would be if you wanted to cancel a low-priority job to free more system resources to be allocated to the new workload. This exit is called IXC_WORK_RESTART.

- Element restart exit

  This exit is called prior to ARM restarting a system. You can use this exit to coordinate your restart with other automation packages. This exit is called IXC_ELEM_RESTART.

- Event notification exit

  This exit is provided by the program that is registering to become an ARM element. You can use this exit to prepare a system for restart or prevent the restart of an system.

Note: The load modules for these exits must reside in LPA or LINKLIB. They need to run in key zero, supervisor state within the XCF ASID.

For more information on these exits, refer to *z/OS MVS Installation Exits* SA22-7593.

4.2.6 Summary of ARM components

Figure 4-3 on page 75 shows how the various ARM components fit together.

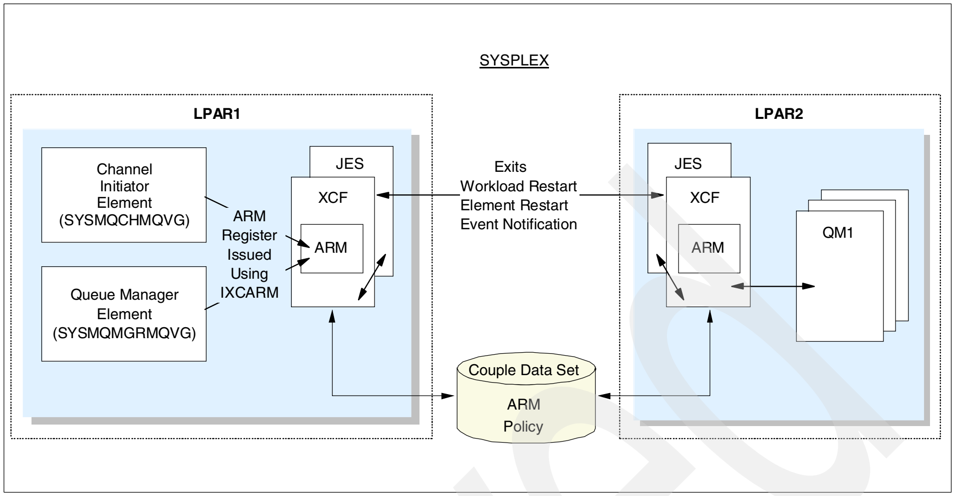


Figure 4-3 ARM components: the big picture

4.3 Displaying ARM status

You can confirm that the required ARM definitions are in place with a series of console commands. These are summarized in Table 4-1. Detailed syntax for commands can be found in *z/OS MVS System Commands,* SA22-7627.

Table 4-1 Commands to display status of ARM data sets and policy

Command	Display results
D XCF,COUPLE,TYPE=ARM	Definition of ARM couple data set
D XCF,POLICY,TYPE=ARM	Definition of ARM policy
D XCF,ARMSTATUS	Status of ARM
SETXCF START,POLICY TYPE=ARM	Start your ARM policy
SETXCF STOP,POLICY TYPE=ARM	Stop your ARM policy

4.4 Using ARM with WebSphere MQ

If an active ARM policy exists when WebSphere MQ is started, the queue manager and channel initiator will be automatically registered as ARM elements. If ARM is not active, the following procedure can be used to activate and test an WebSphere MQ ARM policy.

Step 1: Check if ARM is active

To check if ARM is active, you can use the ARMSTATUS command from Table 4-1 on page 75. Example 4-1 shows the output received from this command. In this case we can see that ARM is not enabled.

Example 4-1 Response from D XCF,ARMSTATUS command

```
D XCF,ARMSTATUS
IXC392I  10.55.11  DISPLAY XCF 464
ARM RESTARTS ARE NOT ENABLED
```

Step 2: Check for any active policy for WebSphere MQ

Example 4-2 shows the output from D XCF,POLICY,TYPE=ARM command. In this case, the results show that there is no policy started.

Example 4-2 Response from D XCF,POLICY,TYPE=ARM command

```
D XCF,POLICY,TYPE=ARM
IXC364I  11.53.16  DISPLAY XCF 944
TYPE: ARM
POLICY NOT STARTED
```

Step 3: Implement a new ARM policy for WebSphere MQ

Figure 4-4 contains a sample JCL used to define an ARM policy. A WebSphere MQ queue manager will use the element name SYSMQMGR*ssid,* where *ssid* is the four-character queue manager name of MQV2. A detailed description of all ARM parameters can be found in the *z/OS MVS Setting Up a Sysplex,* SA22-7625.

```
//STEP1    EXEC PGM=IXCMIAPU
//SYSIN    DD  *
  DATA TYPE(ARM)
  DEFINE POLICY NAME(ARMPOLMQ)
  RESTART_GROUP(MQV2)
       TARGET_SYSTEM(*)
       RESTART_PACING(20)
       ELEMENT(SYSMQMGRMQV2)
            RESTART_ATTEMPTS(3,300)
            RESTART_TIMEOUT(120)
            TERMTYPE(ALLTERM)
            RESTART_METHOD(BOTH,STC,'-MQV2 START QMGR')
/*
```

Figure 4-4 Sample job to define an ARM policy

The parameters used in this sample policy are as follows:

- `TARGET_SYSTEM(*)` specifies the systems on which elements can be restarted in a cross-system restart. A value of * means any system running ARM could be the target system.
- `ELEMENT(SYSMQMGRMQV2)` specifies the batch or started task that can register as an element of ARM.
- `RESTART_ATTEMPTS(3,300)` specifies the maximum number of times that ARM should attempt to restart the specified element within a given interval. In this case, ARM will attempt a maximum of three restarts over a 300-minute period. The element will be de-registered from ARM when this threshold is reached.
- `RESTART_TIMEOUT(120)` specifies a time limit for a restarted job to re-register with ARM. If this value is exceeded, then the element will be de-registered.
- `TERMTYPE(ALLTERM)` specifies whether the element will be restarted for all unexpected terminations (including system failure) or just when the element itself terminates. In this case we have specified ALLTERM, to cater for all terminations.
- `RESTART_METHOD(BOTH,STC,'-MQV2 START QMGR')` specifies the conditions for restarting a task and the command text. In this case `BOTH` means that if the element or system it is running on fails, restart the element using the -MQV2 START QMGR command.

The full syntax of the administrative commands can be found in *z/OS MVS Setting Up a Sysplex,* SA22-7625.

Step 4: Activate the new ARM policy

This is done with using the SETXCF command. Figure 4-5 on page 78 shows the complete command and the command results. The D XCF,ARMSTATUS command has also been re-issued to show that ARM restarts are now ENABLED.

```
Command to activate policy:

SETXCF START,POLICY,TYPE=ARM,POLNAME=armpolmq

IXC805I ARM POLICY HAS BEEN STARTED BY SYSTEM SC53.
POLICY NAMED ARMPOLMQ IS NOW IN EFFECT.

Command to verify policy activation:

D XCF,ARMSTATUS

  IXC392I  13.43.22  DISPLAY XCF 566
  ARM RESTARTS ARE ENABLED
  -------------- ELEMENT STATE SUMMARY --------------   -TOTAL-  -MAX-
  STARTING  AVAILABLE  FAILED  RESTARTING  RECOVERING
         0          9       0           0           0        9     50
```

Figure 4-5 Activating an ARM policy

Step 5: Verify the succesful registry at startup

Next we verify that the queue manager and channel initiator started tasks successfully register with ARM on startup. Figure 4-6 on page 79 shows us a job log extract for both address spaces showing that the ARM registration is successful. The element names registered are SYSMQMGRMQV2 for (MSTR ASID) and SYSMQCHMQV2 (CHIN ASID). Also, the D XCF,ARMS,JOBNAME=MQV2MSTR,DETAIL command can be issued to find out the current ARM status of an element. The results from this command show current statistics such as the first and last ARM restart time, and that the job is currently available for ARM restarts.

```
Job Log: MQV2MSTR
 CSQY201I -MQV2  CSQYSTRT ARM REGISTER for element
 SYSMQMGRMQV2 type SYSMQMGR successful

Job Log: MQV2CHIN
 CSQY201I -MQV2 CSQXJST ARM REGISTER for element
 SYSMQCHMQV2 type  SYSMQCH successful:


D XCF,ARMSTATUS,JOBNAME=MQV2MSTR,DETAIL
IXC392I  12.18.01  DISPLAY XCF 877
ARM RESTARTS ARE ENABLED
-------------- ELEMENT STATE SUMMARY --------------   -TOTAL-  -MAX-
STARTING  AVAILABLE  FAILED  RESTARTING  RECOVERING
       0          1       0           0           0        1    400
RESTART GROUP:DEFAULT          PACING  :    0    FREECSA:    0       0
ELEMENT NAME :SYSMQMGRMQV2     JOBNAME :MQV2MSTR STATE   :AVAILABLE
  CURR SYS :SC61               JOBTYPE :STC       ASID    :03F9
  INIT SYS :SC61               JESGROUP:XCFJES2A TERMTYPE:ALLTERM
  EVENTEXIT:*NONE*             ELEMTYPE:SYSMQMGR LEVEL   :    2
  TOTAL RESTARTS :        3    INITIAL START:08/16/2002 12:01:11
  RESTART THRESH :   0 OF 3    FIRST RESTART:08/16/2002 12:02:20
  RESTART TIMEOUT:      300    LAST  RESTART:08/16/2002 12:08:13
```

Figure 4-6   ARM registration messages from console log

Step 6: Verify that ARM policy definitions are correct

We now verify, that we specified the correct policy options.

Enter the -MQV2 STOP QMGR MODE(RESTART) shutdown command.

Normally, WebSphere MQ is deregistered when the queue manager is stopped with either STOP QMGR MODE(QUIESCE) or STOP QMGR MODE(FORCE).

The STOP QMGR MODE(RESTART) command will not deregister the queue manager from ARM, so it is eligible for immediate automatic restart. If the ARM policy has been set up correctly, the queue manager will be restarted immediately. For this restart, the CSQY201I ARM registration message will not be issued by the queue manager because it is still registered from the previous startup. Figure 4-7 on page 80 shows messages from the console log indicating that ARM has automatically restarted the started task.

```
IXC812I JOBNAME MQV2MSTR, ELEMENT SYSMQMGRMQV2 FAILED. 944
THE ELEMENT WAS RESTARTED WITH OVERRIDE START TEXT.
IXC813I JOBNAME MQV2MSTR, ELEMENT SYSMQMGRMQV2 945
WAS RESTARTED WITH THE FOLLOWING START TEXT:
-MQV2START QMGR
THE RESTART METHOD USED WAS SPECIFIED BY THE ELEMENT ON REGISTRATION.
$HASP100 MQV2MSTR ON STCINRDR
IEF695I START MQV2MSTR WITH JOBNAME MQV2MSTR IS ASSIGNED TO USER STC
   , GROUP SYS1
$HASP373 MQV2MSTR STARTED
IEF403I MQV2MSTR - STARTED - TIME=12.53.00 - ASID=03FA - SC61
```

Figure 4-7 Console messages issued during an ARM restart

Only the queue manager should be restarted by ARM. The channel initiator should be restarted from CSQINP2 initialization data set. You should set up your WebSphere MQ environment such that the channel initiator and associated listeners are started automatically when the queue manager is restarted.

> **Note:** In fact, the channel initiator is always deregistered from ARM when it is stopped normally. It is also deregistered when the queue manager is stopped normally or abnormally. The only instance when the channel initiator remains registered is when it ends abnormally. This is the only time that ARM should be allowed to restart the channel initiator.

4.4.1 ARM restart on the same image

To enable ARM to restart WebSphere MQ on the same z/OS image with either TCP/IP or LU 6.2 communications, you need to do the following:

1. Start the channel initiator automatically by adding START CHINIT to the CSQINP2 data set.
2. Start the listeners automatically by adding START LISTENER TRPTYPE(LU62) or START LISTENER TRPTYPE(TCP) to the CSQINPX data set

Remote channel connections will be able to reconnect to the same LU62 or TCP channels without any configuration change.

For more information on CSQINP2 and CSQPINX, refer to *WebSphere MQ System Setup Guide,* SC34-6052.

4.4.2 ARM restart (LU6.2 listener) on a different image

To restart WebSphere MQ on a different z/OS image and re-enable LU62 connectivity, you need to perform the following steps:

- Define each queue manager with a unique subsystem name.
- Define each channel initiator with a unique LUNAME.
- Add an entry for each channel initiator's LUNAME in the APPC side information data set. Refer to *MVS Planning APPC/MVS Management,* SA22-7599, for more information on APPC side information.
- Add an entry in APPCPMxx member of SYS1.PARMLIB for each channel initiator. This activates the APPC side information that was defined above. This member has to be shared across all z/OS images.
- Use the LU62ARM keyword of the CSQ6CHIP macro to specify the xx suffix of the SYS1.PARMLIB member. When the channel initiator is restarted, the appropriate LUNAME contained in the SYS1.PARMLIB member is activated by a z/OS SET APPC=xx command. This command is automatically issued during an ARM restart of the queue manager and associated channel initiator on a different z/OS image.

Figure 4-8 on page 82 shows how WebSphere MQ LU62 communications are reestablished for queue manager MQV2 after an LPAR failure.

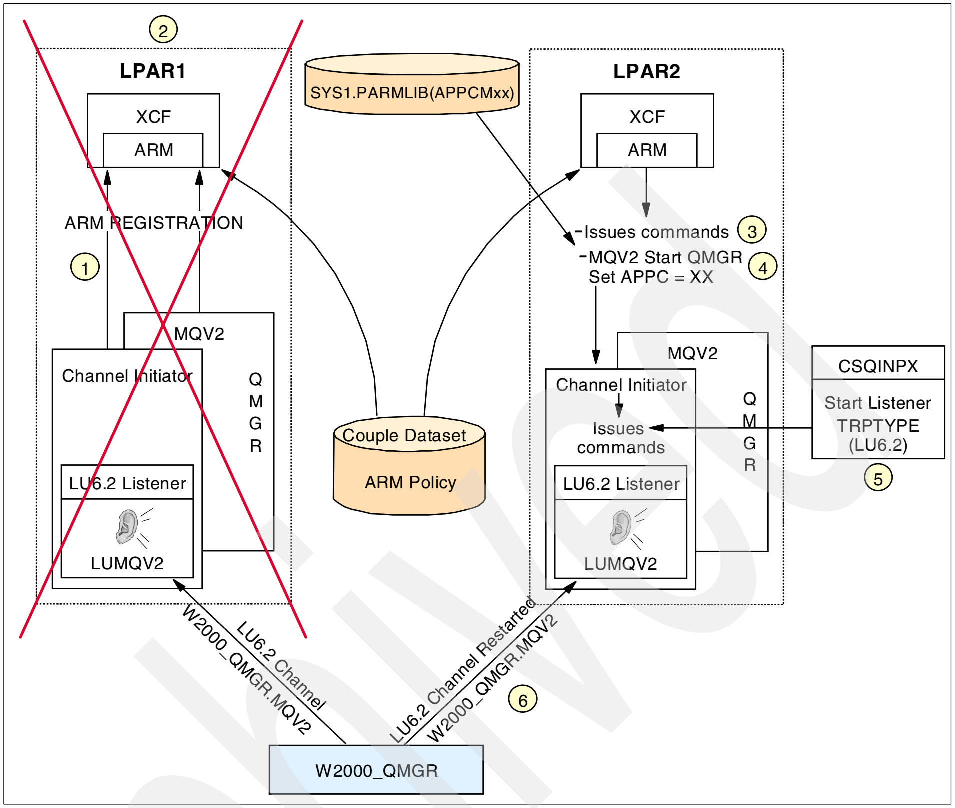


Figure 4-8 Restart process for LU62 communication (ARM restart on a different LPAR)

The following steps summarize the queue manager LU62 recovery shown in Figure 4-8.

1. The queue manager MQV2 is started on LPAR1 and registers with ARM.
2. LPAR1 experiences an outage.
3. ARM issues a restart for queue manager MQV2 on LPAR2.
4. During ARM restart, another command is issued to activate APPC side info to allow MQV2 to use LU62 on that LPAR.
5. QMGR restarts LU62 listener using a start listener command from the CSQINPX data set.
6. Client queue manager W2000_QMGR successfully restarts the LU62 channel.

4.4.3 ARM restart (TCP listener) on a different image

To restart WebSphere MQ on a different z/OS image and reestablish TCP connectivity, there are a number of options depending on the method used to establish the TCP connection:

1. If virtual IP addresses are being used, they can be automatically recovered on the new image. All queue managers that were connected to that queue manager will be able to reconnect as soon as this has occurred. The time taken to restart the channel connection will also depend on the short and long retry parameters on the remote queue managers sending channel definition.
2. If WebSphere MQ clustering is being used, the new IP address can be advertised to the rest of the cluster by simply modifying the channel initiator command input file (CSQINPX).

   A unique channel initiator command input file (CSQINPX) will need to be created for every queue manager/LPAR combination. This file will have a command to start the TCP listener and an ALTER CHANNEL command to change the connection name on the cluster receiver channel to reflect the new IP address and port. Refer to Example 4-3.

   Example 4-3 Alter channel to reflect the new IP address and port for ARM restart

   ```
   ALTER CHANNEL(TO.MQV2) CHLTYPE(CLUSRCVR) CONNAME(9.12.7.8(1500))
   ```

   After processing these commands during restart, the updated connection name will be sent to the cluster repository. The details will then be published to all queue managers with an interest in queue manager MQV2. Any new channel connections will then be successful using the new IP address.
3. If Dynamic DNS is being used, you define a name server within the z/OS sysplex and DNS can utilize WLM to select the best system to process the connection request. This method of connection would be useful if cloned WebSphere MQ queue managers and applications are running across the sysplex. If you specify a secondary name server within your sysplex, this server can take over the functions of the initial server should it fail.

   When a queue manager fails and ARM restarts it on another z/OS LPAR, the group listener registers with DNS from this new image. This can occur with no operator intervention required, and no changes to the channel definitions. Any channels that were interrupted will be able to reconnect to the same queue manager once ARM restart is complete. For more information on WLM and DNS, refer to *TCP/IP in a Sysplex,* SG24-5235.
4. Dynamic VIPA

   VIPA is the *virtual* IP address. Traditional IP addresses were actual device addresses. VIPA is an IP address that is not dependent on a physical device, but exists in software. VIPA is device independent. If ARM restarts a queue manager on another LPAR, inbound connection requests specifying a VIPA address and a predefined port can be routed to the same queue manager running on the new LPAR. For more information on TCP/IP and Dynamic VIPA, refer to *TCP/IP in a Sysplex,* SG24-5235. VIPA addressing is also discussed in 5.2, “What are VIPA and DVIPA?” on page 89.

Figure 4-9 shows how WebSphere MQ TCP communications are reenabled and the new IP address is advertised to the cluster when queue manager MQV2 is restarted on a different LPAR.

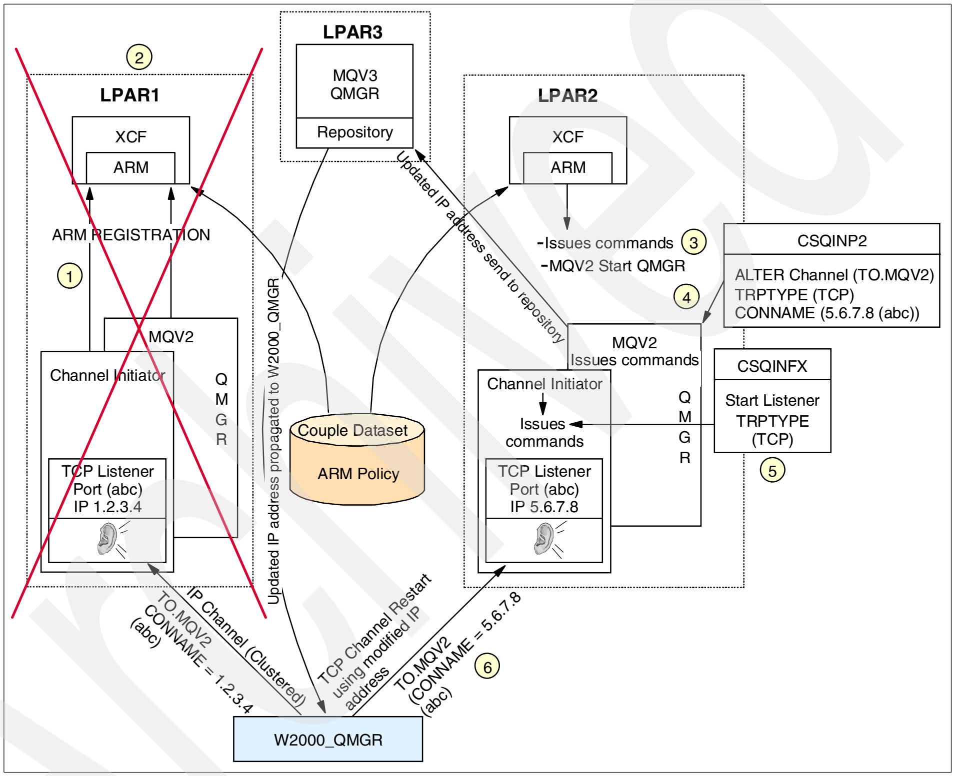


Figure 4-9 Restart process for TCP communication (ARM restart on different LPAR)

The following steps summarize the queue manager TCP communication recovery shown in Figure 4-9:

1. The queue manager MQV2 is started on LPAR1 and registers with ARM.
2. LPAR1 experiences an outage.
3. ARM issues a restart for queue manager MQV2 on LPAR2.
4. QMGR issues the ALTER CHANNEL command for the cluster receiver channel, specifying the new IP address required for all new connections.

5. QMGR restarts TCP listener using a START LISTENER command from the CSQINPX data set.
6. Client queue manager W2000_QMGR successfully restarts TCP cluster sender channel.

5

Sysplex Distributor

Sysplex Distributor is a function introduced in IBM Communications Server for OS/390 V2R10 IP. It uses the XCF dynamics and Dynamic VIPA support to provide a very high level of availability and workload balancing in a sysplex.

Workload can be distributed to multiple server instances within the sysplex without requiring changes to clients or networking hardware and with virtually no delays in connection setup.

Communications Server for z/OS IP provides the way to implement a Dynamic VIPA as a single network visible IP address for a set of hosts that belong to the same sysplex. Any client located anywhere in the IP network is able to see the sysplex as one IP address regardless of the number of hosts that it includes.

Because of these advantages of the Sysplex Distributor, we decided that this would be the best network solution for our queue-sharing group. In this chapter, we give a brief overview of the functionality and setup of the Sysplex Distributor.

For more detailed information, refer to *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 5,* SG24-6517.

5.1 What is Sysplex Distributor?

Sysplex Distributor was designed to address the requirement of one single network-visible IP address for the sysplex cluster and let the clients in the network receive the benefits of workload distribution and high availability within the sysplex cluster. With Sysplex Distributor, client connections seem to be connected to a single IP host even if the connections are established with different servers in the same sysplex cluster.

Because the Sysplex Distributor function resides on a system in the sysplex itself, it has the ability to factor "real-time" information concerning the multiple server instances including server status as well as Quality of Service (QoS) and policy information provided by CS for z/OS IP's Service Policy Agent. By combining these "real-time" factors with the information from the WLM, the Sysplex Distributor has the unique ability to ensure that the best destination server instance is chosen for a particular client connection. The Sysplex Distributor has more benefits than other load-balancing implementations, such as the Network Dispatcher or DNS/WLM. Their limitations are removed with the Sysplex Distributor. In summary, the benefits of the Dynamic VIPA and the Sysplex Distributor include:

1. Removes configuration limitations of Network Dispatcher.

   Target servers can use XCF links between the distributing stack and target servers as opposed to LAN connections such as an Open Systems Adapter (OSA).

2. Removes dependency of specific hardware in WAN.

   Provides a total CS for z/OS IP solution for workload distribution.

3. Provides real-time workload balancing for TCP/IP applications even if clients cache the IP address of the server (a common problem for DNS/WLM).
4. Enhances VIPA takeover and takeback support.

   Allows for nondisruptive takeback of the VIPA original owner to get workload where it belongs. The distributing function can be backed up and taken over.

5. Enhances Dynamic VIPA support and non-disruptive application server instance movement

In summary, Dynamic VIPA and the Sysplex Distributor provide:

- Single network-visible IP address of a sysplex cluster service. One IP address can be assigned to the entire sysplex cluster (usually for each service provided, such as Telnet).

Sysplex Distributor will query the Policy Agent to find if there exists any policy defined for routing the incoming connection requests. WLM and QoS policy can be specified for workload balancing in real-time on every new connection request.

- From OS/390 V2R10, it raises the limit of 64 DVIPAs on a stack to 256.
- Backup capability is enhanced. In case of failure of the distributing IP stack, the connections distributed to other IP stacks in the sysplex will not be disrupted.
- Dynamic VIPA takeback without any disruption in the connections already established.
- Commands to display both the connection routing table and destination port table information, showing information of configuration and current connection distribution are available.

5.2 What are VIPA and DVIPA?

A VIPA (virtual IP address) is configured the same way as a normal IP address for a physical adapter, except that it is not associated with any particular interface. VIPA uses a virtual device and a virtual IP address that other TCP/IP hosts can use to select an OS/390 IP stack without choosing a specific network interface on that stack. The virtual IP address is added to the home address list as shown in Figure 5-1 on page 90. The virtual device defined for the VIPA is always active and never fails.

Dynamic VIPA was introduced by SecureWay Communications Server for OS/390 V2R8 IP to enable the dynamic activation of a VIPA as well as the automatic movement of a VIPA to another surviving z/OS image after a z/OS stack failure. There are two forms of Dynamic VIPA, both of which can be used for takeover functionality:

- Automatic VIPA takeover allows a VIPA address to move automatically to a stack (called a backup stack) where an existing suitable application instance is already active and allows the application to serve the client formerly going to the failed stack.
- Dynamic VIPA activation for an application server allows an application to create and activate VIPA so that the VIPA moves when the application moves.

Nondisruptive, immediate, automatic VIPA takeback was introduced by IBM Communications Server for OS/390 V2R10 to move the VIPA back to where it originally belongs once the failed stack has been restored. This takeback is nondisruptive to existing connections with the backup stack and the takeback is not delayed until all connections with the backup stack have terminated (as was the case with CS for OS/390 V2R8 IP). New connections are handled by the new (original) primary owner, thereby allowing the workload to move back to the original stack.

For more information about VIPA and how it is defined, refer to *Communications Server for z/OS V1R2 Implementation Guide Volume 5: Availability, Scalability, and Performance, S*G24-6517.

.

```
D TCPIP,TCPIPA,N,HOME
EZZ2500I NETSTAT CS V2R10 TCPIPA 874
HOME ADDRESS LIST:
ADDRESS          LINK             FLG
9.12.6.85        OSA2860LNK       P
10.1.100.61      EZASAMEMVS
10.1.100.61      EZAXCF62
10.1.100.61      EZAXCF48
10.1.100.61      EZAXCF53
10.1.100.61      EZAXCF47
10.1.100.61      EZAXCF69
10.1.100.61      EZAXCF04
10.1.100.61      EZAXCF43
10.1.100.61      EZAXCF42
10.1.100.61      EZAXCF67
10.1.100.61      EZAXCF50
10.1.100.61      EZAXCF66
10.1.100.61      EZAXCF49
10.1.100.61      EZAXCF54
10.1.100.61      EZAXCF52
10.1.100.61      EZAXCF55
9.12.8.10        VIPL090C080A
127.0.0.1        LOOPBACK
19 OF 19 RECORDS DISPLAYED
```

Figure 5-1 Display of the home address list including the VIPA 9.12.8.10

5.3 VIPA takeover and takeback

There is always one stack defined as the DVIPA owner within the sysplex, and one or more are defined as backup. All stacks share information regarding DVIPA using z/OS XCF messaging so that all stacks know which stack currently owns the DVIPA and which stacks will participate in backup if the active one fail.

Figure 5-2 shows the output of the z/OS D TCPIP,SYSPLEX,VIPADYN command, which displays the information for all stacks participating in the sysplex at once. You see SC61 as the active (owning) stack and SC62 as the backup.

```
D TCPIP,TCPIPA,SYSPLEX,VIPADYN
EZZ8260I SYSPLEX CS V2R10 472
VIPA DYNAMIC DISPLAY FROM TCPIPA   AT SC61
IPADDR: 9.12.8.10  LINKNAME: VIPL090C080A
  ORIGIN: VIPADEFINE
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPIPA   SC61     ACTIVE      255.255.255.0   9.12.8.0        BOTH
  TCPIPA   SC62     BACKUP 100                                  DEST
2 OF 2 RECORDS DISPLAYED
```

Figure 5-2 SYSPLEX VIPADYN for all stacks participating in the sysplex

Figure 5-3 shows the output from the z/OS D TCPIP,NETSTAT,VIPADYN command for the IP stacks participating in the sysplex cluster. The output for this command shows the actual Dynamic VIPA information for the local host. With this command, you can see if the DVIPA is active or a backup for the local stack. In addition, the command shows if the DVIPA is being used as the distributing VIPA or destination VIPA.

```
SC61,D TCPIP,TCPIPA,N,VIPADYN
D TCPIP,TCPIPA,N,VIPADYN
EZZ2500I NETSTAT CS V2R10 TCPIPA 549
IP ADDRESS      ADDRESSMASK     STATUS    ORIGINATION     DISTSTAT
9.12.8.10       255.255.255.0   ACTIVE    VIPADEFINE      DIST/DEST
1 OF 1 RECORDS DISPLAYED

SC62,D TCPIP,TCPIPA,N,VIPADYN
D TCPIP,TCPIPA,N,VIPADYN
EZZ2500I NETSTAT CS V2R10 TCPIPA 306
IP ADDRESS      ADDRESSMASK     STATUS    ORIGINATION     DISTSTAT
9.12.8.10       255.255.255.0   BACKUP    VIPABACKUP      DEST
1 OF 1 RECORDS DISPLAYED
```

Figure 5-3 NETSTAT VIPADYN output for SC61 and SC62 IP stacks

In case of a failure, Automatic VIPA Takeover allows a DVIPA to move automatically to the first stack in the backup list. This stack automatically defines DEVICE, LINK and HOME statements for the same Dynamic VIPA and notifies its attached routing daemon (that is, OMPROUTE or ORouteD) of the activation. This information is passed onto the routing network through dynamic routing protocols (that is OSPF (Open Shortest Path First) or RIP (Routing Information Protocol)) and ultimately ensures that the DVIPA is still reachable. The previous backup is now the DVIPA owner and new incoming connection requests are automatically sent to it.

Figure 5-4 shows a sysplex with three TCP/IP images. In this example, stack SC61 fails and its DVIPA is subsequently taken over by SC62, which is providing the backup capability for this address.

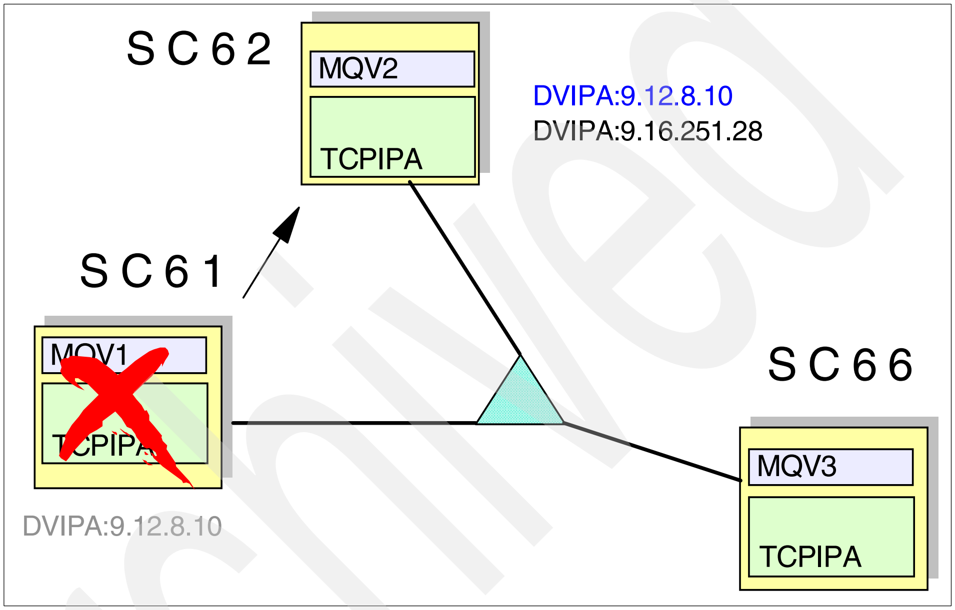


Figure 5-4 Automatic VIPA takeover of 9.12.8.10

The console message for the takeover is as follows:

```
EZZ8301I VIPA 9.12.8.10 TAKEN OVER FROM TCPIPA ON SC61
```

Once the failing stack is activated again, the Dynamic VIPA is moved from the backup stack back to the original owning stack automatically. In this case, new connections are sent to the reactivated stack and connections with the backup stack are not necessarily broken.

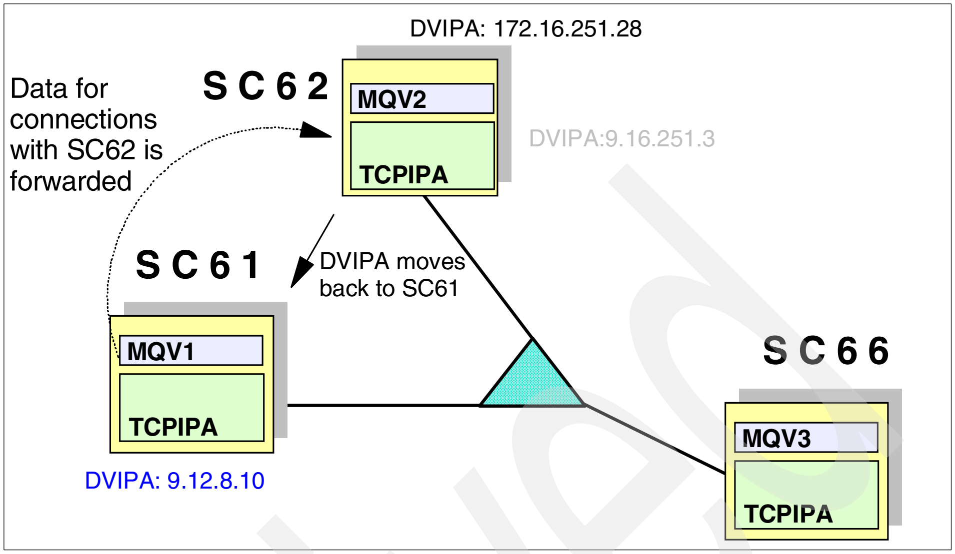


Figure 5-5 9.12.8.10 is taken back by the restarted SC61 stack

Example 5-1 DVIPA takeback

```
EZZ8302I VIPA 9.12.8.10 TAKEN FROM TCPIPA ON SC62
EZZ8303I VIPA 9.12.8.10 GIVEN TO TCPIPA ON SC61
```

Note: IP stacks at CS for OS/390 V2R7 and previous releases do not support Automatic VIPA backup. IP stacks at V2R8 may back up a distributing Dynamic VIPA but takeback is disruptive, whereas at V2R10 and higher the takeback is non disruptive.

5.4 Workload management

All IP stacks in the sysplex have DYNAMICXCF parameter coded in their TCPIP profile. When the distributing stack (the one that owns the Dynamic VIPA) is activated, it dynamically creates its local XCF link due to DYNAMICXCF being coded in its profile. Through XCF signaling it recognizes the other IP stacks in the sysplex.

With the VIPADISTRIBUTE statement in the TCP/IP profile, you code which Dynamic VIPA is assigned to the sysplex and which ports are served. An entry with the distributed IP address is added to the home list of all the IP stacks in the sysplex. The home list entry on the target stacks is actually done with a message that the distributing stack sends to all stacks read from the VIPADISTRIBUTE statement in the TCPIP profile.

With the DESTIP statement, the Dynamic XCF addresses that are candidates to receive new incoming requests are specified.

If the distributing stack receives a connection request, it queries WLM and/or QoS to select the best target stack and forwards the connection request to the chosen stack. Both involved stacks will create an entry for this connection in their connection routing table (CRT) and subsequent incoming data for this connection will be forwarded to the correct target stack.The VCRT can be displayed by using the NETSTAT VCRT command as shown in Figure 5-6.

```
D TCPIP,TCPIPC,N,VCRT,MAX=400
EZZ2500I NETSTAT CS V2R10 TCPIPC 050
DYNAMIC VIPA CONNECTION ROUTING TABLE:
DEST IPADDR      DPORT  SRC IPADDR       SPORT  DESTXCF ADDR
9.12.8.10        01501  9.24.104.75      03448  10.1.100.62
9.12.8.10        01501  9.24.104.75      03451  10.1.100.62
...
9.12.8.10        01501  9.24.104.75      03443  10.1.100.61
9.12.8.10        01501  9.24.104.75      03446  10.1.100.61
9.12.8.10        01501  9.24.104.75      03449  10.1.100.61
...
9.12.8.10        01501  9.24.104.75      03536  10.1.100.62
9.12.8.10        01501  9.24.104.75      03539  10.1.100.62
9.12.8.10        01501  9.24.104.75      03542  10.1.100.62
```

Figure 5-6 NETSTAT VCRT

As soon as the connection no longer exists, the entry is removed from the connection routing table of both stacks.

5.5 Sysplex Distributor functionality

Let us consider the scenario depicted in Figure 5-7 on page 96. This scenario includes four CS for z/OS IP stacks running in the same sysplex cluster in GOAL mode (WLM goal mode). All of them have SYSPLEXROUTING, DATAGRAMFWD, and DYNAMICXCF configured. Let us assume that:

- H1 is configured as the distributing IP stack with V1 as the Dynamic VIPA (DVIPA) assigned to the sysplex cluster.
- H2 is configured as backup for V1.
- H3 and H4 are configured as secondary backups for V1.

- APPL1 is running in all the hosts that are members of the same sysplex cluster. Note that the application could also be running in two or three of the hosts or in all of them at the same time.

With this in mind, here is how Sysplex Distributor works:

1. When IP stack H1 is activated, the definitions for the local XCF1 link are created dynamically due to DYNAMICXCF being coded in the H1 profile. Through XCF signalling, H1 recognizes the other IP stacks that belong to the same sysplex cluster and their XCF associated links: XCF2, XCF3, and XCF4.
2. The DVIPA assigned to the sysplex cluster and the application ports that this DVIPA serves are read from the VIPADISTRIBUTE statement in the profile data set. An entry in the home list is added with the distributed IP address in all the target IP stacks. The home list entry on the target stacks is actually done with a message that H1 sends to all the stacks read from the VIPADISTRIBUTE statement. Only one stack advertises the DVIPA through the RIP or OSPF routing protocol. In this case, it is the one that resides in H1, the host in charge of load distribution.
3. H1 monitors whether there is at least one application (APPL1 in Figure 5-11 on page 67) with a listening socket for the designated port and DVIPA. Actually H2, H3, and H4 will send a message to H1 when a server (in our case APPL1) is bound to either INADDR_ANY or specifically to the DVIPA (and, of course, the designated port). With that information, H1 builds a table with the name of the application and the IP stacks that could serve any connection request for it. The table matches the application server listening port with the target XCF IP address.
4. When a client in the network requests a service from APPL1, the DNS resolves the IP address for the application with the DVIPA address. This DNS could be any DNS in the IP network and does not need to register with WLM.
5. As soon as H1 receives the connection request (TCP segment with the SYN flag), it queries WLM and/or QoS to select the best target stack for APPL1 and forwards the SYN segment to the chosen target stack. In our example, it is APPL1 in H4 that best fits the request.
6. One entry is created in the connection routing table (CRT) in H1 for this new connection with XCF4 as the target IP address. H4 also adds the connection to its connection routing table.

   Note: If a program binds to DVIPA on H4 and initiates a connection, H4 needs to send a message to H1, so H1 can update its connection routing table accordingly. As an example, this is used when the FTP server on H4 would initiate an active data connection (from port 20) to a client.
7. The H1 IP stack will forward subsequent incoming data for this connection to the correct target stack.

8. When the H4 IP stack decides that the connection no longer exists, it informs the H1 IP stack with a message so H1 can remove the connection from its connection routing table.

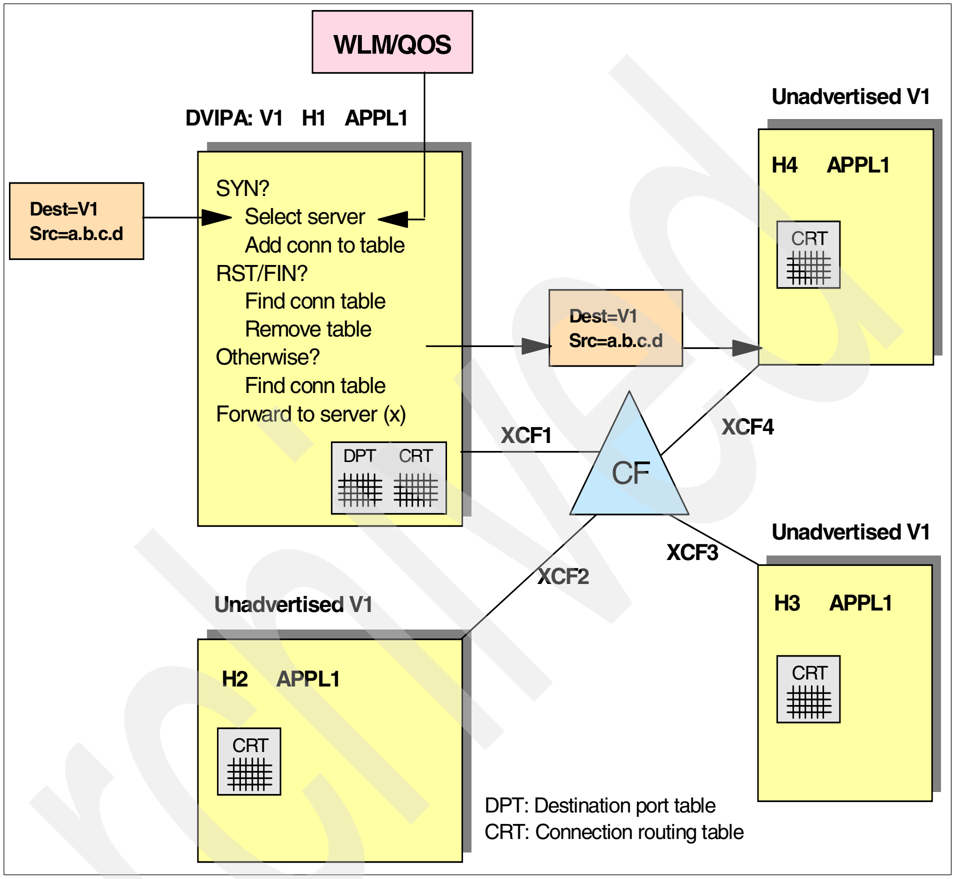


Figure 5-7 Sysplex Distributor example

5.6 Implementation

To implement the Sysplex Distributor, you first have to choose which IP stack is going to execute the Sysplex Distributor distributing function (the VIPA owning stack) and which IP stacks are going to be the backup stacks and in which order. You have to code IPCONFIG DATAGRAMFWD and DYNAMICXCF in the TCPIP.PROFILE data set in all the IP stacks of the sysplex cluster.

If you want to implement a WLM-based distribution, you have to register all IP stacks participating in the sysplex with WLM coding SYSPLEXROUTING in each IP stack. Also verify that all the participating IP images are configured for WLM GOAL mode. Figure 5-9 on page 98 shows an IPCONFIG definition.

If SYSPLEXROUTING is not coded in any IP stack, the distribution for incoming connections to the target applications will be random.

Note: Workload distribution through Sysplex Distributor is available only when the distributing and target stacks are at CS for OS/390 V2R10 or higher.

Code the VIPADYNAMIC/ENDVIPADYNAMIC block using the VIPADEFINE and VIPADISTRIBUTE statement for the distributing stack and for the backup stacks use the VIPABACKUP statement.

VIPA takeover and takeback is active when the VIPADEFINE statement in the TCP/IP profile data set is defined with the MOVE IMMED or MOVE WHENIDLE parameter. The difference between these two parameters is that when MOVE IMMED is defined, the takeback happens immediately after the original owning stack has recovered. With MOVE WHENIDLE coded, the takeback is delayed until all connections are finished on the backup stack. MOVE IMMED is recommended.

Figure 5-10 on page 99 shows a sample definition of a distributing stack where 9.12.8.10 is immediately moved back from the backup to the original owning stack after it has recovered. Depending on WLM, all incoming connection requests for 9.12.8.10 PORT 20, 21, 1023 and 1501 are routed to either 10.1.100.61 or 10.1.100.62. These are the Dynamic XCF addresses defined with the DESTIP statement.

You need at least one system in the sysplex which is defined to be the backup for the DVIPA to take it over in case the DVIPA owning stack fails. Figure 5-11 on page 99 shows the VIPADYNAMIC/ENDVIPADYNAMIC block of the backup IP stack.

5.7 Our test network

Figure 5-8 on page 98 shows the network in which we implemented our WebSphere MQ queue-sharing group.

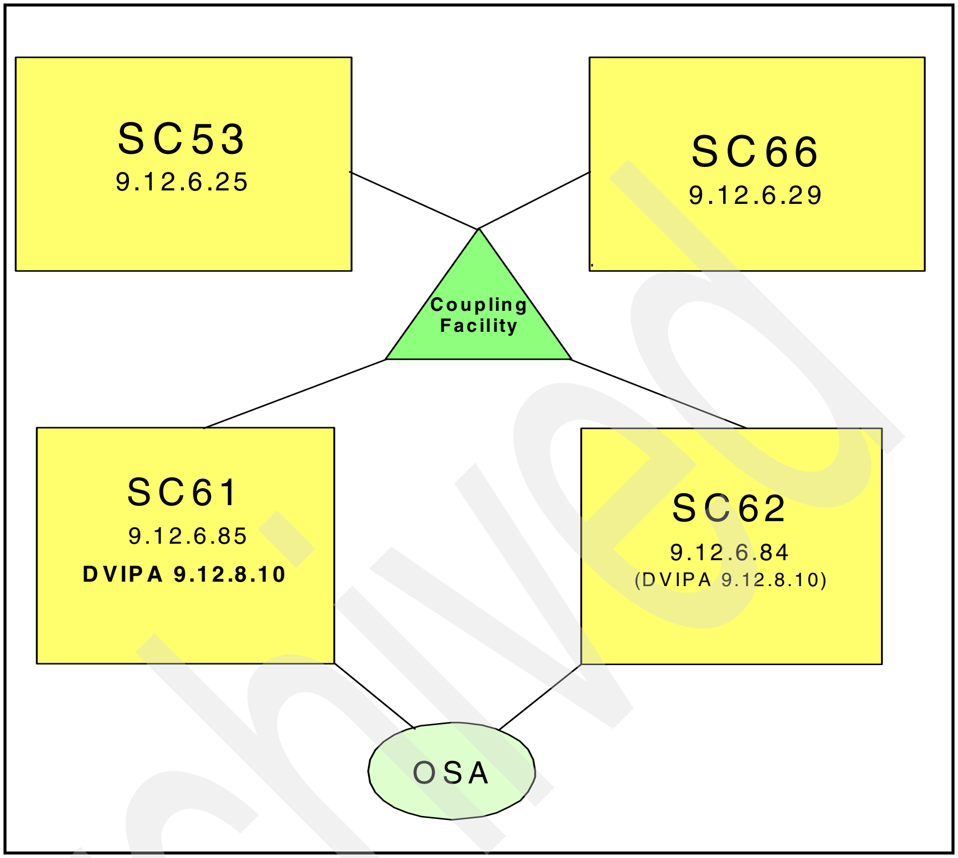


Figure 5-8 Our test network

SC61 is the DVIPA 9.12.8.10 owning LPAR. SC62 is the DVIPA backup.

On these two LPARs we defined our two gateway queue managers as described in Chapter 9, "Creating a shared queue environment" on page 159. Both LPARs have DATAGRAMFWD, DYNAMICXCF and SYSPLEXROUTING defined in their TCPIP.PROFILE dataset IPCONFIG parameter, as shown in Figure 5-9.

```
IPCONFIG
 DATAGRamfwd
  DYNAMICXCF 10.1.100.61 255.255.255.0 1
  SYSPLEXRouting
```

Figure 5-9 IPCONFIG

The VIPADYNAMIC/ENDVIPADYNAMIC block of SC61 contains the VIPADEFINE MOVE IMMEDIATE statement for the DVIPA address 9.12.10.8, the VIPADISTRIBUTE statement for port 1501, where the queue managers will listen on and the DESTIP statement containing the XCF addresses of SC61 and SC62 where the incoming requests shall be routed to. Figure 5-10 shows the configuration.

```
VIPADYNAMIC
  VIPADEFINE MOVE IMMEDIATE 255.255.255.0  9.12.8.10
  VIPADISTRIBUTE 9.12.8.10 PORT 20 21 1023 1501
    DESTIP 10.1.100.61 10.1.100.62
ENDVIPADYNAMIC
```

Figure 5-10   VIPADYNAMIC/ENDVIPADYNAMIC

SC62 is defined as the VIPABACKUP and therefore has the VIPABACKUP statement within the VIPADYNAMIC/ENDVIPADYNAMIC block.

```
VIPADYNAMIC
  VIPABACKUP 100 9.12.8.10
ENDVIPADYNAMIC
```

Figure 5-11   VIPABACKUP

As the routing application we were using OMPROUTE with OSPF (Open Shortest Path First) as the routing protocol. Figure 5-12 on page 100 shows the definition for it.

```
Area    area_number=0.0.0.0
             stub_Area=yes
             Authentication_type=none;
OSPF_Interface
  IP_Address=9.12.6.85
  NAME=OSA2860LNK
  Cost0=1
  Subnet_mask=255.255.254.0
  MTU=1500;
OSPF_Interface
  IP_Address=9.12.8.10
  NAME=DVIPA
  Subnet_mask=255.255.255.0
  Cost0=8
  MTU=32768;
OSPF_Interface
  IP_Address=10.1.100.61
  NAME=DXCF
  Subnet_mask=255.255.255.0
  Cost0=8
  MTU=32768;
AS_Boundary_routing
 Import_direct_routes=no
 Import_RIP_routes=YES
```

Figure 5-12 OMPROUTE definition

SC53 and SC66 are our back-end application LPARs and not target stacks of the Sysplex Distributor. On these LPARS, we defined two other queue managers and connected one CICS region to each of them to process the messages sent from outside the queue-sharing group to the shared queues. The definitions of the shared queues are held with the help of a DB2 data-sharing group in the coupling facilty and the messages which are sent to the shared queues are stored in the coupling facility structures which we defined for the queue-sharing group as described in Chapter 9, “Creating a shared queue environment” on page 159.

All incoming connection requests for 9.12.8.10 first go through an Open System Adapter (OSA). For our test installation it was sufficient to have only one OSA for our two gateway LPARs. If this would be a production environment we would have used two OSA adapters because if one fails our entire queue-sharing group would be isolated from the remote systems.

The OSA adapter will forward the requests to SC61, which is the owner of the DVIPA 9.12.8.10 and the Sysplex Distributor will distribute it to that IP stack which fits best, either to SC61 itself or to SC62.

Part 2

Advanced features

In Part 2 of this redbook, we introduce advanced features of WebSphere MQ for z/OS V5.3, namely the queue-sharing and clustering technologies.

6

WebSphere MQ for z/OS latest enhancements

In this chapter, we highlight the changes and enhancements in WebSphere MQ Version 5.3 as compared to the earlier Versions 2.1 and 5.2. We describe how WebSphere MQ enhancements exploit the features of a Parallel Sysplex environment.

We cover the following topics:

- Introduction to shared queues and queue-sharing groups
- Shared channels
- Shared queue support restrictions for WebSphere MQ Version 5.3
- Other new features with WebSphere MQ Version 5.3
- Increased similarity to WebSphere MQ products on distributed platforms

6.1 Shared queues and queue-sharing groups

Shared-queue support was introduced by MQSeries for OS/390 V5.2 and further improved by WebSphere MQ V5.3 primarily with the persistent shared queue definition. The benefits of using shared queues are availability for new messages, availability for old messages, scalable capacity, low-cost messaging within a Parallel Sysplex, and easier administration.

Another benefit is a capability to implement pull workload balancing. It means that by defining an application's input queue as a shared queue you make any message put to that queue available to be retrieved by any queue manager in the queue-sharing group. You can configure a queue manager on each z/OS image in the Parallel Sysplex and, by connecting them all to the same queue-sharing group, let any one of them access messages on the application's input queue.

6.1.1 Introduction of shared queue support

A shared queue is a type of local queue in which messages on that queue can be accessed by one or more queue managers that are identified to the sysplex. The queue managers that can access the same set of shared queues form a group called a queue-sharing group (QSG).

Queue-sharing groups

A QSG controls which queue managers can access which coupling facility list structures and hence which shared queues. Each queue manager can only belong to a single QSG. Each coupling facility list structure is owned by a QSG (through a naming rule) and can only be accessed by queue managers in that QSG.

The QSG provides a single system image for security administration. Distributed queuing exploits session balancing mechanisms to provide a single system image of the QSG for remote connection.

Multiple queue managers on multiple MVS images within the same queue-sharing group can MQPUT messages to and MQGET messages from the same shared queue. This is achieved by storing all the messages in a shared queue in the same coupling facility list structure.

Multiple queue managers on multiple MVS images within the same queue-sharing group can access the same WebSphere MQ objects. This is achieved by storing the object definitions in tables of a DB2 data-sharing group. You can modify the object definition using MQSC commands and have it used by all the queue managers in the queue-sharing group.

The queue managers can share the queues defined as shared within the same queue-sharing group. The queue managers also maintain their own logs and page sets to use non-shared local queues, and store definitions of private objects on page set zero. Messages that are put into shared queues are not stored on page sets. MQPUT and MQGET operations on persistent messages (on both shared and non-shared queues) are recorded on the queue manager log. This reduces the risk of data loss in the event of a coupling facility or page set failure.

Persistent shared queue messages

A huge advancement with WebSphere MQ Version 5.3 is the introduction of shared queues that can hold persistent messages. Previous versions only provided support for shared queues with non-persistent messages. As we discuss in Chapter 15, “Recovery scenarios” on page 295, persistence simplifies the restart and recovery procedures.

Advantages of using shared queues

The shared queue architecture in a Parallel Sysplex environment, where cloned images acquire messages from a single shared queue, has some very useful properties. It is scalable, by adding new instances of the server application, or even adding a new z/OS image with a queue manager (in the queue-sharing group) and a copy of the application. It is highly available because the failure of a single MVS image does not prevent access to shared queues. In addition, the system naturally performs "pull" workload balancing, based on the available processing capacity of each queue manager in the queue-sharing group.

6.2 Shared channels

The advantage of using shared channels is high availability as opposed to being connected to a single queue manager. There are two perspectives to look at channel agents as either inbound or outbound channels. Channels can be classed as private or shared. An inbound channel is classed as shared if it is connected to the queue manager through a group listener. A group listener is an additional task started on each channel initiator in the queue-sharing group. This task listens on an ip address/port combination specific to that queue manager known as its group address. Each group address may then be registered with a front-end routing mechanism such as VTAM Generic Resources, TCP/IP domain name systems (DNS), Network Dispatcher, or the Sysplex Distributor. The front-end mechanism is used to map a queue-sharing group-wide generic IP address/port to a specific group address.

An inbound channel routed to a group port can be used to put messages to any queue whether the target queue is shared or not. If the target queue is not a shared queue, the messages might be put on any queue in the queue-sharing group with that name. The name of the queue determines the function, regardless of the hosting queue manager.

An outbound channel is considered to be a shared channel if it is getting messages from a shared transmission queue. If it is shared, it holds synchronization information at the queue-sharing group level. This means that the channel can be restarted on a different queue manager and channel initiator instance within the queue-sharing group if the communications subsystem, channel initiator, or queue manager fails. Restarting failed channels in this way is a feature of shared channels called peer channel recovery. An outbound shared channel is eligible for starting on any channel initiator within the queue-sharing group, unless it was to be specified to be started on a particular channel initiator.

Improved channel availability

Channel availability has improved in WebSphere MQ Version 5.3 by the introduction of the channel TCP/IP KeepAlive attribute. Channel KeepAlive is the time interval after which TCP/IP checks the health of an individual channel's connection. This means that WebSphere MQ applications can process messages more quickly after a TCP/IP communications failure. This attribute is applicable to all channel types except client connection channels.

The channel availability was further enhanced in WebSphere Version 5.3 through batch heartbeat support. This adds an extra heartbeat before the sender channel sends the flow to the receiver channel to commit the message. Sending the extra flow reduces the risk of becoming stuck in-doubt because the partner went away after the sender went into in-doubt state, by checking just before going into in-doubt state, that the partner is still there.

6.3 Programming applications with shared queues

This section discusses some design considerations when developing applications that use message-sharing techniques. Not all applications lend themselves to effectively using shared queues. This section should help you decide when to use queue sharing.

Serializing your applications

Some applications must retrieve messages in the order they were written to the queue. An examples of this requirement is a queue containing messages generated from updates to a database image in which the sequence of the image updates must follow the order of the original database update sequence. This is often achieved in local queues by issuing an MQOPEN with the option of MQOO_INPUT_EXCLUSIVE. An application using a shared queue can perform the same MQOPEN, but to take advantage of the shared queue it should use MQCONNX to connect to the queue manager.

Considerations in using shared versus non-shared queues

In handling applications that access shared queues in WebSphere MQ Version 5.3 when migrating from earlier versions, you need to consider whether you will place persistent messages on those queues.

A shared queue and a private queue in the same queue-sharing group cannot share the same name. When an MQOPEN is issued, the application receives a MQRC_OBJECT_NOT_UNIQUE if accessing a duplicate name.

Does your application queue require a message greater than 63 KB? If so, then the queue used must not be shared. You may consider separating the message into smaller messages within the application code. Messages stored on a shared queue have a 63 KB limit.

Does the application retrieve messages with a specific MSGID/CORRELID? The application queue must be set with the proper index.

Do the applications deal with all return codes from an WebSphere MQ API or just decide whether there were any problems? Applications may need to be changed to handle extra MQRC return codes that have to do with shared queue definitions.

6.4 Shared queue support restrictions in Version 5.3

WebSphere MQ for z/OS Version 5.3 shared queue support has the following key restrictions:

- Maximum message size of 63 KB (64512 bytes). The maximum coupling facility list structure entry size is 64 KB, from which we must subtract some message control data. This 63 KB includes all WebSphere MQ message headers.
- Maximum of 512 shared queues per coupling facility list structure. This is an arbitrary implementation limit.
- Maximum of 512 coupling facility structures. This is an MVS limit on the total number of coupling facility structures of any type across an entire sysplex. All users of the coupling facility from all subsystems contribute to this limit.
- Maximum of 63 coupling facility structures per QSG. This limit is imposed by MVS on the number of coupling facility structures a queue manager can

connect to. The limit is 64 but we must subtract 1 for the administration structure.

- Maximum of 8,000,000 messages per coupling facility structure. This is because the coupling facility has a maximum of 2 GB of control storage, and each coupling facility list entry requires 256 bytes of this. The remaining message data is held in extended storage, which does not contribute to the 2 GB. Thus the 8,000,000 maximum is independent of message size.
- Maximum of 32 concurrently active queue managers per queue-sharing group (QSG). This is an MVS limit on the number of concurrent connections to a given coupling facility structure. All members of a QSG must connect to its coupling facility administration structure.

6.5 WebSphere MQ family consistency

In this section, we describe the key enhancements introduced in the latest releases of MQSeries for OS/390 and WebSphere MQ Version 5.3. not previously covered in this chapter.

A requirement for the new functions in WebSphere MQ for z/OS is to make the z/OS product more consistent with the products on the other platforms. This release of WebSphere MQ implements more of the functions already available on the distributed platforms.

6.5.1 Message size

One of these differences is the size of a message. For the major distributed platforms (Windows NT, OS/2, AIX, HP-UX and Solaris), the maximum size of a message was increased to 100 MB. This was introduced at Version 5.0 of the MQSeries product on these platforms. Large messages of up to 100 MB are now part of the WebSphere MQ product on z/OS. For Version 2.1 this function has been introduced with APAR PQ33000. For Version 5.2 and later versions, it is part of the base product.

6.5.2 Programming interfaces

The programming interfaces of WebSphere MQ on z/OS have been extended too. With Version 2.1, we now have a C++ object model that is consistent with the C++ object model on the other platforms. An even bigger enhancement is the addition of the Application Messaging Interface (AMI) on the OS/390 platform (introduced in MQSeries for OS/390 V5.2). AMI is a new programming interface that has been adopted by the Open Application Group as a standard. It is now available on all major WebSphere MQ platforms. Besides the fact that it is a standard API, the obvious advantage is improved programmer's productivity. A programmer needs no, or very little, knowledge of WebSphere MQ specifics. Instead a message is sent according to the specification of a policy. The policy determines the actual WebSphere MQ parameters to be used. Also, the AMI has specific methods for sending request-reply style of messages or publish-subscribe messages or send-and-forget messages. These different patterns of WebSphere MQ usage are now reflected in the interface.

Another addition to the programming interface on z/OS is the verb MQCONNX. This specialized call to connect to a queue manager was first introduced on the major distributed platforms in Version 5.0. It allowed trusted connections between an application and a queue manager. On distributed platforms, use of trusted connections provides a significant performance advantage. On z/OS, MQCONNX allows serialized applications within a sysplex.

Another change in the programming interfaces for z/OS was the addition of support for the unicode character sets UCS-2 and UTF-8. This means that a z/OS-based queue manager can convert unicode messages to an EBCDIC-based character set. Previously, when you wrote Java programs to use WebSphere MQ, you had to make sure that messages were converted on the sending platform before they were sent to the z/OS-based queue manager. Now, you do not have to worry any more about where to do the conversion of your message.

6.5.3 Dead-letter queue handler

Version 5.2 introduced a dead-letter queue handler. This tool was a popular way on the distributed platforms to handle messages on the dead-letter queue. The tool used a rules-based mechanism to decide what to do when a message arrived on the dead-letter queue. The action could be RETRY or DISCARD or IGNORE and the decision could be based on the reason why this message was inserted on the dead-letter queue. For example, a message could be inserted into the dead-letter queue because the target queue was full. Normally, this should be a temporary condition. The receiving program could have had some difficulties to keep up with the message arrival rate. Thus, an obvious action for the dead-letter queue handler is to retry the MQPUT on the destination queue. This useful tool has now been ported to z/OS platform.

6.5.4 Message grouping

WebSphere MQ Version 5.3 introduces a new type of indexed queue, which you use with an MQGET to group messages in a certain order.

Message grouping allows applications to receive a group of messages in a particular order regardless of the physical sequence of the messages. Applications can then process complete groups. This enables different application instances to work concurrently on different sets of related messages.

Message grouping lets you get messages in logical sequence, irrespective of their physical sequence. Each group message has a 24-byte group ID and a 4-byte message sequence number in the MQMD message header. All messages in the same group have the same group ID but different message sequence numbers. The last message in the group is indicated with a flag in the MQMD. Message grouping uses this to get the messages in logical sequence and to optionally wait for a complete group.

6.5.5 Browse with lock

Browse with lock permits you to get a message non-destructively from a single private queue or from a shared queue and lock the record with an MQGET. Each subsequent GET releases the current lock and locks the message under the updated browse cursor, if MQGET option MQGMO_LOCK is again used.

6.5.6 Still missing

Although many efforts have been done to close the gap of functionality between the z/OS platform and the major distributed platforms, it should be noted that there are still some differences. Message segmentation is one of the things that are still missing on z/OS, as is PCF messages. Message segmentation allows for automatic segmentation of big messages into a number of smaller messages. The receiving side can then decide to process each segment individually or the complete message.

Programmable Command Format (PCF) messages are messages that are used internally in WebSphere MQ. This format should be used when writing WebSphere MQ administration programs. It is also used for the management of clustered objects. On the distributed platforms, you can send a PCF message to the command server to manage or monitor the WebSphere MQ objects. On z/OS, the command server can only handle messages in the MQSC format.

6.5.7 Enhanced enterprise technology

The features listed below incorporate the enterprise technology into WebSphere MQ.

Dynamic change of queue manager parameters

You can now dynamically change some of the queue manager startup options while the queue manager is running. This eliminates the need to shut down and restart the queue manager to change system module parameters.

This version provides MQSC SET SYSTEM, SET ARCHIVE, and SET LOG commands. These commands can be used to change many of the CSQZPARM parameters while the queue manager is running.

Secure Sockets Layer (SSL) support

This feature provides out-of-the-box channel security, with protection against eavesdropping, tampering, and impersonation. It uses Public Key Infrastructure (PKI) to authenticate partners, agree, and then exchange a secret key for bulk data encryption. You can choose to authenticate only or authenticate and encrypt. You can specify a template distinguished name to control who can connect to a receiver channel.

Context profile

Context profiles can now be defined for each individual queue.

Transactional Java and JMS

Delivery of the MQSeries classes for Java and the MQSeries classes for Java Messaging Service (JMS) is a free, separately installable feature of the WebSphere MQ for z/OS product (previously available in SupportPac MA88). It also extends WebSphere MQ's use of RRS to support the Java transaction model, which allows both global and local UOWs.

Configuration change notification

The configuration change notification puts a change event message to a new system queue for each change to an object definition. You can dynamically enable and disable this capture and you can force event generation by object type, generic name, and time elapsed since last change.

Job log improvements

Another area where new function has been introduced to improve monitoring is the job log. Since the introduction of PQ28083 for MQSeries V2.1 and included in the base product of Versions 5.2 and 5.3, new messages are issued in the job log during the restart of the queue manager. These messages have been added to indicate the log range to be read for forward and backward recovery phases during restart and shutdown of the queue manager. Messages will also occur periodically during recovery to display the current log RBA being read as an indicator of the progress that recovery has made against the log range indicated. Messages have also been added at each checkpoint to show the media recovery

RBA and checkpoint RBA of each page set. While in the past you could think that your queue manager was in a loop, now you can see the progress that is being made and you can more or less calculate how much time is needed before the queue manager is restarted and available for your applications.

It is even more important to avoid long restarts. APAR PQ28093 for MQSeries V2.1 has included a new message in the job log to indicate the detection of a long-running unit of work. This functionality is also included in the base product of Versions 5.2 and 5.3. A unit of work is considered to be long running when the log records for the unit of work are about to be archived. For such a unit of work, if a rollback should be needed, archived log records will be read. The idea is that a unit of work should always be finished by the time the associated log records are archived. If that is not true, the administrator needs to analyze it and may force the unit of work to either rollback or commit, to avoid a time-consuming restart.

Old unit of work detection

Another improvement was introduced with APAR PQ27752 and is again part of the base product of Version 5.3. Now, the queue manager will prompt an operator when it detects an old unit of work that is in-flight or in-doubt. An old unit of work is defined as a unit of work for which the log records are already archived. Using WTOR, the queue manager asks whether it should commit or roll back the unit of work. Rollback is the normal behavior in this circumstance. However, the impact of a lengthy restart can be so huge for the normal operation of your applications that you may prefer to commit these outstanding units of work.

New utilities and utility functions provided

A new utility has been added, CSQJUFMT, which is an active log format utility. This utility is used to format active log data sets before they are used by a queue manager. If the active log data sets are formatted by the utility, log write performance is improved on the queue manager's first pass through the active logs.

A new page set management function, PAGEINFO, has been added to the CSQUTIL utility to extract the page set recovery RBA from a page set.

There are new options for the FORMAT function of the CSQUTIL utility.

The CSQ1LOGP log print utility, CSQJU003 change log inventory utility, and CSQJU004 print log map utility, have been extended to support logical record sequence numbers (LRSNs).

A new utility, CSQ5PQSG, is introduced to support management of queue-sharing groups.

See the *WebSphere MQ for z/OS System Administration Guide,* SC34-6053, for details of these utilities.

7

Using WebSphere MQ clustering technology

In this chapter, we describe WebSphere MQ clustering technology and explore how it can be used to implement a highly available WebSphere MQ infrastructure. We show that it helps to improve the availability of the system by providing automatic workload distribution across multiple target queue managers. It also simplifies the systems administration overhead by dramatically reducing the number of explicit object definitions required.

In the last section of this chapter, we show how the workload distribution mechanism can be enhanced by the use of cluster workload exits.

7.1 Overview of the clustering technology

The clustering technology was introduced with Version 2.1 of MQSeries for OS/390. This chapter introduces the concepts of queue manager clusters and explains the new terminology. We also compare the clustered environment with the conventional distributed queuing environment.

7.1.1 Concepts and terminology

A WebSphere MQ cluster is a group of WebSphere MQ queue managers that are logically related to each other. They typically serve a common application. The queue managers can be running at different locations and on different platforms. For example a financial institution may have distributed WebSphere MQ queue managers residing in each branch (W2000) and a number of host queue managers running on z/OS. All these queue managers could be logically configured as members of a single cluster. There is no real limit on the number of queue managers you can add to a single cluster. The only restraint may be the underlying networks capacity to support a potentially high number of concurrent connections.

A feature of WebSphere MQ clustering is the ability for a queue manager to host and discover queues. A cluster queue manager can host queues, which it advertises to other members of the cluster. The first time an application on another queue manager attempts to open one of these advertised queues, the queue is discovered and defined dynamically as well as a channel to the hosting queue manager. This cluster queue definition is functionally equivalent to a remote queue definition in the distributed queuing environment. No manual WebSphere MQ configuration tasks are required, which makes a clustered environment extremely flexible. Queues can be advertised and unadvertised and the required WebSphere MQ definitions are created/destroyed as required. Each queue manager normally will have a specific role as either a server/hosting queue manager or a client queue manager. The servers host the queues that are advertised to other members of the cluster and run the applications that process these messages. They will also generate replies back to the client. The clients put requests to the server's queues and may receive back replies. The queue managers in a cluster normally communicate directly with each other through the automatically defined cluster channels.

All the information required to enable this functionality is held by the *cluster repository* queue managers.

The role of the cluster repository queue managers

A queue manager becomes a member of a cluster by connecting to a *repository queue manager.* At the cluster design stage, several queue managers (usually two for redundancy) are designated to be the *full repository* queue managers. These queue managers are responsible for storing information about the names and network connection details of all queue managers in the cluster as well as the queues they host. To join a queue manager to the cluster you need to define a *cluster sender channel* pointing to the full repository queue manager and a *cluster receiver channel* to specify connection details. A queue manager that has joined a cluster will inform the full repository queue manager about itself and will be given details about all the other full repositories that are available to it. This information is supplied so that an alternate repository can be contacted if necessary.

> **Note**: The full repository is critical to the operation of a WebSphere MQ cluster. If a queue manager makes a cluster-related change, an update request will be sent to two full repositories (if defined). This is done in case one of the full repositories is unavailable, to ensure that all members of the cluster receive these updates in a timely fashion.

The full repositories keep themselves in synch by exchanging update requests between themselves. If any duplicate requests are found, they will be discarded. Information about non-repository queue managers and hosted queues will be learned from the repository queue manager as required.

In addition to the full repository queue manager, every other queue manager in the cluster maintains a *partial repository* of information about the cluster. Most queue managers in the cluster will only maintain information in their *partial repository* about objects for which they have a certain interest. If queue manager MQV1 connects to queue manager MQV2 to send a message to cluster queue CL.QUEUE2, then the repository of queue manager MQV1 will contain information on how to reach queue manager MQV2 (for example, channel definition parameters) and information about that specific queue. If queue manager MQV2 hosts cluster queue CL.QUEUE1 and that queue has not been accessed by MQV1, then there will not be any information about that queue in the repository of queue manager MQV1. This is shown in Figure 7-1 on page 118. Over time, a queue manager's partial repository will build up as new cluster objects are discovered. The full repository queue manager will be interrogated each time new information is required.

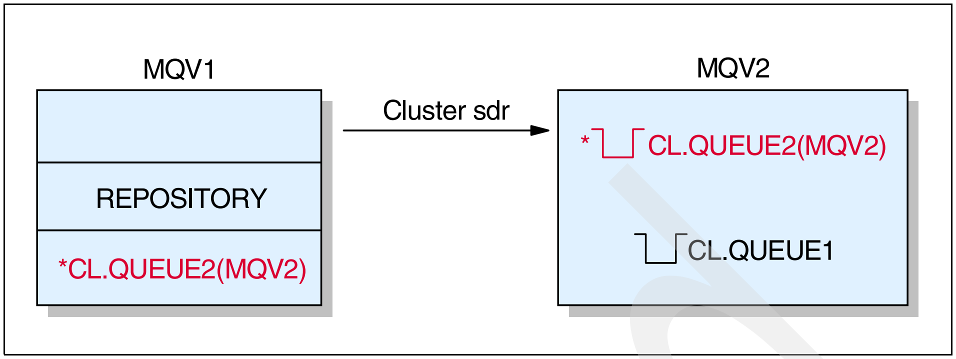


Figure 7-1 Partial repository

A cluster can work adequately with only one full repository, but having two is desirable for high availability. The full repository queue managers should also be located on a reliable and well-managed platform. z/OS should be chosen over a Windows or UNIX platform where possible. If both full repository queue managers become unavailable, all the other queue managers can still exchange messages with each other based on the information that they have already built up in their partial repositories. There will be an impact if new cluster objects need to be discovered or updates to hosted queues need to be propagated across the cluster. In this instance, the other queue managers will not be notified of new objects or updates until one of the full repositories is reinstated.

Keeping all the cluster queue manager up to date with the latest changes to cluster objects is another important responsibility of the full repository queue manager. When a queue manager obtains information about a cluster object from the full repository, it also registers an interest in that object. The full repository maintains a list of interested parties (or subscribers) for a certain object. Whenever there is a change to this object, the full repository queue manager publishes the change to all subscribed queue managers. This publish/subscribe mechanism ensures that all members of the cluster hold the latest information relating to cluster objects. For example, if a member of the cluster stopped hosting a particular queue CL.QUEUE1, all queue managers that have an interest in that queue will be notified by the full repository that the queue has been unadvertised. The queue entry will be removed from all the partial repositories and the queue will no longer be visible within the cluster.

All information in a repository is stored as a number of persistent messages on a system queue SYSTEM.CLUSTER.REPOSITORY.QUEUE. The number of messages in this queue does not reflect the number of objects in the cluster. The number of messages in this queue can actually decrease over time without deleting any cluster objects. This occurs because the repository manager periodically performs a reorganization of this queue.

New cluster objects

MQSeries clustering technology has also introduced some new channel types, *cluster sender* and *cluster receiver*, which are used to enable communication between members of the cluster. Although the names are new, the underlying message channel agent protocols used are the same as in the conventional distributed queuing environment. A queue manager is automatically joined to the cluster when these channels are defined.

A cluster sender channel is defined to point to one of the full repository queue managers. It is not important which full repository queue manager you connect to. They are all equivalent. As with distributed queuing, the name of the cluster sender channel has to match the name of the cluster receiver channel on the full repository queue manager. This cluster sender channel will be used to send cluster information about the queue manager to the full repository queue manager. Refer to 7.2, "Setup of a WebSphere MQ cluster" on page 126 for the detailed steps required to build a simple cluster.

A *cluster receiver* channel is then defined. The fields in this definition are used as a template, which other cluster members use to automatically define a cluster sender channel back to this queue manager. Since this definition of the cluster receiver is used in the creation of a sender type channel, there are fields specified here that would normally be seen only in a sender/server channel definition, for example Disconnect Interval, Connection Name, Short Retry Count. These fields are defined so that all inbound channel connections to this queue manager are consistent and use channel parameters controlled by the target queue manager. This information will be sent to the full repository queue manager as soon as both cluster sender and cluster receiver channels have been defined. If an explicit cluster sender has parameter values specified that conflict with the repository, then the specified parameter values will be merged with values in the repository to enforce consistency.

Three new system queue objects have also been introduced to support clustering. The queue SYSTEM.CLUSTER.TRANSMIT.QUEUE is used to hold all messages that are ready to be sent to any queue manager in the cluster. Instead of defining a transmission queue for each sender channel, you have now a single cluster transmission queue instead. This reduces the number of MQSeries objects required in a clustered environment. For details refer to 7.1.2, "Immediate benefits of using clustering" on page 120.

The queue SYSTEM.CLUSTER.COMMAND.QUEUE is defined on each queue manager and is used to exchange repository information. The command messages placed on this queue are generated and processed by a component of the queue manager called the repository manager.

The queue SYSTEM.CLUSTER.REPOSITORY.QUEUE holds the repository information as a number of persistent messages.

7.1.2 Immediate benefits of using clustering

WebSphere MQ clustering technology provides two major benefits. They are reduced system administration and workload balancing.

Reduced system administration

Even a small cluster will benefit from reduced system administration. Figure 7-2 shows a group of four queue managers.

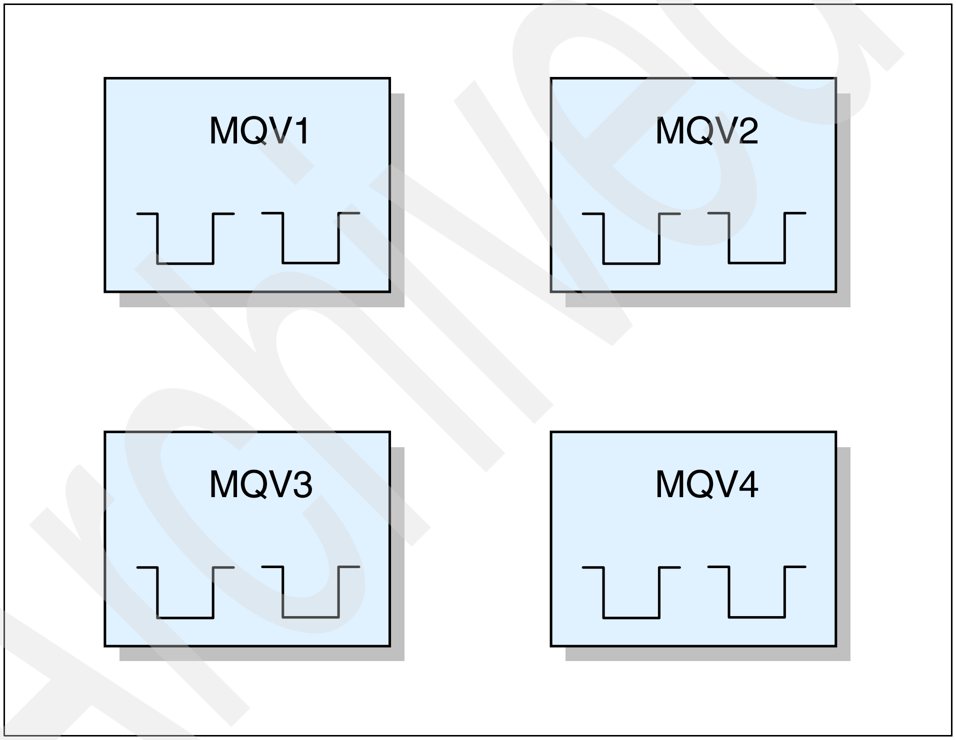


Figure 7-2 A network of four queue managers

Joining these queue managers into a network will require fewer definitions if a clustered configuration is chosen compared with a standard distributed queuing configuration.

Table 7-1 on page 121 shows the number of definitions required to set up a network of four queue managers using distributed queuing.

Table 7-1 Definitions for distributed queuing

Description	Number per Queue Manager	Total Number
Sender channel definitions to every other queue manager	3	12
Receiver channel definitions for every other queue manager	3 (1 if generic receiver definitions are used)	12 (4 if generic receiver definitions are used)
Transmission queue to other queue managers	3	12
Local queue definitions for each local queue	2	8
Remote queue definitions for each remote queue this queue manager wants to put messages to	6	24
Optional. Process definition to trigger channels	3	12

The number of definitions could possibly be reduced by the use of generic receiver channels. However, there could be as many as 20 for each queue manager and 80 for the whole network.

Table 7-2 shows the number of definitions required to set up a network of four queue managers using clustering.

Table 7-2 Cluster definitions

Description	Number per queue manager	Total Number
Cluster sender channel to send messages to a full repository	1	4
Cluster receiver channel to receive messages from other members of the cluster	1	4
Local queue definition for each local queue	2	8

Creating the network of four queue managers using clustering only requires four definitions per queue manager, or a total of 16. This is a significant reduction when you compare the number of definitions required using distributed queuing (80 definitions). Fewer definitions will result in fewer errors during the configuration process. There are also administrative advantages. If a queue is removed from a cluster queue manager, no system management is required on the other members of the cluster. The full repository queue manager will ensure that all relevant members are advised by a repository update message. Any queues that have been previously discovered will be removed from the local repositories.

Workload management

The cluster workload management functionality allows a queue manager to distribute client requests across multiple server queue managers. For this to occur, there must be multiple instances of a particular server queue defined within the cluster. Here are two reasons for doing this:

- To increase the capacity available to process requests.
- To address high availability concerns by introducing an ability to fail over work from one server to another.

For example, in a Parallel Sysplex, data can be stored in shared DB2 tables. This will allow the MQSeries applications to be cloned on each system and have full access to the same data. In this scenario, the failure of one queue manager can be tolerated because the workload will be distributed to other available members of the cluster for processing.

> **Note:** The applications must be designed with workload management in mind. If there is any affinity with a particular server, then this functionality would not be appropriate. Affinities will be discussed later.

There is a built-in workload management algorithm that will select a particular instance of the application queue. If there are several choices and all the destinations are available, a round-robin approach is used.

> **Note:** A local copy of the queue will always take preference over a cluster queue.

If round-robin workload distribution is not adequate, a cluster workload exit can be implemented to make more intelligent routing decisions. Section 7.3, “Workload balancing exits” on page 131 describes a cluster workload exit which is available from the MQSeries support pack Web site. This site can be found at:

`http://www-3.ibm.com/software/ts/mqseries/txppacs/txpm1.htm`

Table 7-3 on page 123 shows a simple cluster where the default round-robin approach is used. In this example, WebSphere MQ request messages are being distributed to three cloned server applications.

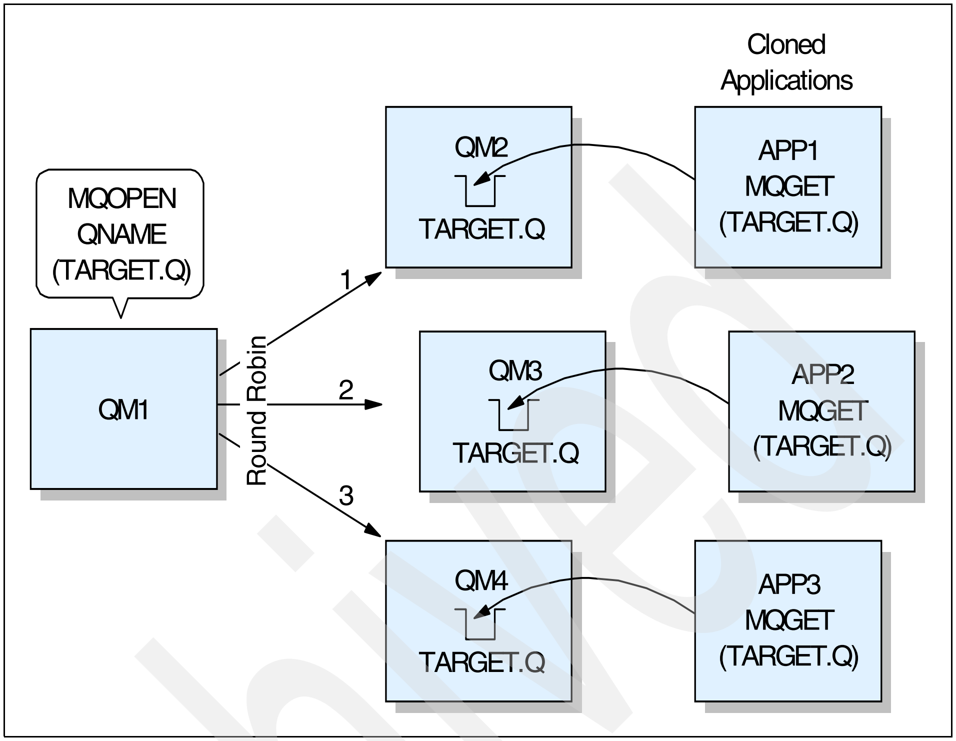


Figure 7-3 Workload balancing - Multiple instances of target queue

If a message is on the SYSTEM.CLUSTER.TRANSMIT.QUEUE destined for a particular queue manager, and that queue manager or the channel to that queue manager fails, the queue manager will call the workload management (or default) exit again to reroute the message to another target queue manager hosting that queue. The only time this will not occur is if the application has decided to override workload distribution because of an affinity with a particular queue manager. In this case, the message will wait on the SYSTEM.CLUSTER.TRANSMIT.QUEUE until the queue manager is available again. In-doubt messages are nor rerouted either.

Application considerations with clustering

In a clustered environment, all applications have to be examined to see if they require their messages to be processed by a particular queue manager, or in a particular sequence. If these dependencies are found, then these applications are said to have an *affinity* with a particular queue manager.

For example, two applications may rely on a series of messages flowing between them in the form of questions and answers. If it is important that all the questions are sent to the same queue manager and all the answers are sent back to the other queue manager, then these applications have affinities with their respective queue managers.

Applications with message affinities can prevent the cluster workload management routines from making the best choice of queue manager.

There are several options to remove application affinities:

- Define a single instance of the queue. This will ensure that all messages will be routed to a single copy of the application. The disadvantage is that the application will not be scalable and there is no redundancy if that application fails.
- Modify the application. It may be possible to redesign the application so it can be run in parallel. It will then be able to take advantage of the scalability and redundancy that cluster workload balancing provides. This will most likely involve relocation of data to make it shareable, for example to shared VSAM files or shared DB2 tables.
- Replication of data. If moving to a data-sharing environment is not possible, an option may be to replicate read-only data on several systems in the sysplex. This approach will only benefit read-only applications.
- Maintain state data in the messages and store this data in non-volatile storage accessible to any copy of the MQSeries server application.

If removing the affinity is not a viable option, a new open option on the MQOPEN call may be useful. This option can be useful if message sequencing is the main concern with clustering. By specifying MQOO_BIND_ON_OPEN on the MQOPEN call, all MQPUTs for messages related to that particular open will be sent to the same destination. For a subsequent MQOPEN call, the cluster workload management routines will select another destination. All MQPUTs related to the new MQOPEN will be sent to the new destination. The use of the bind option is a trade-off for those situations where it is not practical or cost-effective to remove a queue manager affinity from an application. At least a small degree of workload balancing can still occur for each MQOPEN call rather than none at all. The use of the MQOO_BIND_ON_OPEN is shown in Figure 7-4 on page 125.

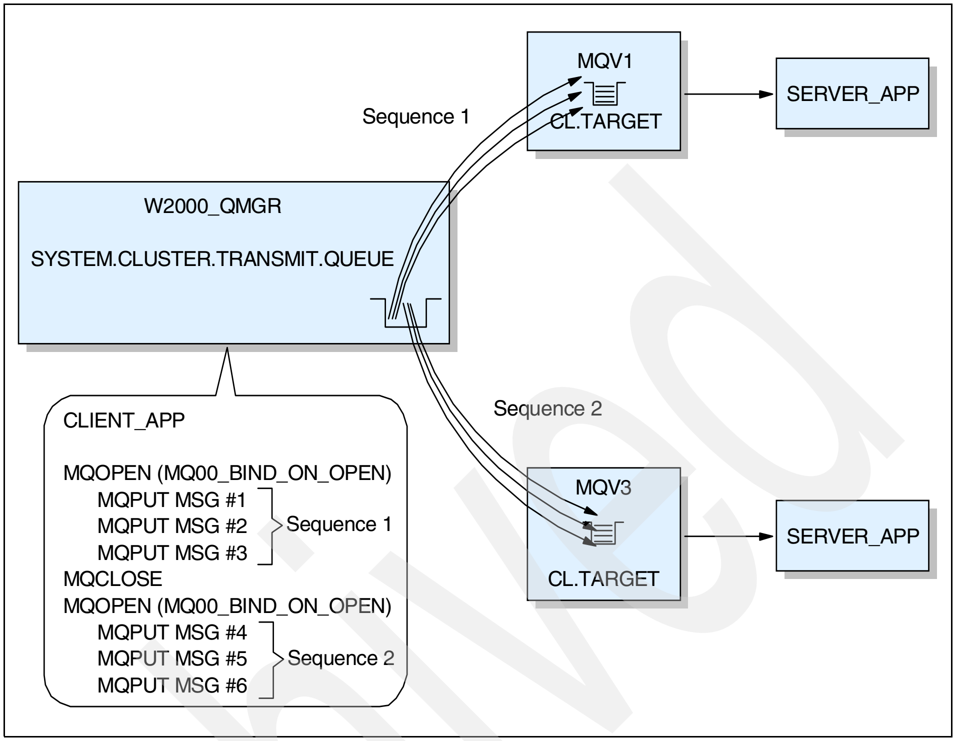


Figure 7-4 Effects of MQOO_BIND_ON_OPEN option on MQOPEN

Application CLIENT_APP is connected to queue manager W2000_QMGR. It needs to send two sets of message and have each set processed sequentially by the server application SERVER_APP.

The first MQOPEN specifying (MQOO_BIND_ON OPEN) will fix the target queue until the queue is closed and re-opened. This series of messages (MSG#1, MSG#2 and MSG#3) will all be sent to the queue manager MQV1 for processing by the local copy of the SERVER_APP application.

The target queue is then closed and reopened prior to sending the next set of messages. The cluster workload algorithm will now select queue manager MQV3 and the series of messages (MSG#4, MSG#5 and MSG#6) will all be processed by the cloned SERVER_APP application.

7.2 Setup of a WebSphere MQ cluster

In this section we describe a setup of a cluster called CLUSPLX1 consisting of four WebSphere MQ for z/OS 5.3 queue managers and one WebSphere MQ for Windows 2000 5.3 queue manager. Two of the z/OS queue managers will be the full repository queue managers and take on a gateway role for the client queue manager (Windows 2000). The other two z/OS queue managers will be the server (or application) queue managers. Work requests will come from the client and be routed to the servers using cluster workload management. For this example, the default round-robin algorithm will be used. This configuration will be enhanced when high availability configurations are discussed in Chapter 11, “High availability setup” on page 209. The application that uses WebSphere MQ in this environment is considered to have no affinities. Each message is independent from another and the processing of a message is the same on each server queue manager. Note that this implies that DB2 data sharing is being used in a Parallel Sysplex.

7.2.1 Configuration of the z/OS queue managers

Figure 7-5 on page 127 shows five independent queue managers prior to establishing the cluster.

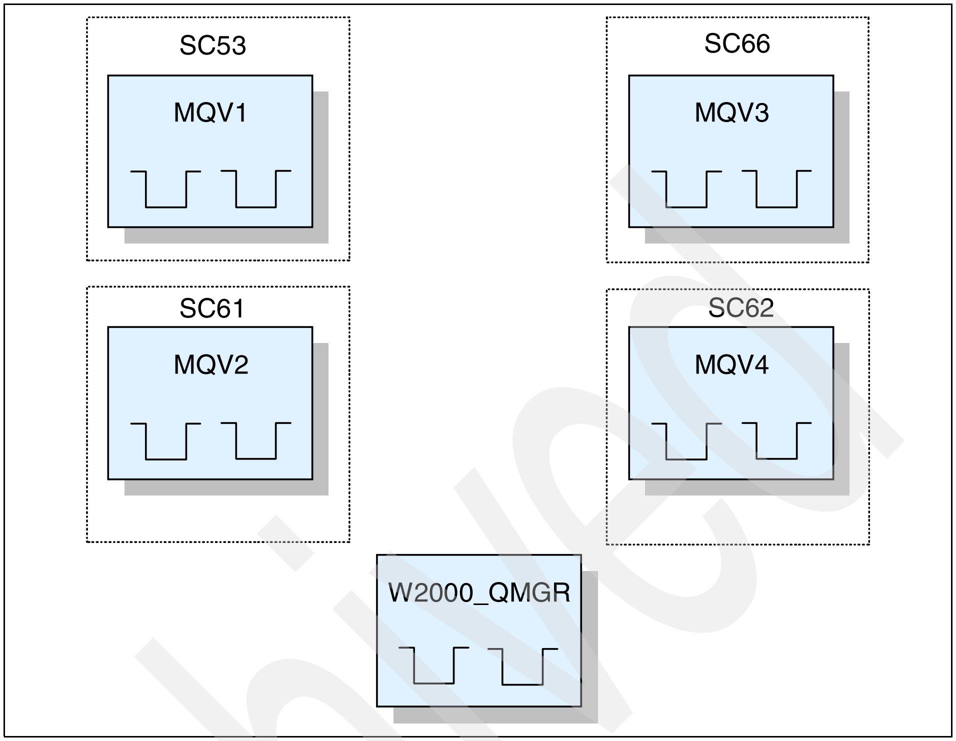


Figure 7-5 Independent queue managers prior to being clustered

Step 1 is to select which queue managers are to host the full repositories. In this example, we are using MQV2 and MQV4, both running on z/OS. This will provide a resilient platform and redundancy. The commands required to do this are shown in Example 7-1 through Example 7-5 on page 129.

Example 7-1 Define full repository queue managers

```
ALTER QMGR REPOS(CLUSPLX1)     on MQV2
ALTER QMGR REPOS(CLUSPLX1)     on MQV4
```

Step 2 is to define cluster sender and cluster receiver channels between the two repository queue managers MQV2 and MQV4. The `CONNAME` parameter in Example 7-2 points to the IP address of the target repository.

Example 7-2 Add repository queue managers to the cluster

```
On MQV2:

DEFINE CHANNEL(TO.MQV2) CHLTYPE(CLUSRCVR) +
   CONNAME('wtsc61.itso.ibm.com(1500)') +
   TRPTYPE(TCP) +
```

```
   CLUSTER(CLUSPLX1)

DEFINE CHANNEL(TO.MQV4) CHLTYPE(CLUSSDR) +
   CONNAME('wtsc62.itso.ibm.com(1500)') +
   TRPTYPE(TCP) +
   CLUSTER(CLUSPLX1)

On MQV4:

DEFINE CHANNEL(TO.MQV4) CHLTYPE(CLUSRCVR) +
   CONNAME('wtsc62.itso.ibm.com(1500)') +
   TRPTYPE(TCP) +
   CLUSTER(CLUSPLX1)

DEFINE CHANNEL(TO.MQV2) CHLTYPE(CLUSSDR) +
   CONNAME('wtsc61.itso.ibm.com(1500)') +
   TRPTYPE(TCP) +
   CLUSTER(CLUSPLX1)
```

Step 3 is to add the remaining queue managers MQV1 and MQV3 to the cluster.

Example 7-3 Join MQV1 and MQV3 to the cluster

```
On MQV1:

DEFINE CHANNEL(TO.MQV1) CHLTYPE(CLUSRCVR) +
   CONNAME('wtsc53.itso.ibm.com(1500)') +
   TRPTYPE(TCP) +
   CLUSTER(CLUSPLX1)

DEFINE CHANNEL(TO.MQV4) CHLTYPE(CLUSSDR) +
   CONNAME('wtsc62.itso.ibm.com(1500)') +
   TRPTYPE(TCP) +
   CLUSTER(CLUSPLX1)

On MQV3:

DEFINE CHANNEL(TO.MQV3) CHLTYPE(CLUSRCVR) +
   CONNAME('wtsc66.itso.ibm.com(1500)') +
   TRPTYPE(TCP) +
   CLUSTER(CLUSPLX1)

DEFINE CHANNEL(TO.MQV2) CHLTYPE(CLUSSDR) +
   CONNAME('wtsc61.itso.ibm.com(1500)') +
   TRPTYPE(TCP) +
   CLUSTER(CLUSPLX1)
```

Step 4 is to add the Windows 2000 client queue manager to the cluster.

Example 7-4 Join W2000_QMGR to the cluster

```
On W2000_QMGR:

DEFINE CHANNEL(TO.W2000_QMGR) CHLTYPE(CLUSRCVR) +
   CONNAME('9.24.104.132(1414)') +
   TRPTYPE(TCP) +
   CLUSTER(CLUSPLX1)

DEFINE CHANNEL(TO.MQV4) CHLTYPE(CLUSSDR) +
   CONNAME('wtsc62.itso.ibm.com(1500)') +
   TRPTYPE(TCP) +
   CLUSTER(CLUSPLX1)
```

Step 5 is to define cloned application queues on the server queue managers and advertise them to the cluster.

Example 7-5 Define clustered application queues

```
On MQV1:
DEFINE QLOCAL(TARGET.Q) CLUSTER(CLUSPLX1)

On MQV3:
DEFINE QLOCAL(TARGET.Q) CLUSTER(CLUSPLX1)
```

Figure 7-6 on page 130 shows the five queue managers in the cluster CLUSPLX1.

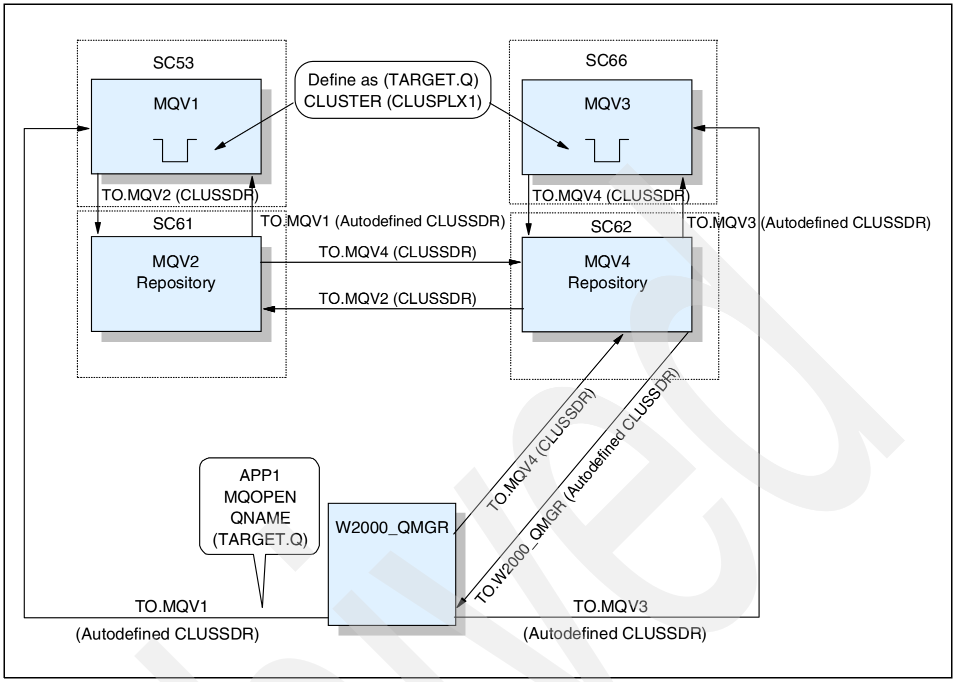


Figure 7-6 The final cluster CLUSPLX1

With this configuration, the client queue manager W2000_QMGR can send requests to both instances of queue TARGET.Q on the server queue managers MQV1 and MQV3. The channels connecting the client queue manager with the servers are defined automatically using data obtained from the full repository. These channels are defined when the first queue hosted by MQV1 or MQV3 is discovered on W2000_QMGR. Additional queues will be discovered on first use but they will be accessed using the same cluster channels.

7.2.2 Extending the cluster with additional queue managers

Additional queue managers can be easily added to the cluster by defining cluster receiver and cluster sender channels. Example 7.6 shows the commands required to add queue manager W2000_QMGR2 to the cluster CLUSPLX1.

Example 7-6 Adding an additional queue manager to the cluster

```
On W2000_QMGR2

DEFINE CHANNEL(TO.W2000_QMGR2) CHLTYPE(CLUSRCVR) +
   CONNAME('9.24.104.133(1414)') +
```

```
   TRPTYPE(TCP) +
   CLUSTER(CLUSPLX1)

DEFINE CHANNEL(TO.MQV4) CHLTYPE(CLUSSDR) +
   CONNAME('wtsc62.itso.ibm.com(1500)') +
   TRPTYPE(TCP) +
   CLUSTER(CLUSPLX1)
```

These commands can be modified for any new queue manager that needs to be added to the cluster.

7.3 Workload balancing exits

For the majority of applications, the default round-robin approach to workload distribution will be sufficient. For applications with different requirements, there are cluster workload balancing exits available to allow more intelligent routing decisions to be made. These exits can exploit any knowledge that may be available relating to the current performance of a particular target server. In this scenario, the workload can be directed to the servers with the most capacity to process a message at a particular point in time. A cluster exit, on entry, will be supplied with a list of all possible destinations. Using data from WebSphere MQ and additional performance data, the message can be routed to the most suitable target queue manager. Details on how to code a cluster workload exit can be found in *WebSphere MQ Queue Manager Clusters,* SC34-5349-04.

7.3.1 Example of a prioritized cluster workload exit

There is a WebSphere MQ SupportPac available (MC76) which provides a cluster workload balancing exit that uses the *network priority* (NETPRTY) parameter on a cluster receiver channel to help decide where to route a message. Normally, the NETPRTY field is only relevant when there are multiple cluster receiver channels to the *same* queue manager and is ignored between channels to *different* queue managers. If there are multiple queues with the same name on different queue managers, then this exit will choose the hosting queue manager that has the highest NETPRTY value on its cluster receiver channel. If multiple channels exist with the same value, then a round-robin approach is adopted as before. All other options such as *bind on open* are still honored.

Figure 7-7 on page 132 shows a simple cluster with a client queue manager and three target queue managers. All the cluster receiver channels have the same NETPRTY of 5. In this configuration, the cluster workload exit will distribute the messages in round-robin order the same as the default algorithm.

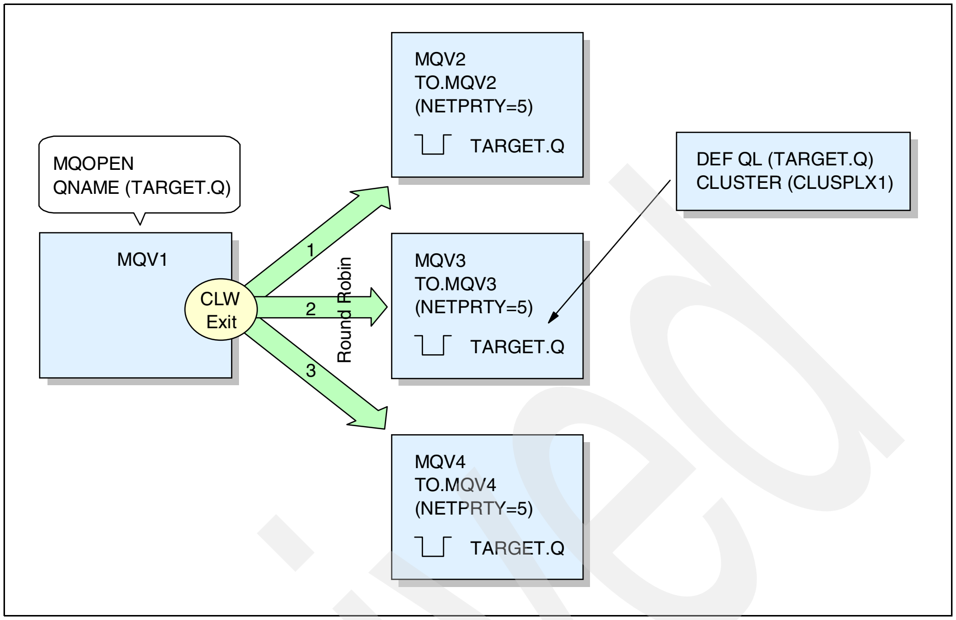


Figure 7-7 Prioritized cluster workload exit (NETPRTY equal)

If queue manager MQV3 needs to be excluded for any reason, the command shown in Example 7-7 can be entered to lower the NETPRTY of its cluster receiver channel.

Example 7-7 Lowering the NETPRTY of a cluster receiver channel

```
On MQV3
ALTER CHANNEL(TO.MQV3) CHLTYPE(CLUSRCVR) NETPRTY(4)
```

Figure 7-8 on page 133 shows how the cluster workload exit will react by excluding MQV3 from its round-robin list, assuming that MQV2 and MQV4 are still available.

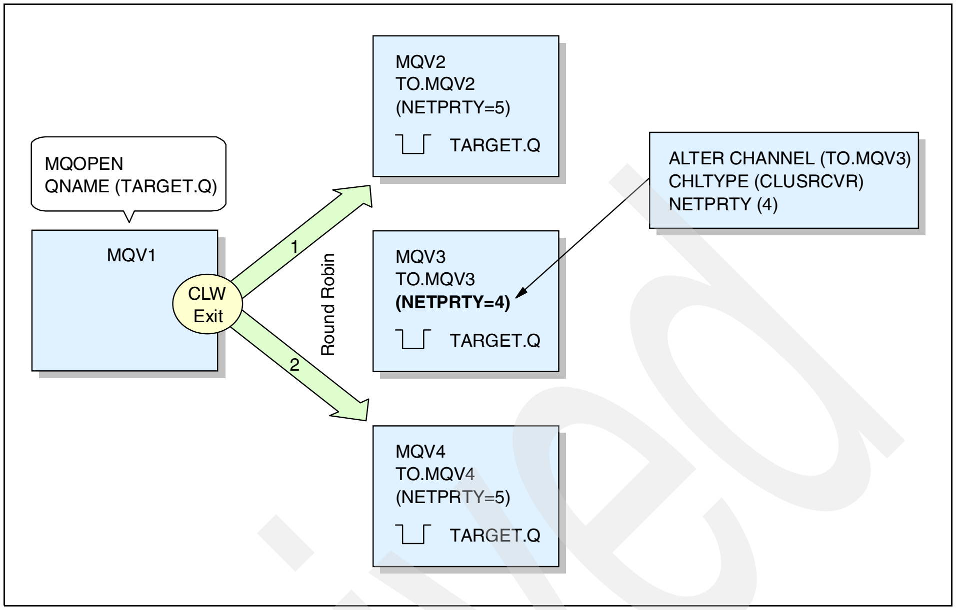


Figure 7-8 Prioritized cluster workload exit (NETPRTY not equal)

This sample exit could also be used to provide automatic failover from an active queue manager to a standby. Figure 7-9 on page 134 shows an application sending to one instance of a queue TARGET.Q on queue manager MQV2. The queue is also hosted on MQV3. However, the NETPRTY value of the cluster receiver channel TO.MQV3 is lower than the cluster receiver on MQV2. As a result of this difference, the cluster workload exit ignores MQV3 and distributes all messages to MQV2.

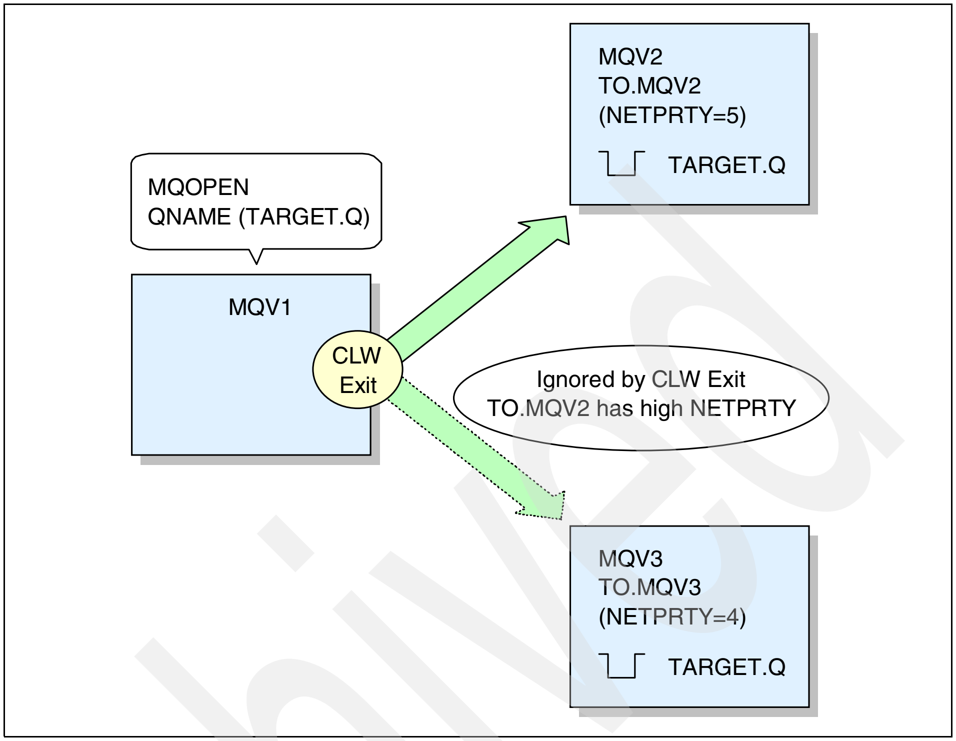


Figure 7-9 Prioritized cluster workload exit (before failover)

If the queue manager MQV2 fails, the workload will automatically be diverted to the instance of TARGET.Q on MQV3, because the cluster receiver channel with the NETPRTY of 4 is now the highest. Figure 7-10 on page 135 shows the behavior of the exit after the failure of queue manager MQV2.

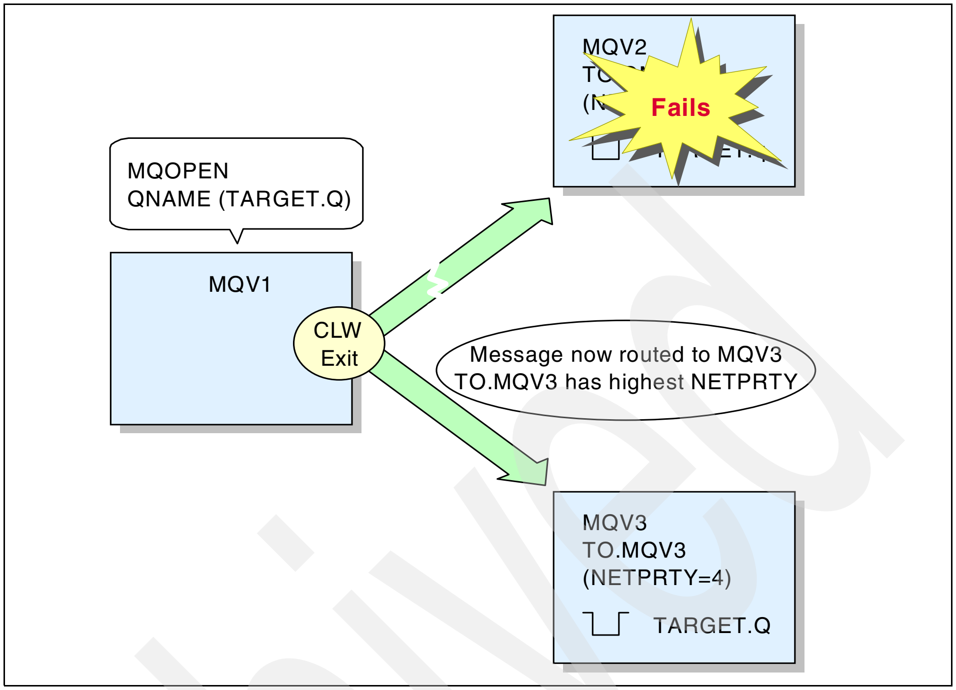


Figure 7-10 Prioritized cluster workload exit (after failover)

This exit can be downloaded from the WebSphere MQ SupportPac Web site at the following address:

http://www-3.ibm.com/software/ts/mqseries/txppacs/mc76.html

7.3.2 Summary

The clustering functionality of WebSphere MQ can provide a highly available environment for your applications by using workload balancing. If messages are less than 63 KB, then workload balancing can also be achieved in the context of a Parallel Sysplex environment by the use of shared queues (which reside in a coupling facility) and queue-sharing groups. This is described in Chapter 8, “Shared queue overview” on page 137. In an environment where large messages (more than 63 KB) are being transferred, a combination of shared queues and clustering can provide an extremely reliable environment for WebSphere MQ applications. Refer to Chapter 11, “High availability setup” on page 209 for a description of these high availability configurations.

For more details on WebSphere MQ functionality, refer to the *WebSphere MQ Queue Manager Clusters*, SC34-6061.

8

Shared queue overview

Since MQSeries for OS/390 V5.2 queue managers in a Parallel Sysplex on z/OS platforms are able to manage nonpersistent shared queues in a queue-sharing group. WebSphere MQ V5.3 for z/OS introduces *persistent* shared queues.

Shared queues are local queues that can be accessed by one or more queue managers in a Parallel Sysplex. Each queue manager that should have access to these shared queues must be a member of a queue-sharing group.

Using shared queues with your applications, you are able to exploit advantages of the Parallel Sysplex, such as high availability, workload balancing and reduction of administrative and operational work.

In this chapter, we overview the concept of shared queues and queue-sharing groups.

8.1 Introduction to a queue-sharing group

In order to use shared queues a queue manager must become a member of a queue-sharing group. A queue-sharing group is a collection of queue managers in a Parallel Sysplex that have access to the same set of shared queues and to shared definitions.

Figure 8-1 shows a setup of a queue-sharing group QSG1 with two queue managers, WMQA and WMQB.

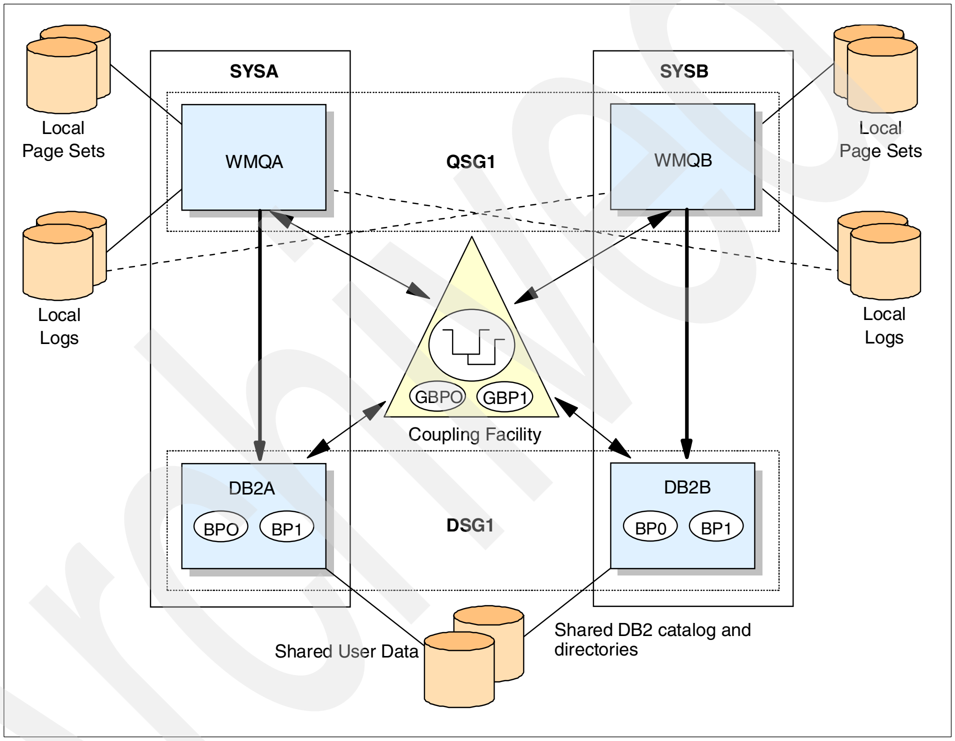


Figure 8-1 Queue-sharing group in a Parallel Sysplex

Both queue managers still have their own local bootstrap data sets, log data sets, and page sets, but in order to allow a queue-sharing group-wide recovery, all queue managers must have access to their own log data sets as well as the log data sets of all other queue managers in the queue-sharing group.

The shared WebSphere MQ object definitions are stored in DB2 tables, and messages belonging to a shared queues reside in one or more coupling facility list structures.

Each queue manager connects to a DB2 subsystem that has to be a member of a DB2 data-sharing group. In our example, the DSQ1 is the data-sharing group with two members, DB2A and DB2B.

8.2 System resources used by a queue-sharing group

If you want to set up a WebSphere MQ queue-sharing group, first of all you have to ensure that all your queue managers that you want to be members of the queue-sharing group are running in the same Parallel Sysplex. The maximum number of queue managers that you can connect to a queue-sharing group is 32.

Coupling facility

WebSphere MQ needs a coupling facility with at least CFLEVEL=9, to process shared messages. This is because CFLEVEL=9 provides XES coupling facility list structure architecture extensions to aid in data management and manipulation in support of WebSphere MQ shared queues.

The coupling facility storage requirements for your queue-sharing group are mainly dependent on:

- How many shared queues you want to define
- How many messages you expect to be in a shared queue at any time

Coupling facility structures

WebSphere MQ uses coupling facility list structures for administrative purposes and for shared queues. You have to define at least two coupling facility structures in your CFRM policy to use shared queues:

1. One administration structure:

   The administration structure does not contain any user data. It is used by WebSphere MQ itself. WebSphere MQ uses the administration structure to coordinate internal activity across the queue-sharing group. This allows all queue managers in the queue-sharing group to exchange and control information.

   The name of an administration structure has to be *qsg-name*CSQ_ADMIN, where *qsg-name* is the name of the queue-sharing group, or the queue-sharing group name padded to four characters with @ symbols, if the queue-sharing group name is less than four characters long. The size of the administration structure should be at least 10 MB.

2. Application structure(s):

   Application structures hold shared messages, which are put to a shared queue.

Depending on your application requirements, you can define one or more application structures, up to a maximum of 63, because the limit for the number of structures a queue manager can connect to is 64, and one is needed for the administration structure. All messages associated with a single shared queue are in a single application structure. A single application structure can contain messages for up to 512 shared queues.

A coupling facility list structure entry has a maximum length of 64 KB. Since each list entry also has to contain header information, the actual size for the messages is limited to 63 KB.

The application structure names have to follow these naming conventions:

– The maximum length of the name is 16 characters.

– The first four characters must equal the queue-sharing group name, or the queue-sharing group name padded to four characters with @ symbols, if the queue-sharing group name is less than four characters long.

– The second part of the structure name, starting with the fifth character, which must be alphabetic, can contain alphabetic and/or numeric characters.

If you refer to your coupling facility structures in your queue manager object definitions, you have to specify only the second part of the application structure, and omit the queue-sharing group name.

In order to size your application structures appropriately, refer to Chapter 16, "Planning your Coupling Facility and DB2 Environment", in *WebSphere MQ for z/OS Concepts and Planning Guide*, GC34-6051.

Note: Do not forget to count the message header length of your message data when calculating the amount of storage needed for your application.

RRS

In order to communicate with a DB2 data-sharing group, you also have to ensure that RRS runs on all images on which you want to implement queue-sharing group queue managers.

Chapter 3, "z/OS Resource Recovery Services (RRS)" on page 43 contains a description of how to set up RRS. This setup was actually done on a different sysplex, but is similar to the setup we use in our test application environment.

8.3 DB2 resources used by a queue-sharing group

To share data in a queue-sharing group, WebSphere MQ uses coupling facility list structures for the shared messages, a set of DB2 tables for the object definitions, and status information. To ensure that all queue managers in a queue-sharing group share the same set of tables, you need to set up a DB2 data-sharing group.

There are three types of tables, which are put into different table spaces:

- Administration tables and all object tables apart from OBJ_B_NAMELIST and OBJ_B_CHANNEL.
- OBJ_B_NAMELIST and OBJ_B_CHANNEL tables. The table space for this table must be associated with a DB2 32 KB buffer pool.
- Channel initiator tables.

Refer to *WebSphere MQ for z/OS Concepts and Planning Guide*, GC34-6051, for a description of all shipped tables and the storage requirements of these tables.

With the exception of the queue-sharing group and queue manager entries, WebSphere MQ adds, updates, and deletes all DB2 entries automatically, when you change your shared queue or group object definitions. In order to manage queue-sharing group and queue manager definitions, WebSphere MQ provides a utility, CSQ5PQSG.

WebSphere MQ queue managers in the queue-sharing group connect to a local DB2 of the data-sharing group. A queue manager needs both:

- The DB2 data-sharing group name
- The name of the local DB2 subsystem or group attachment (which is defined through the subsystem entry of the DB2 subsystem) to which the queue manager is to connect

You have to provide both parameters in the parameter module CSQ6SYSP (see 8.4, "System parameters" on page 141) and you have to create the queue-sharing group and queue manager entries in the DB2 tables.

8.4 System parameters

To support the queue-sharing group environment, one parameter has been added to the system parameter module CSQ6SYSP.

QSGDATA

This is a new parameter that has four positional parameters, which are needed to connect WebSphere MQ to DB2 and the data-sharing group to join a queue-sharing group. The subparameters are:

- QSGNAME

  This is the name of the queue-sharing group to which the queue manager belongs. If you leave this value blank, which is the default setting, the queue manager does not belong to a queue-sharing group.

- DSGNAME

  This is the name of the DB2 data-sharing group on the coupling facility to which the queue manager is connected when it belongs to a queue-sharing group. The DB2 data-sharing group holds the shared repository where all shared objects are stored (for example, queues and channels) that are defined only once and can then be used by any queue manager in the group.

- DB2NAME

  This is the name of the local DB2 subsystem (or group attachment) to which the queue manager is to connect. The DB2 subsystem must be a member of the DB2 data-sharing group specified in the DSGNAME, or if you use the group attachment, the group attachment must be specified in the subsystem entry of the DB2 subsystem belonging to the data-sharing group. All queue managers within the queue-sharing group must specify the same DB2 data-sharing group.

- DB2SERV

  The number of server tasks used for accessing DB2. The value can be from 4 to 10. The default is 4.

8.5 Initialization input data set

Beside the sample input data sets, thlqual.SCSQPROC(CSQ4INSG) and thlqual.SCSQPROC(CSQ4INSX), which contain the system object definitions for non queue-sharing group objects, there is another sample input data set, thlqual.SCSQPROC(CSQ4INSS). This data set contains the system object definitions that are needed if you want to define shared channels in a queue-sharing group. Therefore it has to be added to the CSQINP2 concatenation of the started task procedure of your queue manager.

thlqual.SCSQPROC(CSQ4INSS) adds the following three object definitions: SYSTEM.QSG.CHANNEL.SYNCQ, SYSTEM.QSG.TRANSMIT.QUEUE, and CFSTRUCT.

SYSTEM.QSG.CHANNEL.SYNCQ

This queue is a shared queue that is used to hold synchronization information for shared channels.

SYSTEM.QSG.TRANSMIT.QUEUE

This queue is used to perform fast message transfer between queue managers in a queue-sharing group without defining channels. Each queue manager in the queue-sharing group starts a task called the intra-group queuing agent, which waits for messages to arrive on this queue that are destined for their queue manager. When such a message is detected, it is removed from the queue and placed on the correct destination queue.

CFSTRUCT

This object is used to support the recovery of shared queue messages. Its capabilities are described through the CFLEVEL and RECOVER attributes.

```
DEFINE CFSTRUCT('APPLICATION1') +
        DESCR('CF structure APPLICATION1') +
        CFLEVEL(3) +
        RECOVER(YES)
```

Figure 8-2 Define structure sample copied from CSQ4INSS

DEFINE CFSTRUCT, ALTER CFSTRUCT, DELETE CFSTRUCT, and DISPLAY CFSTRUCT commands have been added for these objects.

> **Note:** The parameter CFLEVEL used by WebSphere MQ to describe the level of the coupling facility structures does not correspond to the CFLEVEL parameter used for the description of the coupling facility levels of the CFCC.

As with the CFSTRUCT attribute of queues, the name is specified without the initial four-character queue-sharing group name that forms the name used by z/OS. All WebSphere MQ messages now use the short form of the name.

- In MQSeries for OS/390 Version 5.2, CF structure objects were implicitly created and deleted. When the first queue naming a CF structure was defined, a CF structure object with that name was implicitly created. Similarly, when the last queue naming a CF structure was deleted, the CF structure object was deleted. These CF structure operations were invisible to the user. Such CF structure objects have CFLEVEL(1).
- CF structure objects defined with CFLEVEL(2) (on a Version 5.3 queue manager) are for compatibility between Version 5.2 and Version 5.3 queue

managers. They can be used by Version 5.2 queue managers, and they can be used and manipulated by Version 5.3 queue managers.

- CF structure objects defined with CFLEVEL(3) are only usable by Version 5.3 queue managers. New function is supported for queues defined on a CFLEVEL(3) CF structure.

Once all queue managers in the queue-sharing group are at Version 5.3 level, you can migrate a CF structure from CFLEVEL(2) to CFLEVEL(3) using the DEFINE REPLACE or ALTER CFSTRUCT commands.

CFLEVEL(3) functions

- Queues defined on a CFLEVEL(3) CF structure can have the new INDXTYPE(GROUPID) attribute.
- Persistent messages can be stored on a queue defined on a CF structure with CFLEVEL(3) and the RECOVER(YES) attribute. New commands BACKUP CFSTRUCT and RECOVER CFSTRUCT are provided to support recovery.

8.6 Shared queues

Queue managers that use a shared queue access the same definition of that shared queue. Each shared queue definition is originally held in a shared repository using DB2 data sharing. The shared queue definition is read from DB2 the first time that shared queue is opened by issuing a MQOPEN, and is then cached in the coupling facility and in local storage. All shared queue messages are owned by the coupling facility and not by the queue manager that originally MQPUT them. Nonpersistent messages thus survive queue manager failure but are lost upon coupling facility failure.

When you define a queue on WebSphere MQ for z/OS, you can choose whether you want to share that queue with other queue managers (a global definition) or whether the queue definition is to be used by one queue manager only (a private definition). This is called the *object disposition*.

QSGDISP(QMGR)

If you define a queue with disposition QMGR, then the definition will be held on page set 0 of the defining queue manager. Messages sent to that queue are available to the queue manager that defined the queue only. This is a private definition.

QSGDISP(COPY)

If a queue is defined with disposition COPY, then the queue is defined on the page set of the queue manager that executes the command using the QSGDISP(GROUP) object of the same name as the LIKE object. For local queues, messages are stored on the page sets of each queue manager and are available only through that queue manager. You cannot define a new queue with the DEFINE QUEUE(q-name) QSGDISP(COPY) command if there is no corresponding GROUP queue. But if you want a local copy of a queue that has previously been defined with QSGDISP(GROUP) for other queue managers in your queue-sharing group, you can do this either via the WebSphere MQ operator screen by choosing **option 4- Manage** or by using the DEFINE command.

QSGDISP(GROUP)

If you define a queue with disposition GROUP, then the object definition is held in the shared repository. If the definition is successful, the following command is generated and sent to all active queue managers to make or refresh local copies on page set zero:

```
DEFINE QUEUE(q-name) REPLACE QSGDISP(COPY)
```

QSGDISP(SHARED)

The definitions of a queue defined with disposition SHARED are held in the shared queue repository and the messages on this queue are available to all queue managers in the queue-sharing group because they are stored in the coupling facility.

8.6.1 Queue parameters

Queue parameters that are either available for shared queues only or have limitations for shared queues are:

- CFSTRUCT(*structure-name)*

  This parameter is supported for local and model queues and specifies the name of the coupling facility structure where you want messages stored when you use shared queues.

  The name of the structure cannot have more than 12 characters and must start with an uppercase letter (A through Z). It can include only the characters A through Z and 0 through 9.

  The name of the queue-sharing group to which the queue manager is connected is prefixed to the name you supply. For example, if the name of the queue-sharing group is QSG1 and you supply the name STRUCT1, then the resulting coupling facility structure name is QSG1STRUCT1.

The structure must be defined in the CFRM policy data set before you can use the queue.

> **Note:** The administration structure for the queue-sharing group (in this case QSG1CSQ_ADMIN) cannot be used for storing messages.

- INDXTYPE

  You can define a shared queue with an index type of NONE, MSGID and CORRELID when you have defined the CF structure with CFLEVEL(1) or CFLEVEL(2). With CFLEVEL(3) you can also define index type GROUPID. Index type MSGTOKEN is not supported on shared queues.

  – NONE

    No index is maintained. Use this when messages are retrieved sequentially.

  – MSGID

    An index of message identifiers is maintained. Use this when messages are retrieved using the message identifier as a selection criterion on the MQGET call with the correlation identifier set to NULL, or use both the message identifier and the correlation identifier as selection criteria.

  – CORRELID

    An index of correlation identifiers is maintained. Use this when messages are retrieved using the correlation identifier as a selection criterion on the MQGET call with the message identifier set to NULL, or use both the message identifier and the correlation identifier as selection criteria.

  – GROUPID

    An index of group identifiers is maintained. Use this when messages need to be retrieved using message grouping selection criteria. You can only specify a shared queue with INDXTYPE(GROUPID) if the queue uses a CF structure at CFLEVEL(3). You cannot set INDXTYPE to GROUPID if the queue is a transmission queue.

- MAXMSGL(integer)

  Because WebSphere MQ uses coupling facility list structures to store messages, the maximum length for messages on a shared queue is 63 KB. Therefore this parameter must be from zero through 64 512 bytes.

- STGCLASS(string)

  If you specify STGCLASS for a shared queue, this parameter is ignored.

- QSVCIEV

  Service Interval High or Service Interval OK events.

  This parameter is not supported on a shared queue and will be ignored if specified.

- QSVCINT(integer)

  The service interval used for comparison to generate Service Interval High and Service Interval OK events.

  This parameter is not supported on a shared queue and will be ignored if specified.

8.6.2 Transmission queues and triggering

A shared transmission queue is used to store messages before they are moved from the queue-sharing group to the destination. It is a shared queue and it is accessible to all queue managers in the queue-sharing group.

Triggering

A triggered shared queue can generate more than one trigger message for a satisfied trigger condition. There is one trigger message generated for each local initiation queue defined on a queue manager in the queue-sharing group. In the case of distributed queuing, each channel initiator receives a trigger message for a satisfied shared transmission queue trigger condition. However, only one channel initiator will actually process the triggered start, and the others will fail safely. The triggered channel is then started with a load-balanced start associated with the triggered shared queue that will be triggered to start channel QSG.TO.QM2.

8.7 Group listener

A queue-sharing group has a generic interface that allows the network to view the group as a single entity. This is achieved by having a single generic address that can be used to connect to any queue manager within the group. Each queue manager in the queue-sharing group listens for inbound session requests on an address that is logically related to the generic address.

For the TCP/IP listener, the specified port has two mutually exclusive means of being connected to the generic address:

- In the case of a front-end router such as the IBM Network Dispatcher (see *Network Dispatcher User's Guide*, GC31–8496), or the Sysplex Distributor that we used in our tests, inbound connect requests are forwarded from the router to the members of the queue-sharing group.

For an example of setting up a Sysplex Distributor, see Chapter 5, “Sysplex Distributor” on page 87.

- In the case of TCP/IP WLM/DNS, each listener registers as being part of the WLM group. This is a registration type model similar to the VTAM generic resource for LU 6.2. For an example of setting up this technology, see “Using WLM/DNS” in the *WebSphere MQ Intercommunication,* SC34-6059. WLM/DNS only maps host names and does not map port numbers. This means that all the group listeners in a queue-sharing group must use the same port number.

8.8 Shared channels

A number of networking products provide a mechanism to hide server failures from the network or to balance inbound network requests across a set of eligible servers. These include:

- VTAM generic resources
- WLM TCP/IP Domain Name System (DNS)
- Network Dispatcher
- Sysplex Distributor

The channel initiator takes advantage of these products to exploit the capabilities of shared queues.

8.8.1 Shared inbound channels

Each channel initiator in the queue-sharing group starts an additional listener task to listen on an IP address/port combination specific to that channel initiator. This combination is the channel initiator’s group address. The group address is made available to the network through one of the technologies mentioned above. This means that an inbound network attach request for the generic address can be dispatched to any one of the listeners in the queue-sharing group that is listening on its group address.

A channel can only be started on the channel initiator to which the inbound attach is directed if the channel initiator has access to a channel definition for a channel with that name. A channel definition can be defined to be private to a queue manager or stored on the shared repository and available anywhere (a group definition). This means that a channel definition can be made available on any channel initiator in the queue-sharing group by defining it as a global definition.

There is an additional difference when starting a channel through a channel initiator's group port: channel synchronization is with the queue-sharing group and not with an individual queue manager. For example, consider a client starting a channel with a CONNAME specifying the generic address. When the channel first starts, the generic address may be mapped to the group address of queue manager QM1. If the channel stops and is restarted on queue manager QM2, information about the number of messages that have flowed is still correct, because the synchronization is with the queue-sharing group.

An inbound channel started through a channel initiator's group port can be used to put messages to both private and shared queues. The client does not know whether the target queue is shared or not.

If the target queue is a shared queue, it can be seen by any queue manager in the queue-sharing group. The client connects through any available channel initiator in a load-balanced fashion and the messages are put to the shared queue.

If the target queue is not a shared queue, a private queue of that name must be present on the queue manager the inbound channel is connected to. When the inbound channel specifies the generic address in its CONNAME, it may be routed to any one of the queue-sharing group queue managers and so it is recommended that a suitable private target queue is defined on each queue manager (the environment is one of replicated private local queues). The name of the queue determines the function regardless of the hosting queue manager.

8.8.2 Shared outbound channels

An outbound channel is considered to be a shared channel if it is taking messages from a shared transmission queue. If it is shared, it holds synchronization information at the queue-sharing group level. This means that the channel can be restarted on a different queue manager and channel initiator instance within the queue-sharing group if the communications subsystem, channel initiator, or queue manager fails. Restarting failed channels in this way is a feature of shared channels called *peer channel recovery*.

Load balanced shared start

A shared transmission queue can be serviced by an outbound channel running on any channel initiator in the queue-sharing group. Load-balanced channel start determines where a start channel command is targeted. An appropriate channel initiator is chosen that has access to the necessary communications subsystem. For example, a channel defined with TRPTYPE(LU6.2) will not be started on a channel initiator that only has access to a TCP/IP subsystem. The choice of channel initiator is dependent on the channel load and the headroom of the channel initiator. The channel load is the number of active channels as a percentage of the maximum number of active channels allowed as defined in the channel initiator parameters (ACTCHL in CSQ6CHIP).

Workload balancing

An outbound shared channel is eligible for starting on any channel initiator within the queue-sharing group, provided that you have not specified that you want it to be started on a particular channel initiator. The channel initiator selected by WebSphere MQ is determined using the following criteria:

- Is the communications subsystem required currently available to the channel initiator?
- Is a DB2 connection available to the channel initiator?
- Which channel initiators have low current workload? The workload includes channels that are active and retrying.

8.8.3 Shared channel status

The channel initiators in a queue-sharing group maintain a shared channel-status table in DB2. This records which channels are active on which channel initiators. If there is a channel initiator or communications system failure, the shared channel status table is used to determine which channels need to be restarted on a different channel initiator in the queue-sharing group. You can display the current channel status of shared channels using the command DISPLAY CHSTATUS(*) CHLDISP(SHARED) CMDSCOPE(*) CURRENT, which will result in an output similar to Figure 8-3 on page 151. That command output shows the responses of our four queue managers in the queue-sharing group. Two channel initiators (MQV1 and MQV3) have no active shared channels, whereas MQV2 and MQV4 have both one active shared channel.

For more information about the command, refer to *WebSphere MQ Script (MQSC) Command Reference*, SC34-6055.

.

```
-MQV1 DISPLAY CHSTATUS(*) CHLDISP(SHARED) CMDSCOPE(*)
CSQN137I -MQV1 'DISPLAY CHSTATUS' command accepted for CMDSCOPE(*),
sent to 4
CSQN121I -MQV1 'DISPLAY CHSTATUS' command responses from MQV1
CSQM134I -MQV1 CSQMDCST  DISPLAY CHSTATUS(*) COMMAND ACCEPTED
CSQN121I -MQV1 'DISPLAY CHSTATUS' command responses from MQV1
CSQM297I -MQV1 CSQXDMSG NO CHSTATUS FOUND MATCHING REQUEST CRITERIA
CSQ9022I -MQV1 CSQXDMSG ' DISPLAY CHSTATUS' NORMAL COMPLETION
CSQN121I -MQV1 'DISPLAY CHSTATUS' command responses from MQV4
CSQM134I -MQV4 CSQMDCST  DISPLAY CHSTATUS(*) COMMAND ACCEPTED
CSQN121I -MQV1 'DISPLAY CHSTATUS' command responses from MQV3
CSQM134I -MQV3 CSQMDCST  DISPLAY CHSTATUS(*) COMMAND ACCEPTED
CSQN121I -MQV1 'DISPLAY CHSTATUS' command responses from MQV2
CSQM134I -MQV2 CSQMDCST  DISPLAY CHSTATUS(*) COMMAND ACCEPTED
CSQN121I -MQV1 'DISPLAY CHSTATUS' command responses from MQV4
CSQM293I -MQV4 CSQXDMSG 1 CHSTATUS FOUND MATCHING REQUEST CRITERIA
CSQM201I -MQV4 CSQXDMSG  DISPLAY CHSTATUS DETAILS 424
CHSTATUS(M23WPH767B.TO.MQVG)
CHLDISP(SHARED)
CONNAME(9.24.104.132)
CURRENT
STATUS(RUNNING)
STOPREQ(NO)
RQMNAME(m23wph767b)
CSQXDMSG END CHSTATUS DETAILS
CSQ9022I -MQV4 CSQXDMSG ' DISPLAY CHSTATUS' NORMAL COMPLETION
CSQN121I -MQV1 'DISPLAY CHSTATUS' command responses from MQV3
CSQM297I -MQV3 CSQXDMSG NO CHSTATUS FOUND MATCHING REQUEST CRITERIA
CSQ9022I -MQV3 CSQXDMSG ' DISPLAY CHSTATUS' NORMAL COMPLETION
CSQN121I -MQV1 'DISPLAY CHSTATUS' command responses from MQV2
CSQM293I -MQV2 CSQXDMSG 1 CHSTATUS FOUND MATCHING REQUEST CRITERIA
CSQM201I -MQV2 CSQXDMSG  DISPLAY CHSTATUS DETAILS 431
CHSTATUS(MQVG.TO.M23WPH767B)
CHLDISP(SHARED)
XMITQ(m23wph767b)
CONNAME(9.24.104.132)
CURRENT
STATUS(RUNNING)
STOPREQ(NO)
RQMNAME(m23wph767b)
CSQXDMSG END CHSTATUS DETAILS
CSQ9022I -MQV2 CSQXDMSG ' DISPLAY CHSTATUS' NORMAL COMPLETION
CSQN122I -MQV1 'DISPLAY CHSTATUS' command for CMDSCOPE(*) normal
completion
```

Figure 8-3 DISPLAY CHSTATUS command

8.9 Intra-group queuing

You can perform fast message transfer between queue managers in a queue-sharing group without defining channels. To do that, you use a system queue called the SYSTEM.QSG.TRANSMIT.QUEUE, which is a shared transmission queue. Each queue manager in the queue-sharing group starts a task called the *intra-group queuing agent* (IGQ agent), which waits for messages to arrive on this queue that are destined for their queue manager. When such a message is detected, it is removed from the queue and placed on the correct destination queue.

Standard name resolution rules are used, but if intra-group queuing is enabled and the target queue manager is within the queue-sharing group, the SYSTEM.QSG.TRANSMIT.QUEUE is used to transfer the message to the correct destination queue manager instead of using a transmission queue and channel.

Intra-group queuing is enabled through a queue manager attribute at startup. It can only be used to move messages with a message length of up to 63 KB, including the transmission-queue header (63 KB is the maximum message length for shared queues). Intra-group queuing moves nonpersistent messages outside of syncpoint scope and persistent messages within syncpoint scope. If intra-group queuing encounters a problem delivering messages to the target queue, it puts them to the dead-letter queue. If the dead-letter queue is full, or if you have not defined a dead-letter queue, nonpersistent messages are discarded but persistent messages are sent back to the SYSTEM.QSG.TRANSMIT.QUEUE, and the IGQ agent retries to deliver the messages until it is successful.

The default setup for intra-group queuing is IGQ(DISABLED). IGQ can be enabled either with the command ALTER QMGR IGQ(ENABLED) or by using the WebSphere MQ operator screens.

For more information about intra-group queuing, refer to *WebSphere MQ Intercommunication*, SC34-6059.

8.10 Advantage of shared queues and shared channels

With shared queues you have the advantage that every queue manager in the queue-sharing group has access to the same set of queues. Because of this, an application does not depend on the availability of one particular queue manager. Any queue manager in the group can service any shared queue and can continue processing a queue if a queue manager in the queue-sharing group stops. WebSphere MQ detects if a queue manager disconnects from the coupling facility abnormally and, where possible, other queue managers in the group perform peer recovery to complete pending units of work for that queue manager.

With shared channels, WebSphere MQ provides functions that give high availability to the network. The channel initiator enables you to use network products such as the Sysplex Distributor for workload balancing across a set of eligible queue managers and to hide queue manager failures. If one queue manager within the queue-sharing group fails, remote queue managers are still able to connect to the queue-sharing group through the shared channel and to put their messages on to shared queues. This is possible because WebSphere MQ uses a generic port for inbound request so that attached requests can be routed to any available channel initiator in the queue-sharing group. Shared outbound channels take the messages they send from a shared transmission queue. Because information about the status of a shared channel is held in one place for the whole queue-sharing group, the channel can be restarted automatically on a different channel initiator in the queue-sharing group if the channel initiator, queue manager, or communications subsystem fails.

8.11 Changes to the security concept

WebSphere MQ now supports security checking based on the queue-sharing group level, which allows you to simplify security administration in a queue-sharing group.

The profiles used for checking access authority on a queue-sharing group level are similar to the profiles used for a single queue manager, except that the profile is not prefixed by the queue manager name, but by the queue-sharing group name. Figure 8-4 shows the commands to define profiles for all queues CICS.MQVG.ECHO*, for both the queue-sharing group MQVG and the queue managers, which are members of MQVG.

```
RDEFINE MQQUEUE MQVG.CICS.MQVG.ECHO*

RDEFINE MQQUEUE MQV1.CICS.MQVG.ECHO*
RDEFINE MQQUEUE MQV2.CICS.MQVG.ECHO*
RDEFINE MQQUEUE MQV3.CICS.MQVG.ECHO*
RDEFINE MQQUEUE MQV4.CICS.MQVG.ECHO*
```

Figure 8-4 Queue profile RACF definitions for queue manager and queue-sharing group

To control the security checking performed by WebSphere MQ, you must define *switch profiles*. A switch profile is a normal RACF profile that has a special meaning in WebSphere MQ. The access list in switch profiles is not used by WebSphere MQ.

To control whether subsystem security is required in a queue-sharing group, WebSphere MQ checks three switch profiles in the following order:

1. *qmgr-name*.NO.SUBSYS.SECURITY, which controls the subsystem security for this queue manager
2. *qsg-name*.NO.SUBSYS.SECURITY, which controls the subsystem security for this queue-sharing group
3. *qmgr.name*.YES.SUBSYS.SECURITY, which is used to specify subsystem security override for this queue manager

If your queue manager is not a member of a queue-sharing group, WebSphere MQ checks for the *qmgr-name*.NO.SUBSYS.SECURITY switch profile only.

Another set of switch profiles is provided to control queue-sharing group or queue manager level security. Checking these switches, WebSphere MQ determines whether checking is required at the queue-sharing group or queue manager level, or both. These checks are not performed if the queue manager is not a member of a queue-sharing group, and they are also not performed when WebSphere MQ has already determined that no security checking is required.

A third set of switches is used to specify resource level checks.

WebSphere MQ now issues messages at startup time and whenever you issue a security refresh to show which switches are active. Figure 8-5 on page 155 shows the output messages written by queue manager MQV1 at startup.

```
CSQH024I -MQV1 CSQHINSQ SUBSYSTEM security switch set
ON, profile 'MQVG.NO.SUBSYS.SECURITY' not found
CSQH024I -MQV1 CSQHINSQ QMGR security switch set ON,
profile 'MQVG.NO.QMGR.CHECKS' not found
CSQH024I -MQV1 CSQHINSQ QSG security switch set ON,
profile 'MQVG.NO.QSG.CHECKS' not found
CSQH024I -MQV1 CSQHISAC CONNECTION security switch
set ON, profile 'MQVG.NO.CONNECT.CHECKS' not found
CSQH024I -MQV1 CSQHISAC COMMAND security switch set
ON, profile 'MQVG.NO.CMD.CHECKS' not found
CSQH024I -MQV1 CSQHISAC CONTEXT security switch set
ON, profile 'MQVG.NO.CONTEXT.CHECKS' not found
CSQH024I -MQV1 CSQHISAC ALTERNATE USER security
switch set ON, profile 'MQVG.NO.ALTERNATE.USER.CHECKS' not found
CSQH024I -MQV1 CSQHISAC COMMAND RESOURCES security
switch set ON, profile 'MQVG.NO.CMD.RESC.CHECKS' not found
CSQH024I -MQV1 CSQHISAC PROCESS security switch set
ON, profile 'MQVG.NO.PROCESS.CHECKS' not found
CSQH024I -MQV1 CSQHISAC NAMELIST security switch set
ON, profile 'MQVG.NO.NLIST.CHECKS' not found
CSQH024I -MQV1 CSQHISAC QUEUE security switch set ON,
profile 'MQVG.NO.QUEUE.CHECKS' not found
```

Figure 8-5 Security switch messages

In this redbook, we do not discuss queue-sharing group security or provide details about the security setup of a queue-sharing group. For more information on security setup, refer to *WebSphere MQ for z/OS System Setup Guide*, SC34-6052.

8.12 Shared queue performance considerations

For information regarding performance considerations in queue-sharing environment, refer to SupportPacs MP16 (Capacity planning and tuning for MQSeries for OS/390) and MP1D (WebSphere MQ for z/OS V5.3 Performance Report).

Part 3

Implementation scenarios

In Part 3 of this redbook, we describe our test implementation scenarios and cover the following topics:

- Implementation of a queue-sharing environment
- Migration from MQSeries V5.2
- High-availability configurations
- CICS, WebSphere MQ Integrator Broker, and IMS applications in a queue-sharing environment

9

Creating a shared queue environment

Implementation of a queue-sharing group with persistent shared messages requires WebSphere MQ for z/OS V5.3. If your queue managers are running with previous versions, you have to migrate to V5.3 first, which we examine in Chapter 10, “Migration to WebSphere MQ for z/OS Version 5.3” on page 193.

If your queue managers are already participants in a queue-sharing group, you are able to define persistent shared queues after you have completed your migration.

If you want to implement a queue-sharing group for the first time, you must follow the instructions in *WebSphere MQ for z/OS System Setup Guide,* SC34-6052.

This chapter contains a sample setup for a queue-sharing group, with two queue managers performing as gateways and two queue managers providing application functions.

We describe the setup steps according to our general description of a queue-sharing group in Chapter 8, “Shared queue overview” on page 137.

9.1 Test specifications

For our test environment we want to have a queue-sharing group with four queue managers. Two queue managers should run on the z/OS images, where our applications are running, and the other two queue managers should work as gateways to all incoming requests.

With this configuration we want to be able to:

- Manage our workload in the sysplex
- Provide high availability in case of failures
- Reduce the time of planned outages, for example for service purposes

In our sysplex, WTSCPLX1, we have four systems that we can use for our queue-sharing group setup. The application environment exists on system SC53 and SC66. For the gateway queue managers, we can use systems SC61 and SC62.

Table 9-1 shows the names of all queue managers, their roles, and the DB2 subsystem to which the queue manager connects, as well as the z/OS image where these subsystems are to be started.

Table 9-1 Queue managers and their role, DB2 subsystems, and z/OS systems in the test scenario

Queue manager	WebSphere MQ user data HLQ	Role of queue manager	DB2 subsystem	z/OS system
MQV1	MQV1	Application queue manager	D7V1	SC53
MQV2	MQV2	Gateway queue manager	D7V2	SC61
MQV3	MQV3	Application queue manager	D7V3	SC66
MQV4	MQV4	Gateway queue manager	D7V4	SC62

All four queue managers are members of the queue-sharing group MQVG, and they all belong to the cluster CLUSPLX1. The HLQ of our WebSphere MQ target data sets is MQ530, which is actually an alias pointing to the current active set of libraries.

All DB2 subsystems are members of the data-sharing group DB7VU. DB7VU is also the HLQ for DB2 user data. The HLQ for our DB2 target data sets is DB7V7.

For our application tests, we use scenarios described in Chapter 12, “CICS applications and queue sharing” on page 225.

9.2 Base system definitions and requirements

We install the WebSphere MQ V5.3 software using SMP/E. Refer to the *Program Directory for WebSphere MQ for z/OS*, GI10-2548, for more information on the space requirements and the SMP/E installation.

9.2.1 Check system prerequisites

First of all, we have to check if our environment fulfills all requirements for a WebSphere MQ queue-sharing group.

All our systems are running in the same Parallel Sysplex

The system command D XCF requests sysplex information. Example 9-1 shows the sysplex and the images running in the sysplex.

Example 9-1 Output for command D XCF

```
D XCF
IXC334I  16.04.53  DISPLAY XCF 103
   SYSPLEX WTSCPLX1:   SC50            SC52            SC53
                       SC54            SC55            SC61
                       SC62            SC66
```

All four systems that we want to use for our queue-sharing group queue managers are members of the sysplex WTSCPLX1.

We have access to a coupling facility

The coupling facilities we can use for our WebSphere MQ structures are CF05 and CF06.

To see the CFLEVEL and the available space we can use for our structures, we use the commands:

```
D CF,CFNAME=CF05
D CF,CFNAME=CF06
```

Figure 2-1 on page 16 contains the output for coupling facility CF05.

In order to display which systems are connected to the coupling facilities and which structures are allocated in the coupling facilities, we use the commands:

```
D XCF,CF,CFNAME=CF05
D XCF,CF,CFNAME=CF06
```

Figure 9-1 shows the output for coupling facility CF06 with the already allocated WebSphere MQ structures.

```
D XCF,CF,CFNAME=CF06
IXC362I  17.59.56  DISPLAY XCF 597
CFNAME: CF06
  COUPLING FACILITY     :   002064.IBM.02.000000010ECB
                            PARTITION: F   CPCID: 00
  POLICY DUMP SPACE SIZE:   2048 K
  ACTUAL DUMP SPACE SIZE:   2048 K
  STORAGE INCREMENT SIZE:   256 K

  CONNECTED SYSTEMS:
  SC04     SC42     SC43     SC47     SC48     SC49     SC50
  SC52     SC53     SC54     SC55     SC61     SC62     SC66
  SC67     SC69

  STRUCTURES:
  CACHECIC          CACHECICS         CICS_CACHE        DB2V714B_SCA
  DB7QU_SCA         DB7RU_GBP0        DB7RU_LOCK1       DB7RU_SCA
  DB7VU_GBP0        DB7VU_GBP1        DB7VU_GBP2        DB7XU_SCA
  IGWLOCK00         ISGLOCK           ISTGENERIC        IXC_DEFAULT_1
  MQVGAPPLICATION1 MQVGCSQ_ADMIN     RRS_DELAYEDUR_1   RRS_RMDATA_1
  SYSTEM_OPERLOG    SYSZWLM_WORKUNIT SYSZWLM_0ECB2064
```

Figure 9-1 Output for command D XCF,CF,CFNAME=CF06

System Logger and RRS are running

RRS is implemented on all four systems. Therefore we know that System Logger is also running, because System Logger is a prerequisite of RRS. The setup of RRS on these four systems is similar to the setup we described in Chapter 3, “z/OS Resource Recovery Services (RRS)” on page 43.

Additional information on the current sysplex setup

In addition to our checks, you might want to have more information about the couple data sets and the structures defined in your sysplex. You find a summary of commands to use for managing coupling facility resources in Chapter 4, “Managing Coupling Facility Resources”, in *z/OS MVS Setting Up a Sysplex,* SA22-7625, and a detailed description of these commands in *z/OS MVS System Commands,* SA22-7627.

9.2.2 Define coupling facility resources

Since we are using only a few shared queues and group definitions, we use the recommended minimum size for the administration structure of 10 MB for the initial size (INITSIZE) of the structure, and we use 20 MB for the SIZE statement.

For our applications we use only one structure with the default name APPLICATION1 (the coupling facility structure name MQVGAPPLICATION1) and a size of 20 MB, which allows us to store approximately 10000 messages with a message size of less then 1024 bytes, when the structure is initially allocated. The SIZE parameter allows you to specify the maximum structure size. Example 9-2 shows the updated statements of the sample thlqual.SCSQPROC(CSQ4CFRM).

Example 9-2 Administration and application structures for queue-sharing group MQVG

```
STRUCTURE NAME(MQVGCSQ_ADMIN)
      INITSIZE(10240)
      SIZE(20480)
      PREFLIST(CF06,CF05)
      REBUILDPERCENT(5)
      FULLTHRESHOLD(85)

STRUCTURE NAME(MQVGAPPLICATION1)
       INITSIZE(20480)
       SIZE(81920)
       PREFLIST(CF06,CF05)
       REBUILDPERCENT(5)
       FULLTHRESHOLD(85)
```

As you can see in Example 9-2 we:

- Use MQVG as the queue-sharing group name.
- Define 10240 (10 MB) for initial size and 20480 (20 MB) for the structure size of the administration structure.
- Define 20480 (20 MB) for initial size and 81920 (80 MB) for the structure size of the application structure.
- Specify our coupling facilities CF05 and CF06 for the PREFLIST parameter, since we want to use these coupling facilities for our WebSphere MQ structures.

For more information and performance-related recommendations, refer to SupportPac MP1D (WebSphere MQ for z/OS V5.3 Performance Report).

9.2.3 General system definitions

If your queue managers are already installed and you only want to implement the queue-sharing group, no further system action is required. Otherwise you have to follow the customization tasks 1-7 in *WebSphere MQ for z/OS System Setup Guide V5.3,* SC34-6052, which include:

- Add necessary libraries to your APF authorization list. The libraries that have to be APF authorized are:
  - SCSQAUTH
  - SCSQLINK
  - SCSQANLx and SCSQSNLx, where x is your language letter
  - SCSQMVR1 for the non-CICS mover
  - SCSQMVR2 for the Interlink native interface SNSTCPACCESS

  You can also add those libraries dynamically by using the following command:

  ```
  SETPROG APF,ADD,DSNAME=dsn-name,VOLUME=volume
  ```

- Add the data sets, where the WebSphere MQ modules CSQ3INI, CSQ3EPX (delivered in thlqual.SCSQLINK) and CSQ3ECMX (delivered in thlqual.SCSQSNL*x*, where *x* is your language letter), to your LPA concatenation. This is necessary to support the REFRESH QMGR TYPE(EARLY) command. Example 9-3 shows the commands we use to add the required modules dynamically to our LPA concatenation.

Example 9-3 Dynamically add WebSphere MQ early code modules to LPA

```
SETPROG LPA,ADD,MODNAME=(CSQ3INI,CSQ3EPX),DSNAME=MQ530.SCSQLINK
SETPROG LPA,ADD,MODNAME=(CSQ3ECMX),DSNAME=MQ530.SCSQSNLE
```

- Add necessary WebSphere MQ data sets to your link list. You can add the following to your link list concatenation:
  - SCSQAUTH
  - SCSQANLx, where x is your language letter
  - SCSQMVR1 or SCSQMVR2. Since all modules in these libraries are reentrant, the libraries can be placed in the LPA as well.

  In our environment, we neither place the libraries in the link list nor in the LPA, but we include them in queue manager and channel initiator procedures STEPLIB DDcard.

- Update the z/OS program properties table by adding an entry for CSQYASCP.
- Define the WebSphere MQ subsystem to z/OS. You need a separate subsystem for each queue manager. The name of the subsystem is the queue manager name. You can add the subsystem dynamically, but you also have to add an entry in IEFSSNxx SYS1.PARMLIB member to make the subsystem available after subsequent IPL. It is recommended that you define every subsystem on every MVS image to allow ARM restart.

In our environment we have all our queue managers defined with sysplex start scope, as shown in Example 9-4. Therefore we only have to route the START QMGR command to a specific system.

Example 9-4 Subsystem definitions for queue managers in queue-sharing group MQGV

```
SUBSYS SUBNAME(MQV1) INITRTN(CSQ3INI) INITPARM('CSQ3EPX,-MQV1,S')
SUBSYS SUBNAME(MQV2) INITRTN(CSQ3INI) INITPARM('CSQ3EPX,-MQV2,S')
SUBSYS SUBNAME(MQV3) INITRTN(CSQ3INI) INITPARM('CSQ3EPX,-MQV3,S')
SUBSYS SUBNAME(MQV4) INITRTN(CSQ3INI) INITPARM('CSQ3EPX,-MQV4,S')
```

- Create procedures for the WebSphere MQ queue manager and channel initiator address spaces in your procedure library. WebSphere MQ provides the samples CSQ4MSTR and CSQ4CHIN in SCSQPROC.

  The first four letters of your procedure names have to match your queue manager name, and the second part of the name has to be either MSTR for the queue manager itself, or CHIN for the channel initiator address space.

  In our environment, we have to add SDSNLOAD to the STEPLIB DDcard of the queue manager procedure and we use SCSQMVR1 for the channel initiator procedure.

  Figure 9-2 on page 166 and Figure 9-3 on page 167 show the procedures we use for our queue manager MQV1.

```
//MQV1MSTR PROC
//* ******************************************** *
//* *  MQV1 queue manager address space        * *
//* ******************************************** *
//PROCSTEP EXEC PGM=CSQYASCP,REGION=0M
//* -------------------------------------------- *
//* -  steplib                                 - *
//* -------------------------------------------- *
//STEPLIB  DD  DISP=SHR,DSN=MQ530.SCSQANLE
//         DD  DISP=SHR,DSN=MQV1.USERAUTH
//         DD  DISP=SHR,DSN=MQ530.SCSQAUTH
//         DD  DISP=SHR,DSN=DB7V7.SDSNLOAD
//* -------------------------------------------- *
//* -  bootstrap and page set data sets        - *
//* -------------------------------------------- *
//BSDS1    DD  DISP=SHR,DSN=MQV1.BSDS01,DISP=SHR
//BSDS2    DD  DISP=SHR,DSN=MQV1.BSDS02,DISP=SHR
//CSQP0000 DD  DISP=SHR,DSN=MQV1.PSID00,DISP=SHR
//CSQP0001 DD  DISP=SHR,DSN=MQV1.PSID01,DISP=SHR
//CSQP0002 DD  DISP=SHR,DSN=MQV1.PSID02,DISP=SHR
//CSQP0003 DD  DISP=SHR,DSN=MQV1.PSID03,DISP=SHR
//CSQP0004 DD  DISP=SHR,DSN=MQV1.PSID04,DISP=SHR
//* -------------------------------------------- *
//* -  queue manager initialization parameters - *
//* -------------------------------------------- *
//CSQINP1  DD  DISP=SHR,DSN=MQV1.INSTALL.JCL(MQV1INP1)
//CSQINP2  DD  DISP=SHR,DSN=MQV1.INSTALL.JCL(MQV1INSG)
//         DD  DISP=SHR,DSN=MQV1.INSTALL.JCL(MQV1INSX)
//         DD  DISP=SHR,DSN=MQV1.INSTALL.JCL(MQV1INSS)
//         DD  DISP=SHR,DSN=MQV1.INSTALL.JCL(MQV1INYG)
//         DD  DISP=SHR,DSN=MQV1.INSTALL.JCL(MQV1STAC)
//         DD  DISP=SHR,DSN=MQV1.INSTALL.JCL(MQV1DISP)
//* -------------------------------------------- *
//* -  output                                  - *
//* -------------------------------------------- *
//CSQOUT1  DD  SYSOUT=*
//CSQOUT2  DD  SYSOUT=*
```

Figure 9-2 Queue manager procedure for MQV1

```
//MQV1CHIN PROC
//* ******************************************** *
//* *  MQV1 channel initiator address space    * *
//* ******************************************** *
//PROCSTEP EXEC PGM=CSQXJST,REGION=0M
//* -------------------------------------------- *
//* -  steplib                                 - *
//* -------------------------------------------- *
//STEPLIB  DD  DISP=SHR,DSN=MQ530.SCSQANLE
//         DD  DISP=SHR,DSN=MQV1.USERAUTH
//         DD  DISP=SHR,DSN=MQ530.SCSQAUTH
//         DD  DISP=SHR,DSN=MQ530.SCSQMVR1
//* -------------------------------------------- *
//* -  channel initiator initialization parms  - *
//* -------------------------------------------- *
//CSQINPX  DD  DISP=SHR,DSN=MQV1.INSTALL.JCL(MQV1STRT)
//* -------------------------------------------- *
//* -  output                                  - *
//* -------------------------------------------- *
//SYSPRINT DD  SYSOUT=*
//CSQSNAP  DD  SYSOUT=*
//CSQOUTX  DD  SYSOUT=*
```

Figure 9-3 Channel initiator procedure for MQV1

9.2.4 Implement your ESM security controls

Besides the security definitions for data sets and your started tasks, you can use an external security manager (ESM) to protect WebSphere MQ resources.

Since in our environment the WebSphere MQ security classes are active, and no security switches are set, we need generic profiles for all classes to permit access to all WebSphere MQ resources.

If you have implemented protection of WebSphere MQ for your queue managers and you want to migrate to a queue-sharing group, you might need to define additional profiles for queue-sharing group checks.

We recommend that you consider changing your security, taking into consideration the new possibilities with group-wide security checks. If you want to migrate to a queue-sharing group before changing your security design, you can *either* add the single profile qsg-name.NO.QSG.CHECKS, or a profile qmgr-name.NO.QSG.CHECKS for each queue manager in your queue-sharing group to avoid security checks at a queue-sharing group level.

9.2.5 Operations and control screens setup

For our operations and control screens setup, we decide to use dynamic allocation of the necessary libraries.

In order to allocate SCSQAUTH, SCSQANLE, and our user library at ISPF startup we use a REXX EXEC, which issues a TSOLIB ACTIVATE command, shown in Example 9-5.

Example 9-5 TSOLIB ACTIVATE call to activate user libraries at ISPF startup

```
"TSOLIB ACTIVATE DSNAME('MQVG.LOADLIB','MQ530.SCSQANLE',",
                        "'MQ530.SCSQAUTH') QUIET"
```

Since we encountered problems when invoking the interface without previously making all ISPF libraries available, we created another REXX EXEC script. We specify the name of this EXEC script in our ISPF primary option menu for the invocation of WebSphere MQ administration. The EXEC script is copied to a data set, which is concatenated to the SYSPROC concatenation at LOGON time.

```
/* REXX */

csqpfx = "MQ530"

"ISPEXEC LIBDEF ISPLLIB DATASET ID('"csqpfx".SCSQAUTH')"
"ISPEXEC LIBDEF ISPMLIB DATASET ID('"csqpfx".SCSQMSGE')"
"ISPEXEC LIBDEF ISPPLIB DATASET ID('"csqpfx".SCSQPNLE')"
"ISPEXEC LIBDEF ISPSLIB DATASET ID('"csqpfx".SCSQSKL')"
"ISPEXEC LIBDEF ISPTLIB DATASET ID('"csqpfx".SCSQTBLE')"
"ALTLIB ACTIVATE APPLICATION(CLIST) DATASET('"csqpfx".SCSQEXEC')"

"ISPEXEC SELECT CMD(%CSQOREXX "csqpfx" E) NEWAPPL(CSQO) PASSLIB"

"ISPEXEC LIBDEF ISPLLIB"
"ISPEXEC LIBDEF ISPMLIB"
"ISPEXEC LIBDEF ISPPLIB"
"ISPEXEC LIBDEF ISPSLIB"
"ISPEXEC LIBDEF ISPTLIB"
"ALTLIB DEACTIVATE APPLICATION(CLIST)"
exit 0
```

Figure 9-4 Sample EXEC to invoke WebSphere MQ operations and control screens

9.3 DB2 definitions

Many customers have dedicated database administrators who are responsible for database setup and maintenance, and therefore can support you when you set up the DB2 environment you need for WebSphere MQ queue-sharing group.

Nevertheless in case of problems, it might be useful for a WebSphere MQ administrator to check not only the WebSphere MQ environment but also DB2 resources.

We provide a short introduction to the tools we use to verify and maintain DB2 resources before we describe our DB2 setup.

9.3.1 DB2 administration

To get information about the DB2 environment we make use of several methods:

- Use the DB2 ISPF application interface DB2I to either:
  - Issue DB2 commands or
  - Process SQL commands via SPUFI
- Use the subsystem interface, to enter DB2 commands on a system console
- Use the DB2 administration tool to manage DB2 resources

In order to set up your ISPF environment to support the DB2 ISPF application interface and the DB2 administration tool, refer to *DB2 UDB for OS/390 and z/OS Installation Guide*, GC26-9936 and *DB2 Administration Tool User's Guide,* SC27-0974.

DB2 ISPF interface

The DB2I primary option menu allows you to select several actions. Figure 9-5 on page 170 shows this menu. In the top right corner, you see the default DB2 subsystem to which we connect. If you want to change the default subsystem or any other global parameter, use option D, DB2I defaults.

```
                            DB2I PRIMARY OPTION MENU        SSID: D7V1
COMMAND ===>

Select one of the following DB2 functions and press ENTER.

 1  SPUFI                   (Process SQL statements)
 2  DCLGEN                  (Generate SQL and source language declarations)
 3  PROGRAM PREPARATION     (Prepare a DB2 application program to run)
 4  PRECOMPILE              (Invoke DB2 precompiler)
 5  BIND/REBIND/FREE        (BIND, REBIND, or FREE plans or packages)
 6  RUN                     (RUN an SQL program)
 7  DB2 COMMANDS            (Issue DB2 commands)
 8  UTILITIES               (Invoke DB2 utilities)
 D  DB2I DEFAULTS           (Set global parameters)
 X  EXIT                    (Leave DB2I)








PRESS:                      END to exit      HELP for more information
```

Figure 9-5 DB2I primary option menu

If you want to execute DB2 commands, select **option 7, DB2 COMMANDS**. You then can enter DB2 commands. For a list of DB2 commands, refer to *DB2 UDB for OS/390 and z/OS Command Reference*, SC26-9934.

If you want to use SQL commands, you can use option 1, SPUFI. In order to use SPUFI, you must provide two data sets:

- Input data set: This data set can be either sequential or partitioned, but it must be fixed blocked with a record length of 80.
- Output data set, which must be a sequential data set.

In addition to the data sets, you can change your SPUFI defaults by specifying YES for the option 5, CHANGE DEFAULTS. In Figure 9-6 on page 171 you can see that option 5 is set to NO, because our defaults are set up already. We also use the option AUTOCOMMIT.

```
                           SPUFI                              SSID: D7V1
===>

Enter the input data set name:        (Can be sequential or partitioned)
 1  DATA SET NAME ... ===> 'MQRES1.SQL.CNTL(SELECT)'
 2  VOLUME SERIAL ... ===>            (Enter if not cataloged)
 3  DATA SET PASSWORD ===>            (Enter if password protected)

Enter the output data set name:       (Must be a sequential data set)
 4  DATA SET NAME ... ===> 'MQRES1.DB2.OUT'

Specify processing options:
 5  CHANGE DEFAULTS   ===> NO         (Y/N - Display SPUFI defaults screen?)
 6  EDIT INPUT ...... ===> YES        (Y/N - Enter SQL statements?)
 7  EXECUTE ......... ===> YES        (Y/N - Execute SQL statements?)
 8  AUTOCOMMIT ...... ===> YES        (Y/N - Commit after successful run?)
 9  BROWSE OUTPUT ... ===> YES        (Y/N - Browse output data set?)

For remote SQL processing:
10  CONNECT LOCATION  ===>


PRESS:  ENTER to process    END to exit             HELP for more information
```

Figure 9-6 DB2 SPUFI menu

If option 6, EDIT INPUT is set to YES, your input data set is opened and you can edit your SQL statements. In order to process your SQL statements, close the input data set by pressing the End key (PF3). Once again the SPUFI menu is displayed, but you can see an asterisk for option 6, EDIT INPUT. If you then press the Enter key, option 7, EXECUTE is processed. The result is displayed in an ISPF browse session and stored in your output data set.

DB2 subsystem interface

If DB2I is not available, and you want to execute DB2 commands, you can use the DB2 subsystem interface by entering your commands on a console. To use the subsystem interface, you have to place the command prefix string of your DB2 subsystem in front of your command. Figure 9-7 on page 172 shows the DB2 command to display the DB2 data-sharing group and the output returned on a system console.

```
-D7V1 DISPLAY GROUP
DSN7100I  -D7V1 DSN7GCMD
*** BEGIN DISPLAY OF GROUP(DB7VU   ) GROUP LEVEL(710)
                                     GROUP ATTACH NAME(D7VG)
--------------------------------------------------------------------
DB2                                  DB2 SYSTEM    IRLM
MEMBER   ID  SUBSYS CMDPREF   STATUS   LVL NAME     SUBSYS IRLMPROC
-------- --- ----   --------  -------- --- -------- ----   --------
D7V1       1 D7V1   -D7V1     ACTIVE   710 SC53     I7V1   D7V1IRLM
D7V2       2 D7V2   -D7V2     ACTIVE   710 SC61     I7V2   D7V2IRLM
D7V3       3 D7V3   -D7V3     ACTIVE   710 SC66     I7V3   D7V3IRLM
D7V4       4 D7V4   -D7V4     ACTIVE   710 SC62     I7V4   D7V4IRLM
--------------------------------------------------------------------
SCA   STRUCTURE SIZE:     8192 KB, STATUS= AC,   SCA IN USE:     1 %
LOCK1 STRUCTURE SIZE:     8192 KB
NUMBER  LOCK ENTRIES:     2097152
NUMBER  LIST ENTRIES:       20516, LIST ENTRIES  IN USE:           7
*** END DISPLAY OF GROUP(DB7VU   )
DSN9022I  -D7V1 DSN7GCMD 'DISPLAY GROUP ' NORMAL COMPLETION
```

Figure 9-7   Display of the data-sharing group DB7VU

DB2 Administration Tool

Many DB2 administrative tasks can be performed with the help of DB2 Administration Tool for z/OS, part of the IBM family of data management tools. We strongly recommend you make use of this product.

9.3.2 Set up the DB2 environment

Before we run the setup jobs, we check that we have sufficient DB2 authority. Since our DB2 subsystems are only set up for WebSphere MQ support, we have DB2 authority SYSADM, and therefore we are able to run all setup jobs.

For all queue managers in the queue-sharing group, we use local DB2 subsystems to which the queue managers connect, and which we reference in the system parameter module. In addition, these DB2 subsystems must all be members of the same data-sharing group. In Example 9-6 we show the subsystem definitions of all four DB2 subsystems and their associated IRLM subsystems that we use in our test environment.

Example 9-6   DB2 and IRLM subsystem definitions

```
SUBSYS SUBNAME(I7V1)                                    /* IRLM FOR D7V1 */
SUBSYS SUBNAME(I7V2)                                    /* IRLM FOR D7V2 */
SUBSYS SUBNAME(I7V3)                                    /* IRLM FOR D7V3 */
```

```
SUBSYS SUBNAME(I7V4)                                      /* IRLM FOR D7V4 */
SUBSYS SUBNAME(D7V1)                                      /* DB2 V710 D7V1 */
       INITRTN(DSN3INI) INITPARM('DSN3EPX,-D7V1,S,D7VG')
SUBSYS SUBNAME(D7V2)                                      /* DB2 V710 D7V2 */
       INITRTN(DSN3INI) INITPARM('DSN3EPX,-D7V2,S,D7VG')
SUBSYS SUBNAME(D7V3)                                      /* DB2 V710 D7V3 */
       INITRTN(DSN3INI) INITPARM('DSN3EPX,-D7V3,S,D7VG')
SUBSYS SUBNAME(D7V4)                                      /* DB2 V710 D7V3 */
       INITRTN(DSN3INI) INITPARM('DSN3EPX,-D7V4,S,D7VG')
```

We check that by using the DB2 command -DISPLAY GROUP. In the output you can see in Figure 9-7 on page 172, we see the group DB7VU with all members and their status (in our case all members are active), and where those members are currently running.

The setup jobs have to be done once for each data-sharing group. In our case there is only one data-sharing group and therefore we need to run these jobs only once.

All jobs for the DB2 setup are contained in the hlq.SCSQPROC library and use the same JCL, except that the input SYSIN changes. Figure 9-8 shows the job we use.

```
//CSQ45JOB JOB
//STEP1    EXEC PGM=IKJEFT01,REGION=4M,DYNAMNBR=20
//STEPLIB  DD  DISP=SHR,DSN=DB7V7.SDSNLOAD
//SYSPRINT DD  SYSOUT=*
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
  DSN SYSTEM(D7V1)
  RUN PROGRAM(DSNTIAD) PLAN(DSNTIA71) LIB('DB7VU.RUNLIB.LOAD')
/*
//SYSIN    DD  *
  .
  . changed input from sample jobs
  .
/*
```

Figure 9-8 DB2 sample set up job

In Figure 9-8 you can see that we connect to our DB2 subsystem D7V1 and we are using the DB2 plan DSNTIA71 for program DSNTIAD, which we stored during DB2 setup into DB7VU.RUNLIB.LOAD.

CSQ45CSG - create DB2 storage group

With this job we create a DB2 storage group, which we use for our WebSphere MQ DB2 resources. Example 9-7 shows the input for this job.

Example 9-7 Create DB2 storage group

```
//SYSIN    DD *
  CREATE STOGROUP MQVGSTG
         VOLUMES('TOTDCR') VCAT DB7VU;
/*
```

Beside the DB2 storage group name, we have to specify the volumes where we want our data stored and the ICF catalog shot name, which equals the high-level qualifier of the data sets that are allocated by DB2 and that contain our user data.

You might want to skip this step and use a DB2 storage group, which already exists on your system. You then have to use this name in the subsequent jobs.

CSQ45CDB - create DB2 database

This job creates a database for our queue-sharing group data. We assign the storage group MQVGSTG as the default storage group and the bufferpool BP32K1 as the default buffer pool to this database. Example 9-8 shows our input.

Example 9-8 Create DB2 database

```
//SYSIN    DD *
  CREATE DATABASE MQVGDB
         BUFFERPOOL BP32K1
         STOGROUP MQVGSTG;
/*
```

Allthough you can use an existing database, we recommend that you define a separate database.

CSQ45CTS - create tablespaces

This job creates three tablespaces:

- MQ4KTBL is used for the administration tables and all object tables apart from OBJ_B_NAMELIST and OBJ_B_CHANNEL.
- MQ32KTBL is used only for OBJ_B_NAMELIST and OBJ_B_CHANNEL tables.
- MQCHTBL is used for channel initiator tables.

Tablespace MQ4KTBL is associated with bufferpool BP2, MQ32KTBL with bufferpool BP32K2 and MQCHTBL with bufferpool BP1. In addition the bufferpool also determines the page size of the table space.

The parameters PRIQTY and SECQTY are used for the primary and secondary allocation of the table space data sets, and are specified in kilobytes. For a detailed description of how to determine the space requirements for your tablespaces, refer to Chapter 16, “Planning your coupling facility and DB2 Environment”, in *WebSphere MQ for z/OS Concepts and Planning Guide*, GC34-6051.

Also be sure that the bufferpools and group bufferpools you use for WebSphere MQ tablespace are big enough.

For our test environment we use the input shown in Example 9-9.

Example 9-9 DB2 tablespaces

```
CREATE TABLESPACE MQ4KTBL  USING STOGROUP MQVGSTG
       PRIQTY 500          SECQTY 1000
       PCTFREE 20          SEGSIZE 64
       BUFFERPOOL BP2      LOCKSIZE ANY   CLOSE NO
       IN MQVGDB;
CREATE TABLESPACE MQ32KTBL USING STOGROUP MQVGSTG
       PRIQTY 500          SECQTY 1000
       BUFFERPOOL BP32K2   LOCKSIZE ANY   CLOSE NO
       IN MQVGDB;
CREATE TABLESPACE MQCHKTBL USING STOGROUP MQVGSTG
       PRIQTY 500          SECQTY 1000
       FREEPAGE 10         PCTFREE 30     SEGSIZE 64
       BUFFERPOOL BP1      LOCKSIZE ANY   CLOSE NO
       IN MQVGDB;
```

CSQ45CTB - create DB2 tables and associated indexes

This job creates all DB2 tables and their associated indexes. Changes of row names or attributes are not allowed.

CSQ45ATB - alter DB2 tables and associated indexes

This job is used when you migrate from MQSeries for OS/390 V5.2 to WebSphere MQ V5.3.

CSQ45BPL - bind DB2 plans

This job binds all necessary plans. Each program is associated with a plan name and a DBRM. All DBRMs are supplied in the SCSQDEFS library. All plan names in WebSphere MQ V5.3 have their names in CSQ5x530 format. Example 9-10 shows the DB2 bind input for plan CSQ5B530, which is used for the batch utility program CSQ5PQSG.

Example 9-10 Bind statements for plan CSQ5B530

```
BIND PLAN(CSQ5B530) -
```

```
MEM(CSQ5B530) -
ACQUIRE(USE) RELEASE(COMMIT) -
CURRENTDATA(NO) -
ACT(REP) RETAIN ISOLATION(CS) -
LIB('MQ530.SCSQDEFS')
```

CSQ45GEX - grant execute authority

This job grants execute authority to the plans that are bound in the job before. Different users have to be granted to different plans:

- The queue manager started task user ID needs access to the plans CSQ5A530, CSQ5C530, CSQ5D530, CSQ5K530, CSQ5L530, CSQ5R530, CSQ5S530, CSQ5T530, CSQ5U530 and CSQ5W530.
- The channel initiator address space needs access to CSQ5S530 and CSQ5K530.
- To execute the SDEFS function of the CSQUTIL batch utility, you need access to CSQ52530.
- CSQ5B530 is the plan associated with utility program CSQ5PQSG.
- CSQ5Z530 is associated with the utility CSQUZAP.

9.3.3 Define queue-sharing group and queue managers to DB2

The last step you need to do is to define the queue-sharing group and queue manager to DB2. To manage queue-sharing group and queue manager entries in DB2, WebSphere MQ provides the utility CSQ5PQSG. Example 9-11 provides a sample that adds a queue-sharing group MQVG, with the parameters DB7VU for the data-sharing group name and D7V1 for the local DB2 name. If the first step executes successfully, step 2 defines the queue manager MQV1.

Example 9-11 CSQ5PQSG sample for queue-sharing group MQVG and queue manager MQV1

```
//MQ000001 JOB
//ADDQSG   EXEC PGM=CSQ5PQSG,REGION=4M,
//              PARM='ADD QSG,MQVG,DB7VU,D7V1'
//STEPLIB  DD  DISP=SHR,DSN=MQ530.SCSQANLE
//         DD  DISP=SHR,DSN=MQ530.SCSQAUTH
//         DD  DISP=SHR,DSN=DB7V7.SDSNLOAD
//SYSPRINT DD  SYSOUT=*
//*
//ADDQMGR  EXEC PGM=CSQ5PQSG,REGION=4M,COND=(0,NE),
//              PARM='ADD QMGR,MQV1,MQVG,DB7VU,D7V1'
//STEPLIB  DD  DISP=SHR,DSN=MQ530.SCSQANLE
//         DD  DISP=SHR,DSN=MQ530.SCSQAUTH
//         DD  DISP=SHR,DSN=DB7V7.SDSNLOAD
```

```
//SYSPRINT DD  SYSOUT=*
```

This utility also creates the XCF group, when it creates the queue-sharing group DB2 table entry. The XCF group has the name CSQG*qsg*, where *qsg* is the queue-sharing group name.

If you create a queue manager entry, a member with the name of the queue manager is added to the XCF group of the queue-sharing group.

To display all XCF groups, you can use the command D XCF,GROUP. Figure 9-9 is the display of the XCF group for our queue-sharing group.

```
D XCF,GROUP,CSQGMQVG,ALL
IXC333I  22.08.21  DISPLAY XCF 525
   INFORMATION FOR GROUP CSQGMQVG
   MEMBER NAME:        SYSTEM:     JOB ID:    STATUS:
   MQV1                SC53        MQV1MSTR   ACTIVE
   MQV2                SC61        MQV2MSTR   ACTIVE
   MQV3                SC66        MQV3MSTR   ACTIVE
   MQV4                SC62        MQV4MSTR   ACTIVE

 INFORMATION FOR GROUP CSQGMQVG MEMBER MQV3

 MEMTOKEN: 0A00001C 00530003      ASID: 03EF

 SIGNALLING SERVICE
   MSGO ACCEPTED:         0  NOBUFFER:         0

   MSGI RECEIVED:         0  PENDINGQ:         0
   MSGI XFER CNT:         0  XFERTIME:       N/A

 GROUP SERVICE
   EVNT RECEIVED:         0  PENDINGQ:         0
```

Figure 9-9   Display of XCF group CSQGMQVG

In addition to the member names, this display also shows if the member is connected to the group and the job ID of the member. Information for the group member that is running on the system where you issued the command (in our case MQV3) is added as well.

To display the WebSphere MQ queue-sharing group, you can use the WebSphere MQ command DISPLAY GROUP. Sample output is shown in Figure 9-10.

```
-MQV1 DISPLAY GROUP
CSQ5100I -MQV1 DISPLAY GROUP report ...
CSQ5102I -MQV1 Queue managers in group MQVG
-----------------------------------------------------------------
 Name  Num  Prefix    Status      Ver  DB2   Connection
 ----  ---  --------  ----------  ---  ----  ----------
 MQV1    1  -MQV1     ACTIVE      530  D7V1  ACTIVE
 MQV2    2  -MQV2     ACTIVE      530  D7V2  ACTIVE
 MQV3    3  -MQV3     ACTIVE      530  D7V3  ACTIVE
 MQV4    4  -MQV4     ACTIVE      530  D7V4  ACTIVE
-----------------------------------------------------------------
 End of queue managers report
CSQ9022I -MQV1 CSQ5DQSG ' DISPLAY GROUP' NORMAL COMPLETION
```

Figure 9-10 WebSphere MQ DISPLAY GROUP output

9.4 Create bootstrap and log data sets

Program CSQJU003 prepares the bootstrap and log data sets WebSphere MQ needs for tracking and logging of data. Each queue manager needs at least one BSDS and three log data sets. For our test system, we defined two BSDS and four log data sets for each queue manager using the sample JCL that is provided in thlqual.SCSQPROC(CSQ4BSDS). And, because it is a test system, we decided to use small logs and BSDSs. In a production environment, the sizes should be much larger. Figure 9-11 on page 179 contains a sample for the primary BSDS and one log data set LOGCOPY1.DS01.

```
DEFINE CLUSTER (NAME(MQV1.BSDS01)                  -
                     VOLUMES(TOTMQF)               -
                     SHAREOPTIONS(2 3) )           -
                DATA(RECORDS(60 60)                -
                     RECORDSIZE(4089 4089)         -
                     CONTROLINTERVALSIZE(4096)     -
                     FREESPACE(0 20)               -
                     KEYS(4 0) )                   -
               INDEX(RECORDS(5 5)                  -
                     CONTROLINTERVALSIZE(1024) )
DEFINE CLUSTER (NAME(MQV1.LOGCOPY1.DS01)           -
                     LINEAR                        -
                     VOLUMES(TOTMQF)               -
                     SHAREOPTIONS(2 3)             -
                     RECORDS(18000) )
```

Figure 9-11 Sample BSDS (BSDS01) and log data set definition for LOGCOPY1

When you plan your production system, you should follow the recommendations in the *WebSphere MQ Concepts and Planning Guide,* GC34-6051 for the size of your BSDS and log data sets.

9.4.1 SHAREOPTIONS parameter

Make sure that you define your log data sets with SHAREOPTIONS(2 3) or higher. Otherwise, recovery operations in a data-sharing environment will not be permitted. If you have not allocated your active logs with SHAREOPTIONS(2 3) or higher, you will get the message shown in Example 9-12.

Example 9-12 Messages for active log with shareoptions lower than 2 3

```
CSQJ250I -MQGV CSQJDS02 DATA-SHARING REQUIRES ALL ACTIVE LOG DATA SETS TO HAVE
SHAREOPTIONS(2 3) OR HIGHER
CSQJ251I -MQGV CSQJDS02 SHAREOPTIONS FOR DATA SET. MQGV.LOGCOPY1.DS01 ARE TOO
LOW FOR DATA-SHARING
```

The message text for CSQJ250I in *WebSphere MQ for z/OS Messages and Codes Version 5.3, GC34-6056,* is not correct. The message was added late in the development cycle when there was no longer a chance to change the books. The explanation in the book is:

> An active log data set was detected with share options that do not permit recovery operations in a data-sharing environment. All active log data sets must have SHAREOPTIONS(2 3) at least.

And it should be:

> An active log data set was detected with share options that do not permit CF Structure recovery in a queue-sharing group environment. All active log data sets must have SHAREOPTIONS(2 3) at least to allow CF structure recovery. This can occur when the queue manager's own log data sets are checked during startup, or when a RECOVER CFSTRUCT command is issued that requires access to another queue manager's log data sets.

The manual says that the system will then issue message CSQJ251E and terminate. This is also not correct. What really happens is that execution continues, but the RECOVER CFSTRUCT command fails if you use it.

The message does not happen for a V5.2 queue manager because it does not check its own log SHAREOPTIONS at startup, because its logs will never be required for structure recovery. V5.2 cannot perform a RECOVER CFSTRUCT command, nor can it do recoverable updates to a recoverable CFSTRUCT, since it cannot even connect to a CFLEVEL(3) structure.

When you ignore these messages at startup, and then have to do a RECOVER CFSTRUCT which fails with these messages, you will have the following choice:

- Attempt to do the RECOVER CFSTRUCT command on the queue manager whose log's SHAREOPTIONS are too low (because it can read its own logs anyway, and there's a chance that the other queue managers in the group are correct) or
- Stop the queue manager with the wrong SHAREOPTIONS, alter the SHAREOPTIONS and then perform the RECOVER CFSTRUCT in parallel with restarting the queue manager.

9.5 Define page sets

A page set is a linear VSAM data set (LDS) that has been specially formatted to be used by WebSphere MQ. Page sets are used to store most messages and object definitions. The exceptions to this are global definitions, which are stored in a shared repository on DB2, and the messages on shared queues. These are not stored on page sets.

Page sets must be allocated and formatted. Sample JCL and control statements to allocate and format page sets can be found in thlqual.SCSQPROC(CSQ4PAGE). We show the allocation and format of page sets 0 and 1 of our queue manager MQV1 in Figure 9-12 on page 181.

Because page sets are needed to store object definitions and messages, we have to define page sets for every queue manager even if we are only using shared queues.

- We defined five page sets for each of our queue managers.
- For a test system it is not important to have them defined on different volumes, but in a production environment it is recommended that page sets be distributed over different volumes. Also make sure that your page sets are on different volumes from your active log data sets.
- To make it easier to identify which page set belongs to which queue manager, we included the name of the queue manager in the high-level qualifier.

```
//PAGESET  JOB
//DEFINE   EXEC PGM=IDCAMS,REGION=4M
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
   DEFINE CLUSTER (NAME(MQV1.PSID00)     -
           RECORDS(1000 500)             -
           LINEAR                        -
           VOLUMES(TOTMQF)               -
           SHAREOPTIONS(2 3) )
   DEFINE CLUSTER (NAME(MQV1.PSID01)     -
           RECORDS(1000 500)             -
           LINEAR                        -
           VOLUMES(TOTMQG)               -
           SHAREOPTIONS(2 3) )
/*
//FORM     EXEC PGM=CSQUTIL,COND=(0,NE)
//STEPLIB  DD  DISP=SHR,DSN=MQ530.SCSQANLE
//         DD  DISP=SHR,DSN=MQ530.SCSQAUTH
//SYSPRINT DD  SYSOUT=*
//CSQP0000 DD  DISP=OLD,DSN=MQV1.PSID00
//CSQP0001 DD  DISP=OLD,DSN=MQV1.PSID01
//SYSIN    DD  *
  FORMAT
/*
```

Figure 9-12 Page set allocation and formatting

If you want to know more about page sets, refer to *WebSphere MQ Concepts and Planning Guide* V5.3, GC34-6051.

9.6 System parameter module

Because the default values for the parameters in the macros CSQ6SYSP, CSQ6LOGP and CSQ6ARVP are the recommended production values, we changed some definitions to meet the test environment requirements. And we had to add some definitions to be able to run in a queue-sharing group. A sample JCL to make or change the system parameter module is provided in thlqual.SCSQPROC(CSQ4ZPRM). In the following sections, we list the parameters that we changed to meet our needs.

9.6.1 CSQ6LOGP

CSQ6LOGP establishes your logging options. The parameters DEALLCT, MAXARCH, MAXTRU and WRTHRSH can be changed with the SET LOG command while your queue manager is still running.

- OFFLOAD = NO

  Archive logging is not required in a test environment but strongly recommended for a production system.

- TWOACTV = NO

  For a test system, single active logs are sufficient, but in a production environment you should use dual active logging. For more information about the use of single and dual logging, refer to *WebSphere MQ for z/OS Concepts and Planning Guide,* GC34-6051.

- TWOARCH = NO

  We switched dual archive off because we don't use archive logging. In a production system with dual active logging, you should also use dual archive logging.

- TWOBSDS = NO

  To run in single BSDS mode is OK for a test system, but to minimize the risk of running into problems during restart, you should choose to run in dual mode, where both BSDSs record the same information. For more information about bootstrap data sets and what they are for, refer to WebSphere MQ for z/OS *Concepts and Planning Guide*, GC34-6051.

- WRTHRSH = 15

  For some reason the default value for WRTHRSH parameter is still 20 although the recommended value for both test and production is 15. WRTHRSH specifies the number of 4 KB output buffers to be filled before they are written to active log data sets. 15 is the optimum value because it corresponds to the maximum number of buffers written in a single I/O.

9.6.2 CSQ6ARVP

CSQ6ARVP establishes the archiving environment. Because we do not use archiving in our test installation, as per the CSQ6LOGP settings in 9.6.1, “CSQ6LOGP” on page 182, we do not need to modify any of the parameters. They are ignored.

If you use archiving and you need to change any of the values within CSQ4ARVP, you can do this using the SET ARCHIVE command. Planning your archive storage is discussed in the *WebSphere MQ for z/OS Concepts and Planning Guide,* GC34-6051. For more information regarding the SET ARCHIVE command, refer to *WebSphere MQ Script (MQSC) Command Reference,* SC34-6055.

9.6.3 CSQ6SYSP

Macro CSQ6SYSP sets the system parameters. You can set the parameters CTHREAD, IDBACK, IDFORE, LOGLOAD, STATIME and TRACTBL dynamically using the WebSphere MQ SET SYSTEM command while your queue manager is still running.

CSQ6SYSP contains the important parameter that is required to run a queue manager in a queue-sharing group, the queue-sharing group data QSGDATA.

For one of our queue managers we coded QSGDATA as follows:

```
QSGDATA=(MQVG,DB7VU,D7V1,4)
```

where:

- `MQVG` is the queue-sharing group to which the queue manager belongs.
- `DB7VU` is the data-sharing group to which the queue manager is to connect.
- `D7V1` is the local DB2 subsystem to which the queue manager is to connect. We later changed this parameter to the D7VG group attach name to allow an ARM restart on a different image.
- `4` is the default value of DB server tasks used for accessing DB2.

You will find more information about QSGDATA in 8.4, “System parameters” on page 141.

Note: If you change any of the parameters dynamically using the SET command while your queue manager is still running, do not forget to either add the command to a member in the CSQINP2 concatenation of your queue managers started task procedure or change the system parameter module and replace the old one. If you do neither, then your changes will get lost at the next restart of your queue manager.

9.7 Channel initiator parameter module

The channel initiator parameter module, which controls how distributed queuing operates, contains only one macro: CSQ6CHIP.

ADOPTMCA=YES and ADOPTCHK=ALL

We chose these because, when a channel suffers a communications failure, the receiver channel might be left in a “comms receive” state. When communication is re-established and the sender channel attempts to reconnect, the remote queue manager finds that the receiver channel is already running and will not allow another version of the same receiver channel to be started. The message `CSQX514E Channel` *`channel-name`* `is active on` *`queue manager name`* is logged to document this.

With ADOPTMCA=YES, you allow WebSphere MQ to stop a receiver channel automatically and to start a new one in its place when a request to start a duplicate receiver channel is received.

With ADOPTCHK, you specify which options to check before the channel is stopped and restarted.

- ADOPTCHK=NONE means that WebSphere MQ will stop the existing receiver channel and start a new one if the channel name of the new channel is the same as that of the existing channel, regardless of the queue manager name or network address.
- ADOPTCHK=NETADDR means thatWebSphere MQ will stop the existing receiver channel and start a new one if both the channel name and the network address of the new channel are the same as those of the existing channel, regardless of the queue manager name.
- ADOPTCHK=QMNAME means that WebSphere MQ will stop the existing receiver channel and start a new one if both the channel name and the queue manager name of the new channel are the same as those of the existing channel, regardless of the network address.
- ADOPTCHK=ALL means that WebSphere MQ will cancel the existing receiver channel and start a new one only if the channel name, queue

manager name, and network address of the new channel are all the same as those of the existing channel.

If you are in a situation where the network address might change (for example, when using DHCP) use ADOPTCHK=NONE or ADOPTCHK=QMNAME to check the new receiver channel against the old receiver channel.

It is possible for two or more receiver channels with the same name to be running at the same time. AdoptMCA does not change this. The purpose of AdoptMCA is to stop an orphaned instance of a channel automatically when it WebSphere MQ determines that the new instance is a duplicate.

If you specify ADOPTCHK=NONE and an attempt to start a second receiver channel with the same name is made, WebSphere MQ will stop the old receiver channel no matter where the request to start the new channel comes from. You should not use the ADPOTCHK=NONE option if the same channel name is used on more than one queue manager.

DNSWLM = YES

With this parameter we specify that the TCP listener that handles inbound transmissions for the queue-sharing group should register with Workload Manager for Dynamic Domain Name System.

DNSGROUP

This specifies the name of the group the TCP listener, that handles inbound transmissions for the queue-sharing group, should join when using Workload Manager Dynamic Domain Name System support. Because we did not specify a name here but coded DNSWLM=YES, the name of the queue-sharing group is used.

TCPKEEP = YES

With this parameter we specify that the TCP KeepAlive facility, as specified by the KEEPALIVEOPTIONS statement in the TCP profile configuration data set, is to be used. The interval is specified by the channel attribute KeepAliveInterval (KAINT), which is passed to the communications stack for KeepAlive timing for the channel.

For information about the KeepAliveInterval parameter, refer to *WebSphere MQ Script (MQSC) Command Reference,* SC34-6055.

TCPNAME = TCPIPMVS

TCPIPMVS is the TCP/IP subsystem name that is used in our application queue managers.

9.8 WebSphere MQ queue-sharing group objects

Now that all basic definitions that are required to start the queue managers are done, we have to decide which queues, channels, and processes we need to define for our queue-sharing group MQVG.

Shared receiver channels

We need shared receiver channels for messages that will be sent to group MQVG from the WebSphere MQ for Windows queue managers. Shared channels must be defined with QSGDISP=GROUP as shown in Example 9-13.

Example 9-13 Define a receiver channel

```
DEFINE CHANNEL('M23WPM44.TO.MQVG') CHLTYPE(RCVR) QSGDISP(GROUP) TRPTYPE(TCP)
```

After a successful definition there will be a definition in the shared repository, and the command outlined in Example 9-14 is created by WebSphere MQ and sent to all active queue managers in the group to make a local copy of the channel on page set zero.

Example 9-14 Define with QSGDISP(COPY)

```
DEFINE CHANNEL ('M23WPM44.TO.MQVG') CHLTYPE(RCVR) QSGDISP(COPY) TRPTYPE(TCP)
```

Shared sender channels

We need shared sender channels to send messages from the queue-sharing group to the remote queue managers. Again, this has to be defined with QSGDISP=GROUP. And as with the receiver channels, a define command with QSGDISP=COPY is created to distribute the definition to page set zero of all queue managers in the group after the definition of the channel was successful in the shared repository.

When we look at the definitions for the channels by using the operator command screens with the parameters Disposition=Group, Connect name=local queue manager, Target queue manager=local queue manager and Action queue manager=* as shown in Figure 9-13 on page 187. All definitions in the shared repository will be displayed (see Figure 9-14 on page 187).

The same display with Disposition=Copy will show the local definitions on page set zero of every queue manager in the queue-sharing group. Figure 9-15 on page 188 shows such a display.

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

Complete fields. Then press Enter.

Action . . . . . . . . . . 1     1. Display       4. Manage    6. Start
                                 2. Define like   5. Perform   7. Stop
                                 3. Alter

Object type . . . . . . . . CHANNEL      +
Name . . . . . . . . . . . *
Disposition . . . . . . . . G  Q=Qmgr,C=Copy,P=Private,
                               G=Group,S=Shared,A=All

Connect name . . . . . . . MQV2  - local queue manager or group
Target queue manager . . . MQV2
           - connected or remote queue manager for command input
Action queue manager . . . *     - command scope in group
Response wait time . . . . 30    5 - 999 seconds

(C) Copyright IBM Corporation 1993,2002. All rights reserved.
```

Figure 9-13   WebSphere MQ operator screen - Main Menu

```
                       List Channels - MQV2                  Row 1 of 6

Type action codes. Then press Enter.
 1=Display   2=Define like   3=Alter   4=Manage   5=Perform
 6=Start     7=Stop

    Name                    Type          Disposition   Status
<>  *                       CHANNEL       GROUP   MQVG
    MQVG.M23WPF31           SENDER        GROUP
    MQVG.M23WPH767          SENDER        GROUP
    MQVG.M23WPM44           SENDER        GROUP
    M23WPF31.TO.MQVG        RECEIVER      GROUP
    M23WPH767.TO.MQVG       RECEIVER      GROUP
    M23WPM44.TO.MQVG        RECEIVER      GROUP
                  ******** End of list ********
```

Figure 9-14   Sender and receiver channels in the shared repository

```
                             List Channels - MQV2                  Row 9 of 24

Type action codes. Then press Enter.
 1=Display   2=Define like   3=Alter   4=Manage   5=Perform
 6=Start     7=Stop

    Name                     Type          Disposition   Status
<>  *                        CHANNEL       COPY    MQVG
    MQVG.M23WPM44            SENDER        COPY    MQV1  INACTIVE
    MQVG.M23WPM44            SENDER        COPY    MQV2  INACTIVE
    MQVG.M23WPM44            SENDER        COPY    MQV3  INACTIVE
    MQVG.M23WPM44            SENDER        COPY    MQV4  RETRY
    M23WPF31.TO.MQVG         RECEIVER      COPY    MQV1  INACTIVE
    M23WPF31.TO.MQVG         RECEIVER      COPY    MQV2  RUN
    M23WPF31.TO.MQVG         RECEIVER      COPY    MQV3  INACTIVE
    M23WPF31.TO.MQVG         RECEIVER      COPY    MQV4  INACTIVE
    M23WPH767.TO.MQVG        RECEIVER      COPY    MQV1  STOP
    M23WPH767.TO.MQVG        RECEIVER      COPY    MQV2  STOP
    M23WPH767.TO.MQVG        RECEIVER      COPY    MQV3  STOP
    M23WPH767.TO.MQVG        RECEIVER      COPY    MQV4  STOP
    M23WPM44.TO.MQVG         RECEIVER      COPY    MQV1  INACTIVE
    M23WPM44.TO.MQVG         RECEIVER      COPY    MQV2  INACTIVE
    M23WPM44.TO.MQVG         RECEIVER      COPY    MQV3  INACTIVE
    M23WPM44.TO.MQVG         RECEIVER      COPY    MQV4  RUN
                        ******** End of list ********
```

Figure 9-15 Local copies of sender and receiver channels on page set zero

Group listener

To be able to receive messages through the shared receiver channels we need to start a listener that listens on a generic port. In our configuration, generic port 1501 is made available through the Sysplex Distributor technology as described in Chapter 5, “Sysplex Distributor” on page 87. To start the group listener, you either manually enter the command START LISTENER INDISP(GROUP) PORT(1501)or use the operator command screen. Because we wanted the listener to start automatically, we added the command to the CSQ4INPX member within the CSQINP2 concatenation of the started task procedure of the queue manager together with the START CHINIT command.

Shared transmission queue

To send messages to the remote queue manager through the shared sender channel, we defined a shared transmission queue. Example 9-15 on page 189 shows a command that creates an object definition of a shared transmission queue in the shared repository.

Example 9-15 Sample shared transmission queue definition

```
DEFINE QLOCAL('m23wph767b') QSGDISP(SHARED) CFSTRUCT('APPLICATION1')
MAXMSGL(64512) USAGE(XMITQ)
```

Object definitions for the CICS application ECHO

We connected two CICS regions to the queue-sharing group. We wanted the CKTI transaction to monitor a shared initiation queue that gets the trigger messages from a shared application queue. The DEFINE command is shown in Example 9-16. The advantage of a shared application queue is that even if one queue manager or one CICS region is down, the messages are still processed.

Example 9-16 Sample shared queue definition

```
DEFINE QLOCAL('CICS.MQVG.ECHO01') QSGDISP(SHARED) CFSTRUCT('APPLICATION1')
PROCESS('CICS.MQVG.ECHO01.PROCESS') MAXMSGL(64512) TRIGTYPE(EVERY)
TRIGDATA('WORK') INITQ('CICS.MQVG.INITQ01')
```

Note: If you try to define a shared queue like a default local queue, do not forget to change the maximum message length (MAXMSGL) to 64512. Otherwise the define will fail with the following messages:

```
CSQM092I  CSQMAQLC MAXMSGL(4194304) VALUE RANGE ERROR
CSQM090E CSQMAQLC FAILURE REASON CODE X'00D44004' CSQ9023E
CSQMAQLC ' DEFINE QLOCAL' ABNORMAL COMPLETION
```

because you cannot define a shared queue with a MAXMSGL bigger than 64512, due to the coupling facility list stucture limitations.

Next we defined the shared initiation queue for the trigger messages generated for the just-defined shared queue CICS.MQVG.ECHO01. With a shared initiation queue, we can make sure that the messages will be processed even if one of our queue managers or one of our two CICS regions, which are connected to WebSphere MQ by the CICS adapter, fail.

And we need a process definition with the necessary information for the CICS trigger monitor CKTI. Example 9-17 shows the define command.

Example 9-17 Sample process definition

```
PROCESS('CICS.MQVG.ECHO01.PROCESS') QSGDISP(GROUP) DESCR('ECHO 01 APPLICATION
PROCESS') APPLTYPE(CICS) APPLICID('ECHO START')
```

9.9 The test system

After we have made the definitions described above, we had a queue-sharing group named MQVG as shown in Figure 9-16 on page 191.

The two queue managers MQV2 and MQV4 are defined to the queue-sharing group as the gateway queue managers. They have the group listener started for the generic port 1501. Because we are using the Sysplex Distributor to provide high availability and workload management, our queue-sharing group will still be operational even if one of the gateway queue managers fails. Between the gateway queue managers and the remote queue managers, sender and receiver channels are defined (QSGDISP=SHARED on the gateway). MQV2 and MQV4 will process all incoming requests from the remote queue managers for the queue-sharing group and forward all messages from the shared transmission queues to the remote queue managers.

Application queue managers MQV1 and MQV3 both have a CICS connected using the CICS adapter. In both CICS regions, one instance of the task initiator transaction (CKTI) is monitoring the same shared initiation queue. On this shared initiation queue, trigger messages are put generated by a shared triggered application queue.

9.9.1 Sample message flow 1

Here we show the message flow from the remote queue manager through the gateway queue managers to the application queue managers where it will be processed by CICS.

1. The remote queue manager sends a connection request to the Dynamic VIPA address 9.12.8.10. Because we are using the Sysplex Distributor, and LPAR SC61 with queue manager MQV2 is the LPAR that owns the DVIPA, the connection request will first be sent to SC61. The Sysplex Distributor on SC61 queries WLM to select the best target.
2. Let us assume that the Sysplex Distributor selects SC62 as the best target. The remote queue manager is physically connected to MQV4 MQV4 processes the messages for the queue-sharing group.
3. For every message that is put to the shared application queue, a trigger message is written to the shared initiation queue (we defined the queue with TRIGTYPE=EVERY) containing the information provided with the associated process.
4. One of the CKTI transactions will get the trigger message from the initiation queue and because we defined APPLICID(ECHO START), the ECHO transaction will be started and get the messages from the application queue.

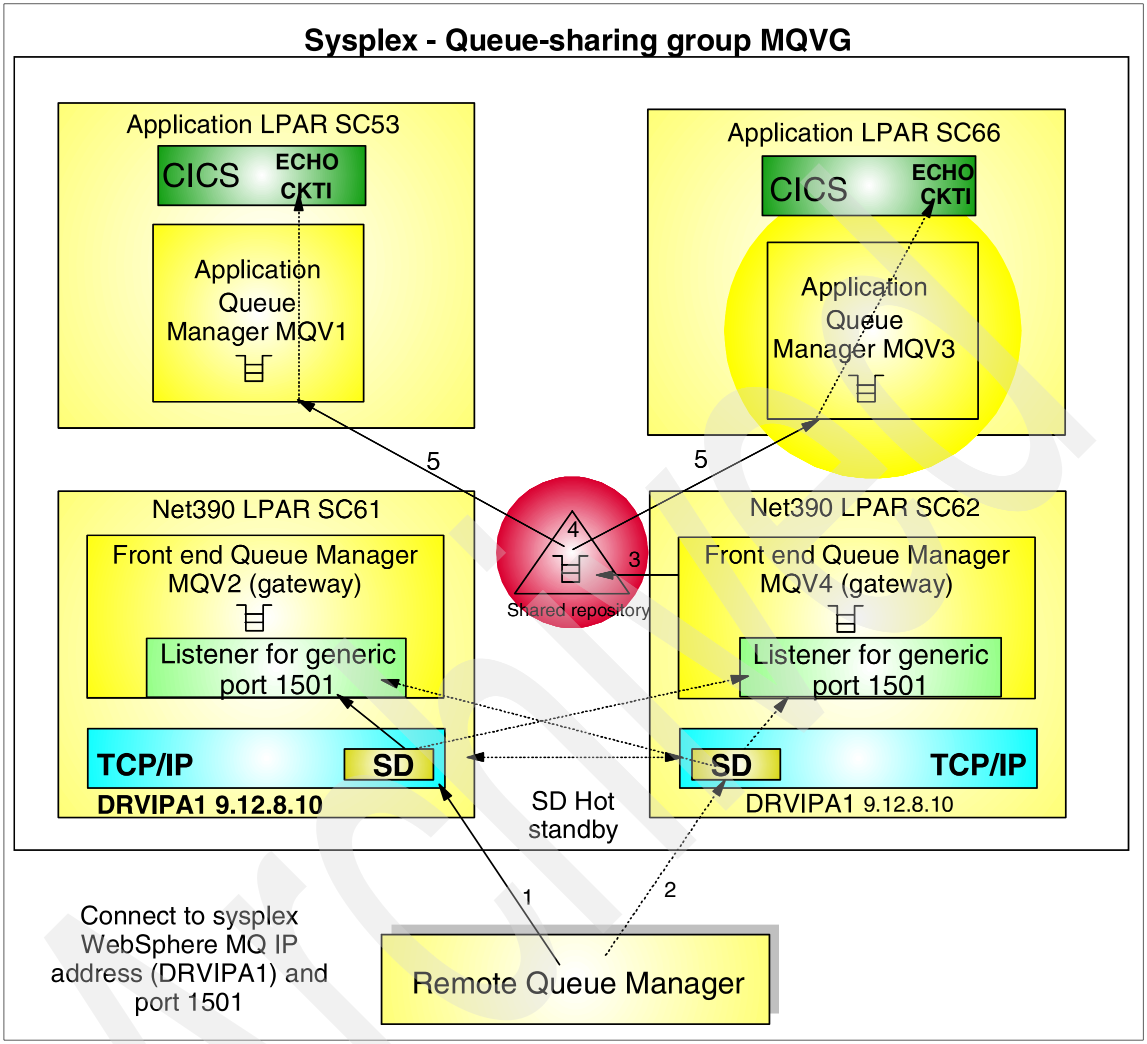


Figure 9-16 Message flow to a CICS transaction through the queue-sharing group

9.9.2 Sample message flow 2

In Figure 9-17 on page 192 we show how a reply message is sent back from the CICS transaction to the remote queue manager using the shared transmission queue and the shared sender channel.

1. The ECHO transaction from the CICS region connected to MQV3 writes a message to the shared transmission queue due to the reply-to queue information it got from the request message.

2. Now the channel initiator starts the shared sender channel that fits best after the channel load of both gateway queue managers have been compared. In our sample the channel is started by MQV4.
3. The message is put to the reply-to queue of the remote queue manager.

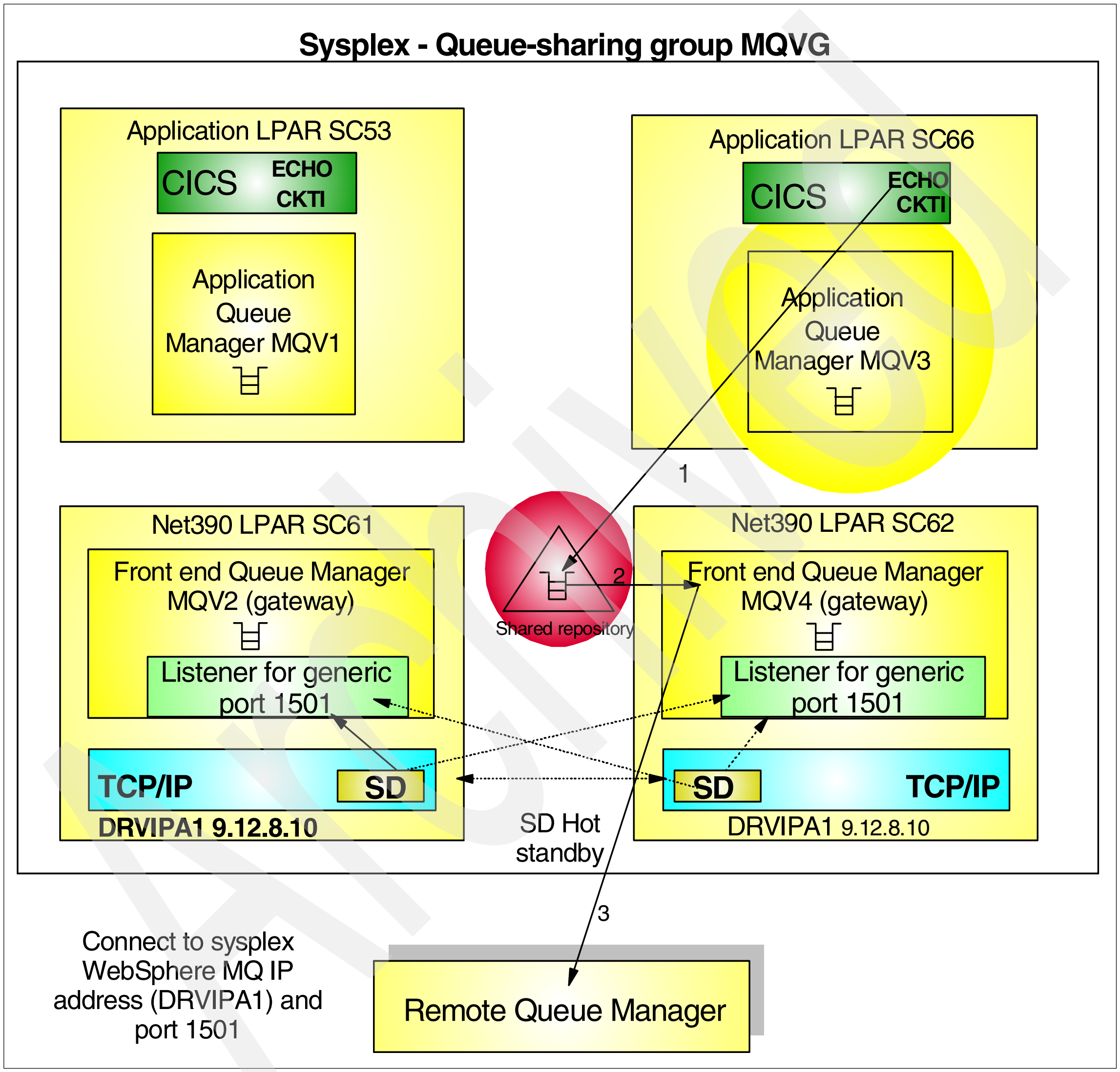


Figure 9-17 Message flow from application LPAR to remote queue manager

10

Migration to WebSphere MQ for z/OS Version 5.3

In this chapter, we take a closer look at the actions you need to take to migrate an existing queue manager using MQSeries for OS/390 V5.2 to the new version of the product, WebSphere MQ for z/OS V5.3. Because the focus of this residency on Parallel Sysplex environment, the MQSeries V5.2 queue manager is part of a WebSphere MQ queue sharing and DB2 data-sharing group. Therefore, we document changes to the WebSphere MQ/DB2 data-sharing group necessary for a successful migration to WebSphere MQ for z/OS V5.3.

We also provide an overview of migrating from MQSeries for OS/390 V2.1 to WebSphere MQ V5.3.

For further information relating to WebSphere MQ/DB2 configuration, refer to Chapter 9, “Creating a shared queue environment” on page 159 and to “Task 8: Set up the DB2 environment” in *WebSphere MQ for z/OS System Setup Guide,* SC34-6052.

For further information relating to DB2 administration, refer to *DB2 UDB for OS/390 and z/OS V7 Administration Guide*, SC26-9931.

10.1 Overview

When you migrate from MQSeries for OS/390 V5.2, you can reuse the existing subsystems with page sets, log data sets, object definitions, initialization data sets and existing queues with WebSphere MQ for z/OS V5.3.

- If your installation is already at MQSeries for OS/390 V5.2 and has implemented a Parallel Sysplex configuration with shared queues being used, then there is not that much to be done. We have outlined the steps in 10.2, “MQSeries V5.2 to WebSphere MQ V5.3 migration” on page 194.
- If your system is a Parallel Sysplex, yet WebSphere MQ has not been configured to use shared queues and is using stand-alone queue managers, it is even easier to migrate your installation to WebSphere MQ V5.3 and you should proceed to 10.3.1, “New system parameters” on page 205.
- If your installation is a Parallel Sysplex and you would like to configure WebSphere MQ to use shared queuing, you can refer to Chapter 9, “Creating a shared queue environment” on page 159 or Chapter 2, “Customizing your queue managers” in *WebSphere MQ for z/OS System Setup Guide*, SC34-6052, for detailed implementation steps.

Probably the most important changes to be made to your system are to replace the early code in the linklist and to adapt the JCL of your started tasks to point to the new product libraries. You can, for example, still use your current parameter modules for both the queue manager and the channel initiator, making this quite an easy migration.

10.2 MQSeries V5.2 to WebSphere MQ V5.3 migration

In this section, we explain the procedures we used to migrate an MQ Parallel Sysplex environment from MQSeries for OS/390 V5.2 to WebSphere MQ for z/OS V5.3.

The procedures used are as follows:

1. Migration to WebSphere MQ for z/OS V5.3 without queue sharing or DB2 data-sharing groups implemented.
2. Migration to WebSphere MQ for z/OS V5.3 with queue sharing and DB2 data-sharing groups implemented.

10.2.1 Migration with no queue sharing implementation

The migration procedure from MQSeries V5.2 to WebSphere MQ for z/OS V5.3 is quite easy if your installation is not configured to use queue-sharing groups. Since the focus of this redbook is on the Parallel Sysplex environment, we only briefly describe this procedure. For more details, refer to Chapter 3 of *WebSphere MQ for z/OS System Setup Guide*, SC34-6052.

Software levels

The minimum levels for some of the prerequisite software for WebSphere MQ have changed. Ensure that you have the correct levels of the following prerequisite software products:

- OS/390 V2.9 or higher
- DFSMS/DFP
- High Level Assembler
- ISPF
- JES
- DB2 (Version 6 or higher)
- Language Environment/370
- Security Server (RACF)
- SMP/E
- TSO/E
- Communications Server

For further detailed information, refer to *WebSphere MQ for z/OS Concepts and Planning Guide,* GC34-6051.

Refresh early code

It will be necessary to deliver the new WebSphere MQ early code to your LPAR using SMP/E and perform an IPL of your system during a normal maintenance window, thus reducing the requirements for an IPL of the system for the migration to V5.3 for your queue manager. Once the IPL has been performed and the new early code is stored within the LLA memory of your system, it is necessary to change your WebSphere MQ subsystem procedures to point to the new WebSphere MQ V5.3 libraries. Once the MSTR and CHIN procedures have been updated with references to the new WebSphere MQ V5.3 code, you can restart your queue manager and channel initiator.

For further detailed information on refreshing the early code, refer to 14.4.1, “Refresh the early code” on page 283 in this redbook.

System parameters

The system parameter QINDXBLD is introduced with WebSphere MQ for z/OS V5.3. This parameter is used to determine whether a queue manager restart waits until all queue manager indexes are rebuilt or completes before all indexes are rebuilt.

The parameter has the following options:

- WAIT

  Private queue indexes are built sequentially as in MQSeries V5.2 and all applications must wait for indexes to be rebuilt before it is possible to connect to the queue manager.

- NOWAIT

  Queue manager restart can complete before all queue index building is completed. If an application attempts to use an indexed queue, it will have to wait for the queue's index to be rebuilt.

If during your migration you will be introducing QINDXBLD into your system parameter module, it will be necessary to re-linkedit your system parameter and perform a restart of your queue manager.

If your system is using a Parallel Sysplex and you would like to configure your new WebSphere MQ V5.3 to use queue sharing and DB2 data-sharing groups, take a look at Chapter 9, "Creating a shared queue environment" on page 159.

Fallback to MQSeries V5.2

Backward migration can be achieved by changing library concatenations that reference the V5.3 product to pick up the V5.2 product libraries that contain the WebSphere coexistence PTF code and restart the queue manager.

10.2.2 Migration with queue sharing

If your environment is configured to use queue sharing and DB2 data-sharing groups in MQSeries V5.2, the migration procedure is much more involved. It will require several steps to alter your DB2 environment to allow for upgrade of your queue-sharing group environment.

PTF levels

To migrate your MQSeries for OS/390 V5.2 queue managers to WebSphere MQ for z/OS V5.3, you have to apply the following maintenance:

- PTF UQ49206 to APAR PQ43381

  This PTF addresses various problems regarding stability of the product.

- PTFs UQ68087 and UQ68088 to APAR PQ62235

  These PTFs enables queue managers currently running at V5.2 and V5.3 levels to coexist within a Parallel Sysplex environment.

The PTFs once applied will allow for the following:

- Forward migration from MQSeries for OS/390 V5.2 to WebSphere MQ V5.3 for z/OS.
- Coexistence of mixed queue manager levels within a queue-sharing group.
- Backward migration from WebSphere MQ for z/OS V5.3 to MQSeries for OS/390 V5.2.

To verify that the PTFs have been installed, you can use the SMP/E tool that is used by most systems programmers to install code on your OS/390 and z/OS LPARs. Because a majority of systems use SMP/E, we do not explain the process. If you need a reference, use *OS/390 V2R7.0 SMP/E User's Guide,* SC28-1740.

There is also a quick procedure that can be used to verify that the PTF has been applied and installed. It is to run the SQL Select statements shown in Example 10-1, which generates the output shown in Figure 10-1 on page 198.

Example 10-1 SQL SELECT from system tables to check ADMIN_B_SCST and OBJ_QUEUE indexes

```
SELECT * FROM SYSIBM.SYSCOLUMNS WHERE TBNAME='ADMIN_B_SCST' ;
SELECT * FROM SYSIBM.SYSINDEXES WHERE NAME LIKE 'OBJ_QUEUE%' ;
```

From this output it is necessary to check that the channel token row exists, if the channel token exists. For the table OBJ_QUEUE, the two indexes OBJ_QUEUE_IX1 and OBJ_QUEUE_IX2 must exist. If both conditions are true, then the PTF has been installed with the hold actions applied and you can proceed with the installation of the WebSphere MQ V5.2/V5.3 coexistence PTF.

```
---------+---------+---------+---------+---------+---------+---------+-----
  SELECT * FROM SYSIBM.SYSCOLUMNS WHERE TBNAME='ADMIN_B_SCST' ;        000
---------+---------+---------+---------+---------+---------+---------+-----
NAME                TBNAME              TBCREATOR   COLNO  COLTYPE   LENGTH
---------+---------+---------+---------+---------+---------+---------+-----
CHLCRESTAMP         ADMIN_B_SCST        CSQ             1  CHAR          10
REMOTEQMGR          ADMIN_B_SCST        CSQ             5  CHAR          48
CHLNAME             ADMIN_B_SCST        CSQ             4  CHAR          20
XMITQ               ADMIN_B_SCST        CSQ             3  CHAR          48
CHLUPDSTAMP         ADMIN_B_SCST        CSQ             2  CHAR          10
QSGNAME             ADMIN_B_SCST        CSQ             6  CHAR           4
LONGRETRYCOUNT      ADMIN_B_SCST        CSQ            12  INTEGER        4
SHORTRETRYCOUNT     ADMIN_B_SCST        CSQ            13  INTEGER        4
OWNINGQMGR          ADMIN_B_SCST        CSQ             7  CHAR           4
CHANNELSTATUS       ADMIN_B_SCST        CSQ            10  CHAR           1
REMOTEMACH          ADMIN_B_SCST        CSQ             9  CHAR          20
CHLTOKEN            ADMIN_B_SCST        CSQ             8  CHAR          20
CHANNELTYPE         ADMIN_B_SCST        CSQ            11  INTEGER        4
NEXTRETRYTIME       ADMIN_B_SCST        CSQ            14  INTEGER        4
DATA                ADMIN_B_SCST        CSQ            15  CHAR           4
UPDT_QMGR           ADMIN_B_SCST        CSQ            16  CHAR          48
UPDT_QMGRNUM        ADMIN_B_SCST        CSQ            17  SMALLINT       2
UPDT_STAMP          ADMIN_B_SCST        CSQ            18  CHAR           8
CREATE_QMGR         ADMIN_B_SCST        CSQ            19  CHAR          48
CREATE_QMGRNUM      ADMIN_B_SCST        CSQ            20  SMALLINT       2
CREATE_STAMP        ADMIN_B_SCST        CSQ            21  CHAR           8
DSNE610I NUMBER OF ROWS DISPLAYED IS 21
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+-----
  SELECT * FROM SYSIBM.SYSINDEXES WHERE NAME LIKE 'OBJ_QUEUE%' ;       000
---------+---------+---------+---------+---------+---------+---------+-----
NAME                CREATOR   TBNAME              TBCREATOR  UNIQUERULE  CO
---------+---------+---------+---------+---------+---------+---------+-----
OBJ_QUEUE_IX1       CSQ       OBJ_B_QUEUE         CSQ        P
OBJ_QUEUE_IX2       CSQ       OBJ_B_QUEUE         CSQ        D
DSNE610I NUMBER OF ROWS DISPLAYED IS 2
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
```

Figure 10-1 Output from Example 10-1 on page 197

If the PTF has not been installed or the hold actions not performed, it is necessary to order PTF UQ49206 from IBM. This is to be done before proceeding with the install of the MQSeries V5.2 coexistence PTF or the migration to WebSphere MQ V5.3.

During this migration, we encountered referential constraints that prevented us from dropping the index CSQ.OBJ_QUEUE and we received the error messages shown in Example 10-2.

Example 10-2 DB2 SQLCODE -669

```
      DROP INDEX CSQ.OBJ_QUEUE
 SQL ERROR DURING EXECUTE IMMEDIATE
DSNT408I SQLCODE = -669, ERROR:  THE OBJECT CANNOT BE EXPLICITLY DROPPED.
REASON
DSNT418I SQLSTATE   = 42917 SQLSTATE RETURN CODE
DSNT415I SQLERRP    = DSNXIDIX SQL PROCEDURE DETECTING ERROR
DSNT416I SQLERRD    = 500  0  0  -1  0  0 SQL DIAGNOSTIC INFORMATION
DSNT416I SQLERRD    = X'000001F4'  X'00000000'  X'00000000'  X'FFFFFFFF'
X'0000
```

To bypass the error message in Example 10-2, we ran a JCL that first alters the table CSQ.OBJ_B_QUEUE to remove the constraint. Once the constraint was removed, we were able to drop the index and proceed with the second job (CSQ45Z02) of the hold data. After we ran this job, we recreated the referential constraint. Note that under normal circumstances you should not encounter such a problem and should not remove referential constraints on WebSphere MQ tables.

Once the PTF is installed and the MQ environment is customized, as per “Task 9: Set up of the DB2 environment” in Chapter 2 of the *WebSphere MQ for z/OS System Setup Guide,* SC34-6052, it is necessary to rebind the application plans for each DB2 data-sharing group against which the CSQ45CTB sample JCL was executed.

The WebSphere MQ coexistence PTFs do not implement any new modules for WebSphere MQ. They simply ship changed DBRMs so all users of queue-sharing groups must rebind their DB2 application plans with the new levels. The PTF once installed and applied will bind the users’ DB2 plans with the new 221 suffix and grant authority to existing users of the rebound plans.

Once the coexistence PTF is installed, you can proceed with installing the new WebSphere MQ V5.3 libraries and customizing your queue-sharing and DB2 data-sharing groups. While not complicated, this procedure does require certain parameters from your environment that need to be located before proceeding with the migration.

These parameters can be extracted from the DB2 catalog tables using SQL SELECT statements or by referencing the JCL that was used when configuration of the DB2 data-sharing groups during the configuration of MQSeries for OS/390 V5.2 was performed.

To execute the SQL SELECT statements, you can use the SPUFI option under ISPF.

Using the SPUFI SQL execution tool is quite painless, requiring only some prior knowledge of SQL or access to a DB2 manual that outlines the DB2 SQL statements. A good manual to reference is *DB2 UDB for OS/390 and z/OS V7 SQL Reference,* SC26-9944.

Example 10-3 DB2 SQL SELECT statement

```
SELECT * FROM SYSIBM.SYSDATABASE ;
SELECT * FROM SYSIBM.SYSTABLES WHERE DBNAME='MQ52DB' ;
```

In Example 10-3 the first SELECT statement will extract all entries from the SYSDATABASE table and write them to your output data set. From the output shown in Example 10-3, you can see the WebSphere MQ-specific databases within the DB2 data-sharing group that are used to store the WebSphere MQ queue-sharing definitions.

```
---------+---------+---------+---------+---------+---------+---------+
NAME      CREATOR   STGROUP   BPOOL       DBID  IBMREQD  CREATEDBY ...
---------+---------+---------+---------+---------+---------+---------+
DSNDB04   SYSIBM    SYSDEFLT  BP0            4  Y        SYSIBM
...
MQ52DB    YUJI      MQ52STG   BP32K1       260  N        YUJI
MQDB2RRS  CLARK     SYSDEFLT  BP0          261  N        CLARK
...
DSNE610I NUMBER OF ROWS DISPLAYED IS 10
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+
```

Figure 10-2 SQL select statement output from SYSIBM.SYSDATABASES

```
   SELECT * FROM SYSIBM.SYSTABLES WHERE DBNAME='MQ52DB' ;
---------+---------+---------+---------+---------+---------+------
NAME                CREATOR   TYPE  DBNAME    TSNAME      DBID ...
---------+---------+---------+---------+---------+---------+------
ADMIN_B_QSG         CSQ       T     MQ52DB    MQ4KTBL      260
ADMIN_B_QMGR        CSQ       T     MQ52DB    MQ4KTBL      260
OBJ_B_NAMELIST      CSQ       T     MQ52DB    MQ32KTBL     260
ADMIN_B_STRUCTURE   CSQ       T     MQ52DB    MQ4KTBL      260
OBJ_B_QUEUE         CSQ       T     MQ52DB    MQ4KTBL      260
OBJ_B_PROCESS       CSQ       T     MQ52DB    MQ4KTBL      260
OBJ_B_STGCLASS      CSQ       T     MQ52DB    MQ4KTBL      260
ADMIN_B_SCST        CSQ       T     MQ52DB    MQCHTBL      260
ADMIN_B_SSKT        CSQ       T     MQ52DB    MQCHTBL      260
TEMP_B_CHANNEL      CSQ       T     MQ52DB    MQ32KTBL     260
OBJ_B_AUTHINFO      CSQ       T     MQ52DB    MQ4KTBL      260
ADMIN_B_STRBACKUP   CSQ       T     MQ52DB    MQ4KTBL      260
OBJ_B_CHANNEL       CSQ       T     MQ52DB    MQ32KTBL     260
DSNE610I NUMBER OF ROWS DISPLAYED IS 13
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+------
```

Figure 10-3 SQL Select statement output from SYSIBM.SYSTABLES

The second SELECT statement in Example 10-3 on page 200 extracts all the database table entries from the database MQ52DB as you can see from the output in Figure 10-3. The output shows the name of the tablespaces, the database name and so on.

In Figure 10-2 on page 200 and Figure 10-3, we do not show the entire output generated by the SQL SELECT statements, yet it is possible to limit your SELECT statements to just the data that you require by using wildcards. A good manual to use as a reference guide is *DB2 UDB for OS/390 and z/OS V7 SQL Reference,* SC26-9944.

Migration job steps

After installing the new early code to the linklist using SMP, and an IPL of the LPAR has been performed, it is necessary to customize the DB2 data-sharing group that is configured for the MQ queue-sharing group. To enable customization, a sample JCL has been provided, CSQ45ATB from install library, <MQHLQ>.SCSQPROC. This sample job comprises eight separate job steps, which we describe below. After successfully running each job step, your DB2 tables will be bound with the new DB2 plans.

When performing the migration for our install, we split the CSQ45ATB job by logical steps into eight separate jobs, primarily to allow each step to complete successfully before submitting the next step. The migration process itself is not that difficult, although it is time consuming to gather all the environment parameters. As a guide, we have created a list of parameters in Table 10-1 that are required for input to the sample JCL before being run.

Table 10-1 CSQ45ATB replaceable parameters

Parameter Name	Replaceable parameter
DB2 target libraries	++DB2QUAL++
MQ V5.3 target libraries	++THLQUAL++
MQ V5.2 target libraries	++V52QUAL++
Language of WebSphere MQ install	++LANGLETTER++
DB2 Data Sharing Group Name	++DSGNAME++
DB2 database SSID name	++DB2SSID++
Current installed version of DB2	++DB2VER++
WebSphere MQ QSG DB name	++DB2DBNAME++
WebSphere MQ DB2 Storage Group	++DB2STGGRP++
4 KB page Tablespace name	++DB2TBLSPCQMGR4K++
32 KB page Tablespace name	++DB2TBLSPCQMGR32K++
Auth-Info Index Primary space quantity allocation	++ICHAPRIQTY++
Auth-Info Index Secondary space quantity allocation	++ICHASECQTY++
User IDs that will use the CSQ5PQSG utility	++PQSGUSERID++

Once the above jobs have been implemented and the data-sharing group has been migrated to WebSphere MQ V5.3 levels, you can restart the queue managers that form the queue-sharing group. If the queue managers restart successfully, as ours did, it is recommended that the CFLEVEL structures are migrated to the new levels to allow for recovery of WebSphere MQ messages that are stored on the coupling facility.

Migrate the coupling facility structure

In MQSeries V5.2 and WebSphere MQ V5.3, local queue objects defined with QSGDISP(SHARED) have their messages stored in a coupling facility so that they can be accessed by other queue managers in the queue-sharing group. In WebSphere MQ V5.3, an object called *CF structure* describes the capabilities through the CFLEVEL and RECOVER attributes. DEFINE CFSTRUCT, ALTER CFSTRUCT, DELETE CFSTRUCT and DISPLAY CFSTRUCT commands have been added for these objects.

In MQSeries V5.2, CFLEVEL(1) structure operations were implicitly created and deleted when required by the queues putting messages to the coupling facility.

WebSphere MQ V5.3 introduces CFLEVEL(2) to allow for compatibility between V5.2 and V5.3 queue managers. These CF structure objects can be used by V5.2 queue managers and they can be manipulated by V5.3 queue managers.

Once all queue managers in the queue-sharing group have been migrated to V5.3, it is recommended that you migrate a CF structure from CFLEVEL(2) to CFLEVEL(3) using the DEFINE REPLACE or ALTER CFSTRUCT commands.

To upgrade the CF structure to CFLEVELs 2 or 3, it is necessary to run the command shown in Example 10-4. We issued this command from a console, using the subsystem interface of our queue manager MQV2, since it is not available to be run from the CSQINP1 initialization input data set. The output in Example 10-5 on page 204 should be received. The output should confirm that the CF structure has been upgraded to the new level.

Example 10-4 Alter CF structure command

```
-MQV2 ALTER CFSTRUCT(APPLICATION1) CFLEVEL(3) RECOVER(Y)
```

The above command contains the following:

- `-MQV2` is the queue manager subsystem identifier, addressing the queue manager, to which the commands are to be routed to from the MVS console.
- `CFSTRUCT(APPLICATION1)` specifies our WebSphere MQ structure APPLICATION1
- `CFLEVEL(3)`

  This parameter should be set to the target CF structure level. In our case we want to migrate to CFLEVEL(3). Use the following rule of thumb:

  – Queue Manager = MQ V5.2; then use CFLEVEL(1) structure
  – Queue Manager = MQ V5.2 + MQ V5.3; then use CFLEVEL(2) structure
  – Queue Manager = MQ V5.3; then use CFLEVEL(3) structure
- `RECOVER` can be set to either Y (YES) or N (NO)

Example 10-5 Alter CF structure output

```
  CSQU121I CSQUTIL Connected to queue manager MQV2
0 CSQU055I CSQUTIL Target queue manager is MQV2
 ALTER CFSTRUCT(APPLICATION1) CFLEVEL(3) RECOVER(Y)
0CSQN205I   COUNT=       2, RETURN=00000000, REASON=00000000
 CSQ9022I -MQV2 CSQMACFS ' ALTER CFSTRUCT' NORMAL COMPLETION
- CSQU143I CSQUTIL 1 COMMAND statements attempted
0 CSQU144I CSQUTIL 1 COMMAND statements executed successfully
0 CSQU148I CSQUTIL Utility completed, return code=0
```

Remember, it is possible to use the RECOVER CF structure facility only if all queue managers within the WebSphere MQ queue-sharing group have been migrated to WebSphere MQ for z/OS V5.3.

- Benefits of CFLEVEL(3) structures:
  - Persistent messages can be stored on a CF structure with CFLEVEL(3) and the RECOVER(YES) attribute.
  - New commands, BACKUP CFSTRUCT and RECOVER CFSTRUCT, are provided for support recovery.
  - It is recommended that regular backups of the CF structures are implemented within your installation.
  - Higher resilience to enable full recovery of messages after a system crash or power outage.
- Restrictions:
  - All queue managers in the sysplex must be migrated to WebSphere MQ V5.3 to enable upgrade to CFLEVEL(3), thereby enabling recovery of the CF structure in case of a power outage.

For further detailed information relating to configuration of the coupling facility, refer to “Task 9: Set up the Coupling Facility” in *WebSphere for z/OS MQ V5.3 System Setup Guide Version 5 Release 3,* SC34-6052.

Fallback to MQSeries V5.2

Backward migration can be achieved by changing library concatenations that reference the V5.3 product to pick up the V5.2 product libraries that contain the WebSphere coexistence PTF code and restart the queue manager.

10.3 MQSeries V2.1 to WebSphere MQ V5.3 migration

To support shared queues, a number of parameters have been added to the system parameter module that need to be added to your system environment when migrating to WebSphere MQ V5.3. We discuss them here briefly. In Chapter 9, “Creating a shared queue environment” on page 159, we go into specific detail in regards to implementing queue sharing within your WebSphere MQ application. All these parameters were implemented with MQSeries V5.2, except for the QINDXBLD parameter, which has just been included with WebSphere MQ for z/OS V5.3.

10.3.1 New system parameters

CSQ6SYSP

Given that WebSphere MQ is using DB2 services to support shared queues, it is not surprising to see some DB2 specific parameters.

- QINDXBLD

  This new parameter determines whether a queue manager waits until all queue indexes are rebuilt. The parameters are:

  – Wait

    Queue manager restart waits for all queue index builds to complete. This means that all indexes are created before any applications can connect to the queue manager.

  – Nowait

    Queue manager restart completes before the queue index build is complete.

- QSGDATA

  This is a new parameter, which has four positional parameters:

  – Qsgname

    The name of the queue-sharing group to which the queue manager belongs. The default value is blank. In this case the queue manager does not belong to a queue-sharing group.

  – Dsgname

    The name of the DB2 data-sharing group to which the queue manager is to connect.

  – Db2name

    The name of the DB2 subsystem to which the queue manager is to connect.

– Db2servers

  The number of server tasks used for accessing DB2. The value can be from 4 to 10. The default is 4.

- RESAUDIT

  The default value is YES. In this case RACF audit records are written for RESLEVEL security checks during connection processing. This parameter is new.

- LOGLOAD

  The default setting for this parameter has been changed. The value is now 500 000, which is a more reasonable value for a production environment. If you still use the old default value in your environment, it is highly advisable to reconsider the value for LOGLOAD or to just implement the new default.

CSQ6LOGP

This section of the system parameter module has a few changes. Some control over the use of tape devices is now possible and two parameters now have another default value that better fits the needs of a production environment.

- DEALLCT and MAXRTU

  These new parameters allow for better tuning of the usage of tapes by WebSphere MQ. DEALLCT specifies the interval in minutes that a tape unit can be allocated without being used. The default is 0, which means that MQSeries decollates the tape immediately after it has finished its archive read process.

  MAXRTU specifies the number of tape devices that WebSphere MQ can use concurrently to read archive logs. Having more than one device available allows WebSphere MQ to optimize the archive log reading process.

  MAXRTU supersedes the MAXALLC parameter, which is no longer used. MAXALLC specified how many archive log *volumes* could be allocated, whereas MAXRTU limits the number of *devices*.

  It is also worthwhile noting that there is a WebSphere MQ SET LOG command to change the values of these parameters. This removes the need to stop the queue manager for loading a new parameter module. There is also a DISPLAY LOG command to show the active values for these parameters and to report the current usage of tape devices.

- INBUFF and OUTBUFF

  The default values for these parameters have been changed. The new values are again more suitable for a production system than the old values. These two parameters, together with the LOGLOAD parameter, are important keys to optimize the logging component of WebSphere MQ.

CSQ6ARVP

Here, too, some small changes have been done to provide more control over the archive process. The keyword UNIT2 has been added to specify the device type or unit name that is used to store the second copy of the archive log data set. The existing keyword UNIT is then only used for the first copy. This will be useful to enable backups to be made that can be sent offsite and stored for disaster recovery purposes.

10.3.2 ADOPTMCA

The ADOPTMCA feature was available as a PTF in earlier releases of MQSeries for OS/390, but it is now shipped with the base code.

The ADOPTMCA feature provides a solution where receiving or inbound channels are stuck in a receive-wait condition. The receiving side thinks that the session with the sender is still active, while in reality the connection is broken. On the other hand, the sending channel is aware of the broken session and has started its normal retry mechanism. It sends a start-channel request to the listener on the receiving platform. Without the use of the ADOPTMCA feature, the channel initiator will respond to this start-channel request with a channel-in-use error message. The sending channel will log this error message and start another retry cycle. However, without any intervention, the sending channel will not succeed.

If the queue manager was configured to use the TCP/IP KeepAlive feature, then at some point the TCP/IP subsystem will notify the waiting receiver channel that its connection has been dropped. However, the interval at which this checking occurs is normally set rather high and its value is not suitable for typical WebSphere MQ communication. Note that this problem is typical for a TCP/IP network while it is not known in an SNA network.

The AdoptMCA feature introduces the following logic to the channel initiator: if a start-channel request arrives for a running receiving channel and the start-channel request meets certain criteria and the ADOPTMCA feature is activated, then the running receiving channel is killed. By canceling the orphaned receiving channel, the start-channel request can be honored.

To activate this feature, you set the ADOPTMCA parameter in the parameter module to YES. By default it is set to NO to maintain compatibility with the current behavior. To control the criteria for canceling the orphaned channel, you can use the ADOPTCHK parameter. You can specify that the start-channel request should meet one of the following conditions:

- The originating queue manager is the same for the start-channel request as for the orphaned receiving channel.

- The originating network address is the same for the start-channel request as for the orphaned receiving channel.
- Both values, the queue manager name and the network address, should match.
- None of the values are checked.

This additional checking may be necessary in some situations. On one hand, it is a way to make sure that the running receiving channel is an orphaned receiving channel. If someone had accidentally defined a sender channel on another queue manager and/or another machine, then either of the above conditions are not met when the misdefined sender channel was started. If the checking was not done, it would open an easy way to interrupt the normal communication of your channels. Each time the misdefined sender channel was started, the receiving channel - in communication with the correctly defined sender - would be cancelled. When the correctly defined sender issues a new start-channel request, the session with the badly defined sender would be killed. In the end, you have a situation where the communication is turned on and off again for no apparent reason.

This would make one believe that the checking of both values is always necessary. Consider, however, the case where the machine that hosted the sending queue manager is completely out of business because of some serious hardware malfunction. Then the WebSphere MQ administrator would have to restart the queue manager on another system by reconnecting to disks to that system. That new system could possibly run with another network address. If the receiving channel initiator were to check both conditions, the communication would not be re-established and an intervention on the OS/390 side would be necessary.

But even the checking on the name of the queue manager can be wrong.

11

High availability setup

In this chapter, we describe the high availability IP Parallel Sysplex configuration that enables an application infrastructure with no single point of failure. Each component was implemented with a backup component to ensure that if an outage at one point occurs due to the configuration of the application, a backup component would be able to take over. During discussions on the best practice to establish a no single point of failure architecture during this residency, the following scenarios were designed and tested for recommendation to IBM customers for implementation.

11.1 Introduction

In Chapter 8, “Shared queue overview” on page 137 and Chapter 9, “Creating a shared queue environment” on page 159, we provide an overview of the queue-sharing environment and guidance on setting it up.

Basically a queue-sharing group (QSG) requires that all LPARs are members of a DB2 data-sharing group and that the Resource Recovery Services (RRS) component is available. All queue managers in the QSG share a channel definition. They also all start INDISP (GROUP) listeners with the same shared port number to which remote systems can connect. The receiver channel status is kept in a shared queue (SYSTEM.QSG.CHANNEL.SYNCQ) so that a given remote queue manager can reconnect to any of the queue manager instances in the QSG and resynchronize its sending channel status with the corresponding receive channel status. A distributed VIPA (in this example DVIPA1) can be used to represent the Parallel Sysplex, and the Sysplex Distributor can distribute incoming connections to any of the available channel initiators that are up and listening on the shared port number.

Other connection balancing technologies will also work, including DNS/WLM or external connection balancers (dispatcher type of balancers). Any queue manager in the QSG can accept a channel setup request from the remote system and write messages to the shared queue. In MQSeries V5.2, shared queues can only be used for non-persistent messages, WebSphere MQ for z/OS V5.3 enables the use of persistent messages to be used in a queue-sharing group. Shared messages can have a length of up to 63 KB. It is possible to connect up to 32 queue managers to a single QSG and if one queue manager were to fail, messages on a shared queue within the coupling facility can be accessed by other queue managers within the QSG. Even if all the queue managers within the QSG fail, messages in the coupling facility will be kept.

11.2 WebSphere MQ gateway or bus topology

There are two basic network topologies to be used with WebSphere MQ (see Figure 11-1 on page 211):

- A bus topology where all queue managers can directly connect to all other queue managers and deliver messages directly to the target QM, using a sender/receiver message channel pair
- A hub-and-spoke (gateway) topology where messages between the two WebSphere MQ domains pass through a gateway WebSphere MQ

Each model has advantages. The bus model gives optimal performance (no intermediate hops), while the gateway model gives security and isolation (one focal point to implement security functions). The two models can be combined; this is the approach we follow in our scenario, by having a gateway model at the edge of the Parallel Sysplex having all messages entering and leaving the sysplex pass through a gateway queue manager. This provides security and availability (no single-point-of-failure), while inside the sysplex, all queue managers can exchange messages directly with each other (a bus model).

In this project, we decided to implement the scenario described in 11.3.3, “Scenario 3: One large QSG” on page 217. This configuration implements a complete queue-sharing environment between gateway and application queue managers and has the advantage of separation between communication management and application functions.

Alternatively, you might consider a configuration where all members of the queue-sharing group are clones of one another. In such a bus configuration, there are no gateways and inbound channels can connect to any member of the queue-sharing group.

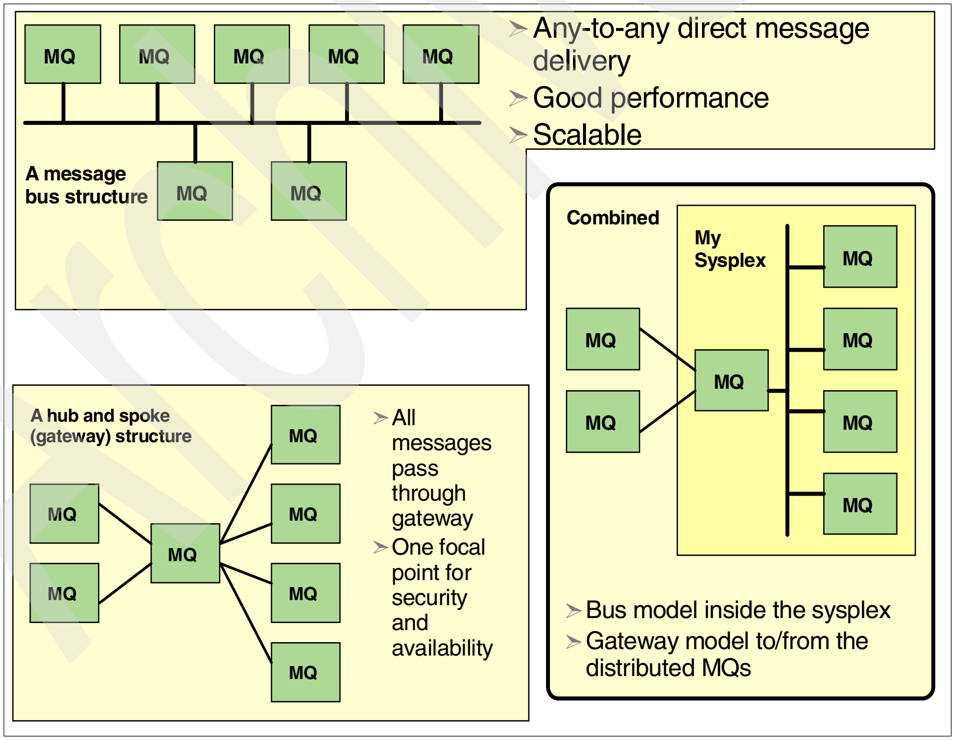


Figure 11-1 Example: WebSphere MQ gateway or bus structure

11.3 Queue manager gateways in sysplex scenarios

The main goal of this project was to explore the use of WebSphere MQ V5.3 within a Parallel Sysplex environment to determine the best configuration to exploit the queue-sharing group and data-sharing group features. In this chapter, we outline the three scenarios that were considered by us, documenting the benefits of each scenario. We have also included a detailed implementation plan for the final and most effective scenario that was chosen by the residency team as the most effective implementation of no single point of failure architecture.

11.3.1 Scenario 1: Restart single gateway queue manager

WebSphere MQ clustering is used in the scenario shown in Figure 11-2 on page 213 to simplify administration of WebSphere MQ definitions. Only the queue managers in the Parallel Sysplex are members of the sysplex WebSphere MQ cluster, not the remote queue managers. Clustered queue managers reduce the amount of definitions within the cluster. When a queue is defined on MQV1, for example, the queue definitions are propagated to the repository queue manager(s), and other queue managers in the cluster can locate information about the queue and transmit data to it without having transmit channels predefined.

The queue manager on SC53 (MQV1) could serve as the cluster's full repository queue manager with a replica on one of the other application LPARs. In this scenario, the intermediate queue manager itself (MQV2) does not own any application queues, but just belongs to the sysplex WebSphere MQ cluster, so that all queues can be reached from the gateway queue manager without predefining channels between this queue manager and all the other application nodes in the sysplex. Remote queue managers point to DVIPA1 as their CONNAME on their sender channel definition into the sysplex.

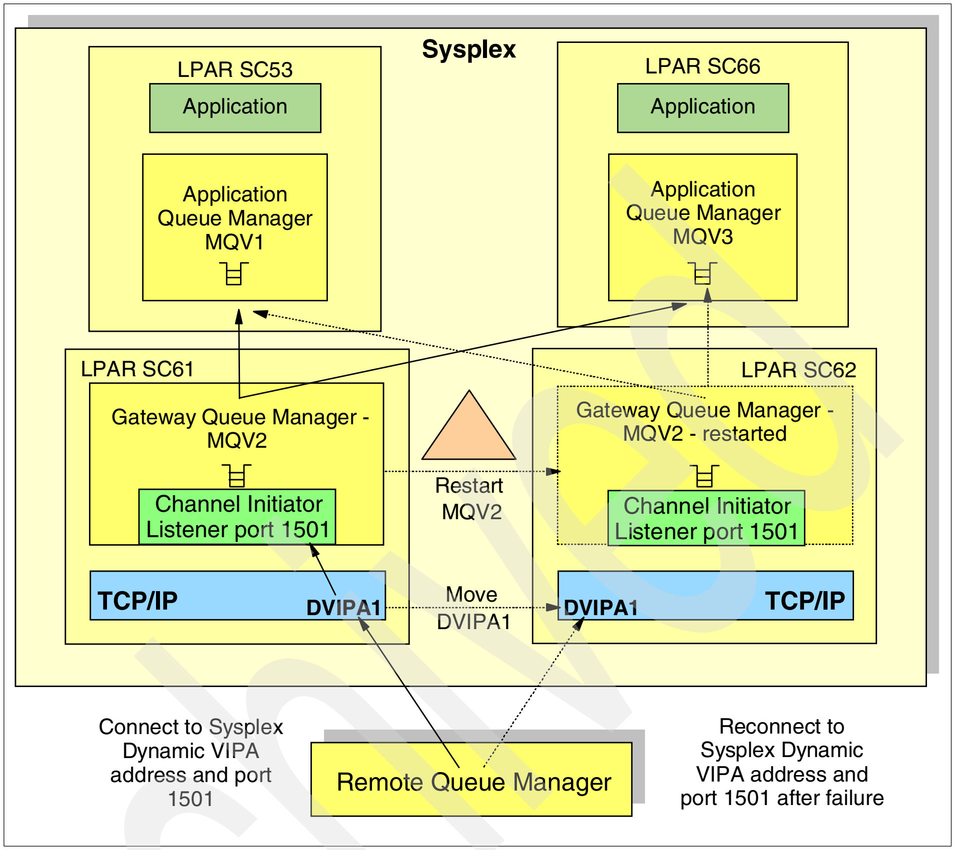


Figure 11-2 Scenario 1: restart single gateway queue manager

Remote queue managers have a transmission queue served by the aforementioned channel on their queue remote definitions for any queue in the sysplex, thereby providing a single interface into the sysplex WebSphere MQ cluster that allows for a single point of control to implement security controls. MQV2 on the gateway defines queue manager aliases for all the distributed queue managers.

- The basic approach here is to restart the failed gateway queue manager (MQV2) on another LPAR in case of failure of the LPAR or the gateway queue manager itself on its original LPAR.
- At any point in time there is only a single gateway queue manager in the sysplex (called MQV2 in this example).
- By associating the queue manager with an application-specific Dynamic VIPA, the Dynamic VIPA moves with the queue manager when the queue manager is restarted on the backup LPAR. Remote queue managers can

reconnect and synchronize their channel status, since it is the same queue manager instance as they were connected to before the failure - it now runs on another LPAR in the sysplex.

- The remote queue managers use a regular sender channel to connect to DVIPA1.
- Applications connected to the remote queue manager that wish to put a message on a queue in the sysplex cluster hosted by one of the application LPARs put to a regular queue remote definition that targets a clustered remote queue definition on the gateway queue manager. WebSphere MQ clustering in the sysplex then resolves which application LPAR to deliver to and places the message on the SYSTEM.CLUSTER.TRANSMIT.QUEUE on the gateway queue manager for delivery to the appropriate target queue manager through a cluster-sender channel.
- This design does not provide load-balancing or real-time backup capability. In case of a failure, the failed queue manager must be restarted either manually or through ARM policies on the backup LPAR, and the IP routing topology must converge to the new status before remote queue managers can reconnect. This may take from less than a second to three minutes depending on the dynamic routing protocol in use (OSPF is as always the recommended dynamic routing protocol due to its speed of convergence - RIP protocol may take up to three minutes).
- Note that WebSphere MQ does provide procedures for how to take an LPAR down for planned maintenance in such a way that no messages get left in the transmit queues during the planned outage.

Benefits

This is a simple design to implement. It provides gateway backup capability, but note that the backup is not immediate. It may take some time before the gateway queue manager has been restarted and remote queue managers cannot reconnect their channels until the gateway queue manager has been restarted.

11.3.2 Scenario 2: Replicated gateway queue managers in a QSG

The basic principles of scenario 2, shown in Figure 11-3 on page 216, are taken from a WebSphere MQ clustering perspective, almost the same as in the previous sample. All Parallel Sysplex queue managers are in a single WebSphere MQ cluster. LPARs SC61 and SC62 are configured as WebSphere MQ gateways to which remote queue managers connect using an IP cluster (sysplex) identity (such as the Sysplex Distributor single IP address and port). Since we now have two concurrently active gateway queue managers, we can have both of them serve as cluster full repository managers. Both queue managers in the QSG share a channel definition. They also both start

INDISP(GROUP) listeners with the same shared port number to which remote systems can connect. Inbound connections go to one of the two gateway queue managers on these nodes and messages are then transferred to their final target queue manager inside the sysplex using any of the available WebSphere MQ transports.

The two gateway queue managers share a channel definition with the same shared port number to which remote systems can connect. The receive channel status is kept in a shared queue (SYSTEM.QSG.CHANNEL.SYNCQ) so that a given remote queue manager can re-connect to any of the two gateway queue manager instances in the QSG and resynchronize its sending channel status with the corresponding receive channel status.

Only the gateway queue manager nodes need to be in a queue-sharing group (QSG) and a corresponding data-sharing group in this scenario, since only these two queue managers share queues and queue status. Since only the gateway queue managers are in a QSG, the gateway queue managers cannot use shared queues for handling messages to the application LPARs, but must use the standard WebSphere MQ message channel agent (MCA) to pass them on whatever transport technology may be available inside the sysplex.

Applications that are connected to the remote queue managers that wish to put a message on a queue in the Parallel Sysplex cluster that is hosted by a queue manager in one of the application LPARs will do exactly as in the previous setup.

But note that, although the message arrives at the gateway queue manager through a shared channel, the message is put on a cloned non-shared queue for delivery to the target queue manager in the Parallel Sysplex.

This is because the SYSTEM.CLUSTER.TRANSMIT.QUEUE cannot be a shared queue. If one of the gateway queue managers goes down, normal retry processing in the regular sender channel on the remote queue manager will allow it to immediately reconnect to the other gateway queue manager. Any messages that were on the SYSTEM.CLUSTER.TRANSMIT.QUEUE on the original gateway queue manager for delivery to the target queue managers in the sysplex will not be available for delivery until that gateway queue manager becomes available again, since they are not on a shared queue.

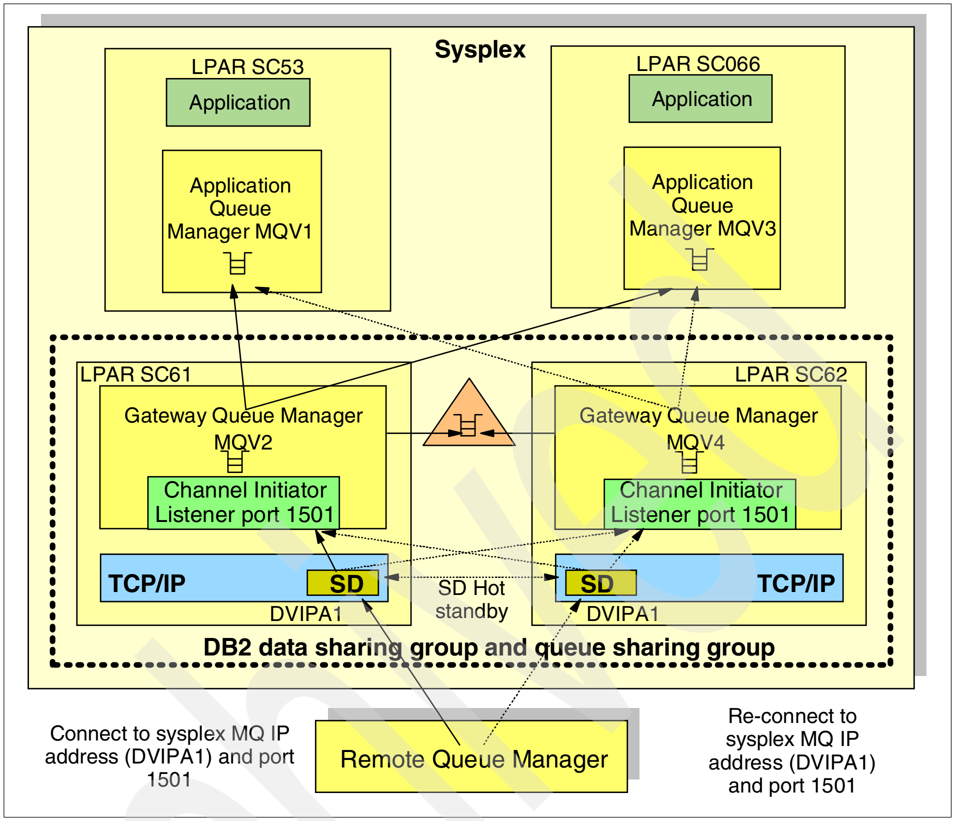


Figure 11-3 Scenario 2: Replicated gateway queue managers in a QSG

Benefits

This design provides load balancing between two or more gateway queue managers in the Parallel Sysplex. If one goes down, the other is presumably still up and remote queue managers that were connected to the failed gateway queue manager can reconnect immediately and begin transmitting new messages in the sysplex. Transient messages that were put onto a sender channel to a target queue manager in the sysplex will not be recovered until the failed gateway queue manager on the same LPAR where the other gateway queue manager is running. In cases where an LPAR is taken down for a period of time for maintenance, such a procedure could reduce the time any potential messages were sitting on a non-shared sender channel. If applications are sensitive to the order of message delivery, they should issue their MQOPEN with MQOO_BIND_ON_OPEN to ensure that messages are not being delivered through another queue manager when the first is down, risking that some messages may pass messages sitting in the transmit queue of the failed queue manager.

11.3.3 Scenario 3: One large QSG

In scenario 3, shown in Figure 11-4, all LPARs in the Parallel Sysplex are now part of one and the same queue-sharing group, which allows for the gateway queue managers to use shared queues for passing inbound and outbound messages to/from the application LPARs.

This scenario requires a full DB2 data-sharing group across all LPARs and a queue-sharing group across LPARs.

WebSphere MQ clustering is still used to reduce administrative definitions.

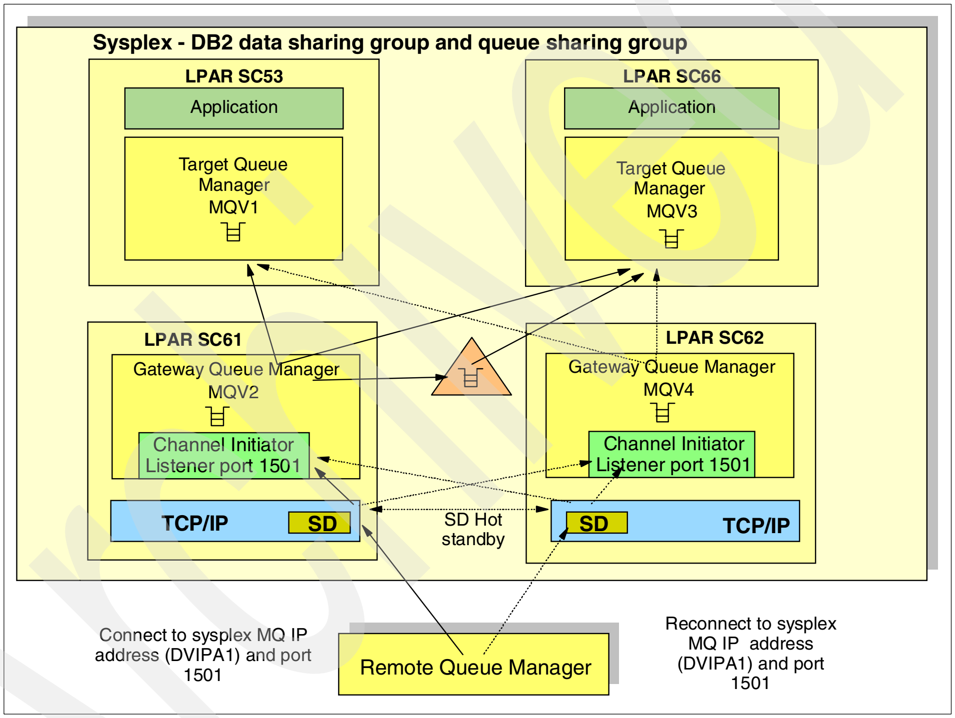


Figure 11-4 Scenario 3: One large QSG

Target queues on the application LPARs may or may not be shared queues in this scenario and the capabilities differ somewhat depending on whether the queue has a disposition of shared (QSGDISP(SHARED)) or not.

If target queues are shared:

- The remote queue manager uses a regular sender channel to connect to DVIPA1 IP address and talk to either one of the gateway queue managers.

- Applications connected to the remote queue manager that wish to put a message on a sysplex queue that is accessible by one of the target queue managers put to a regular queue remote definition that targets the Sysplex gateway queue manager, which can immediately place it on the shared queue. The application connected to the target queue manager can get the message off the target shared queue and process it. There is no need for clustering to deliver the message from the gateway queue manager to the target queue manager.
- If one of the gateway queue managers goes down, normal retry processing in the regular sender channel on the remote queue manager will allow it to immediately reconnect to the other gateway queue manager. All queues used are shared queues and so are available to both the gateway queue managers, and work will continue.

On the other hand, if target queues are not shared:

- The remote queue manager uses a regular sender channel to connect to the DVIPA1 IP address and talk to either one of the gateway queue managers.
- Applications connected to the remote queue manager that wish to put a message on a sysplex queue that is accessible by one of the target queue managers put to a regular queue remote definition that targets a clustered remote queue. As stated in the second scenario setup, the SYSTEM.CLUSTER.TRANSMIT.QUEUE cannot be a shared queue, but if you have intra-group queuing enabled, then because the destination queue manager is in the same QSG, the message can be placed on the SYSTEM.QSG.TRANSMIT.QUEUE for delivery to that target queue manager. So clustering is used for queue name resolution, but no cluster channels are needed to deliver the messages to the target queue managers unless the messages are larger than 63 KB.
- If one of the gateway queue managers goes down, normal retry processing in the regular sender channel on the remote queue manager will allow it to immediately reconnect to the other gateway queue manager. All queues used for delivery of messages are shared queues and so are available to both gateway queue managers and work will continue. Only the target queues are non-shared and so if one of the target queue managers goes down, work may be stuck until it is restarted.

Benefits

This design, as in scenario 2, provides load balancing between two or more gateway queue managers in the Parallel Sysplex, as well as fault tolerance. The sender channels of the two gateway managers can be shared, thus eliminating the message delivery sequence problem that might occur in scenario 2 in case of failure of one of the gateway systems.

11.3.4 Additional considerations

There are some additional considerations.

Reply message handling (Remote queue manager)

WebSphere MQ requests may generate replies that may be too large for delivery using the facilities of the queue-sharing groups as outlined in scenarios 2 and 3. To cater for this situation, there a number of approaches that can be taken in order to send the reply messages back to the remote queue manager in the most efficient manner.

- If replies from the application are less than or equal to 63 KB, they can be put straight on to a shared transmission queue and delivered to the remote queue manager using a shared outbound channel. The server application will open the reply-to queue qualified by the reply-to queue manager that it obtains from the message descriptor of the original request. This will resolve to the shared transmission queue. This would be the conventional approach to take for a server application.
- If replies from the application are greater than 63 KB, then they will be too long to be delivered using the shared transmission queue. These replies will have to be sent to the gateway using a private sender channel, and from the gateway to the remote queue manager using a private sender channel.

  To make use of this alternate path, a reply-to queue manager alias on both the application and gateway queue managers needs to be defined. This alias will resolve to a non-shared (private) transmission queue serviced by the private sender channel. Application changes will also be required on the server side.

  The server application will need to be modified so that it is aware that it is sending a “long” reply. If this is the case, then it can use this reply-to queue manager alias when it opens the reply queue.

  > **Note:** The server application must have prior knowledge of this alias name, since it will not be supplied in the message descriptor.

  The queue manager will resolve the destination to the private transmission queue defined for this class of reply. This scenario is shown in Figure 11-5 on page 220.

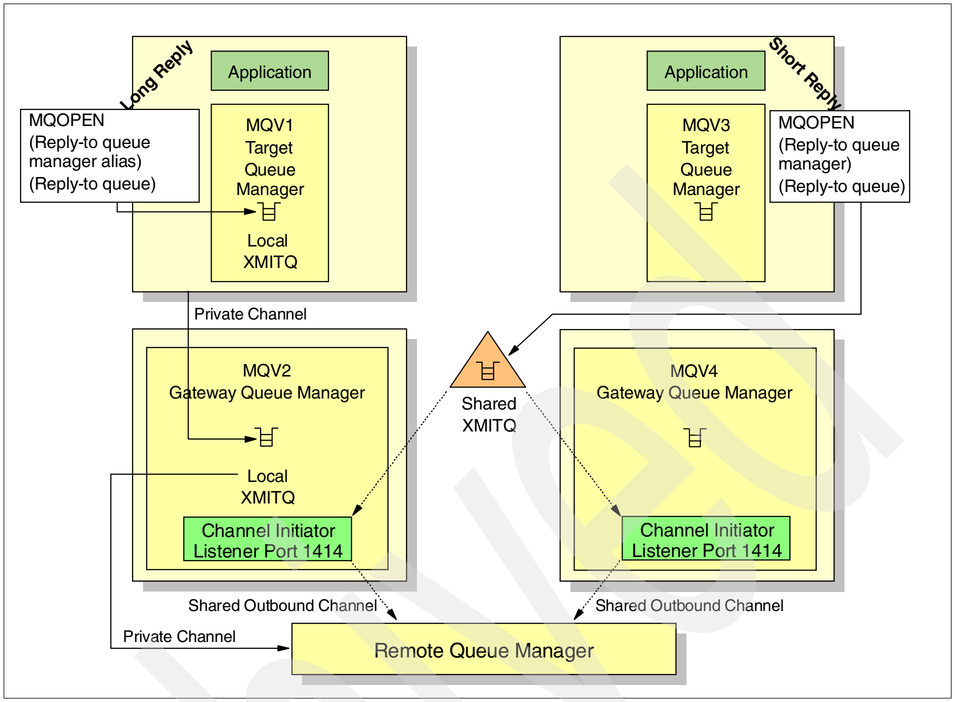


Figure 11-5 Handling of reply messages to a remote queue manager

Reply message handling (WebSphere MQ client)

If the client application connects directly to the gateway using the WebSphere MQ client, then IGQ (Intra group queuing) can be used. In this case, the logic to handle both "short" and "long" replies will be handled by WebSphere MQ.

> **Note:** IGQ will allow the queue manager to "hop" suitable messages to another queue manager in the QSG using a shared queue instead of a channel.

The server application will always open the reply queue qualified with the reply-to queue manager as supplied in the reply-to queue manager field. No action is required from the server application this time. IGQ will determine how it will be delivered to the gateway. The "short" replies will be intercepted by IGQ and "hopped" to the gateway using an IGQ shared queue called SYSTEM.QSG.TRANSMIT.QUEUE. The "long" replies will be placed on the private transmission queue and sent using a conventional channel. Either way, they will be delivered to the gateway, where they will ultimately be sent back to the WebSphere MQ client.

This scenario is shown in Figure 11-6.

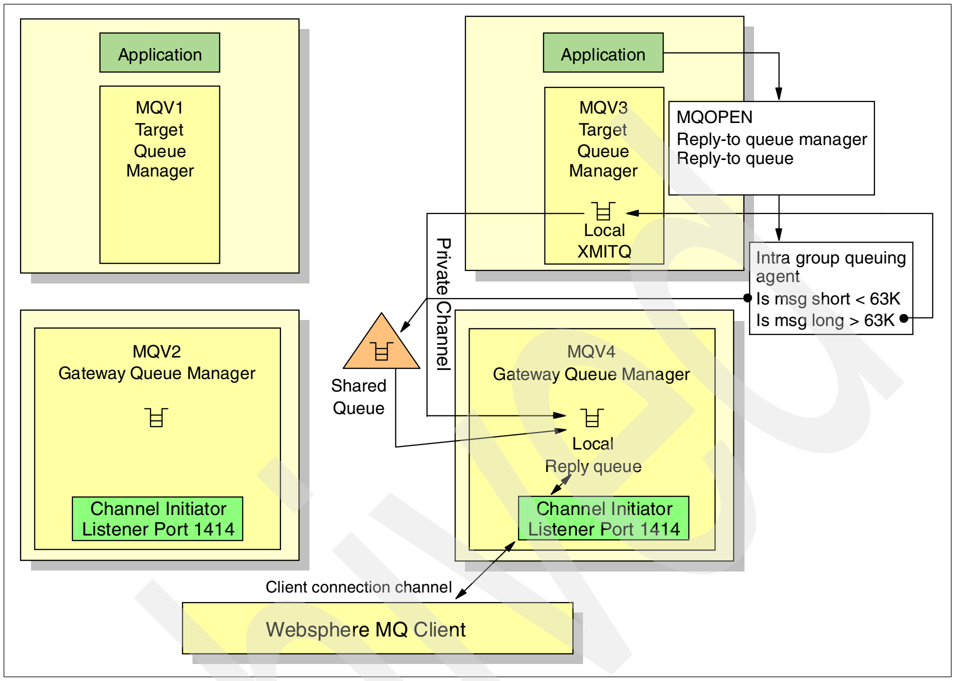


Figure 11-6 Handling of reply messages using intra-group queuing

Gateway connection using cluster channels

The high availability scenarios described previously suggest the use of the z/OS Sysplex Distributor and virtual IP addresses, combined with the WebSphere MQ shared inbound channels as the preferred method of connecting a remote queue manager to the gateways. If the use of this technology is not possible for whatever reason, the remote queue managers could be added to the WebSphere MQ cluster and requests could be sent to the gateway using cluster sender channels. In this scenario, the shared application queue could be added to the cluster. However, if a shared queue is clustered, it will appear to be hosted by all members of the queue-sharing group. When the remote queue manager discovers the application queue, it will see four instances of the queue and define additional cluster sender channels directly to the application queue managers MQV1 and MQV3. This configuration would work, but the preferred option is to route all requests through the gateway queue managers MQV2 and MQV4. This could potentially bypass any security that is being centrally controlled on the gateways. Figure 11-7 on page 222 shows this configuration.

.

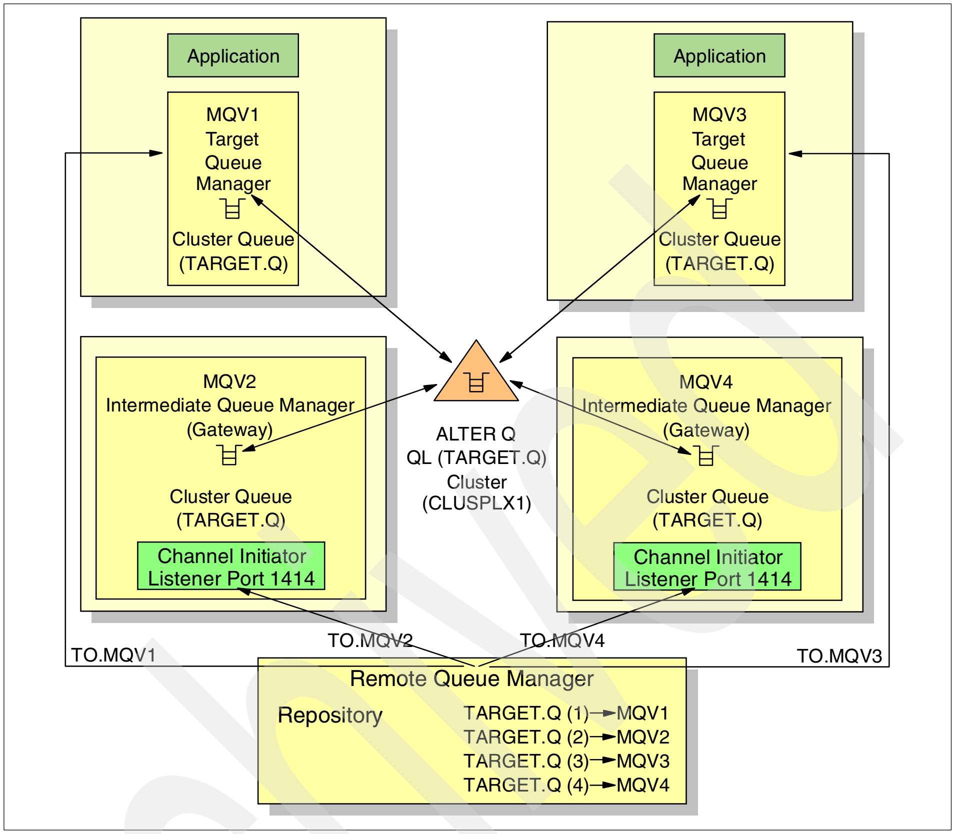


Figure 11-7 Result of clustering a shared queue

To maintain our gateway approach, we need to prevent the cluster sender channels being auto-defined to the application queue managers (MQV1 and MQV3). To achieve this, instead of clustering the shared application queue, we defined a clustered alias to the shared queue, but only on the gateway queue managers. As a result of this configuration, the remote queue manager will only see two instances of the queue, hosted by the gateway queue manager (MQV2 and MQV4). Load balancing can then occur across the two gateways. This is shown in Figure 11-8 on page 223.

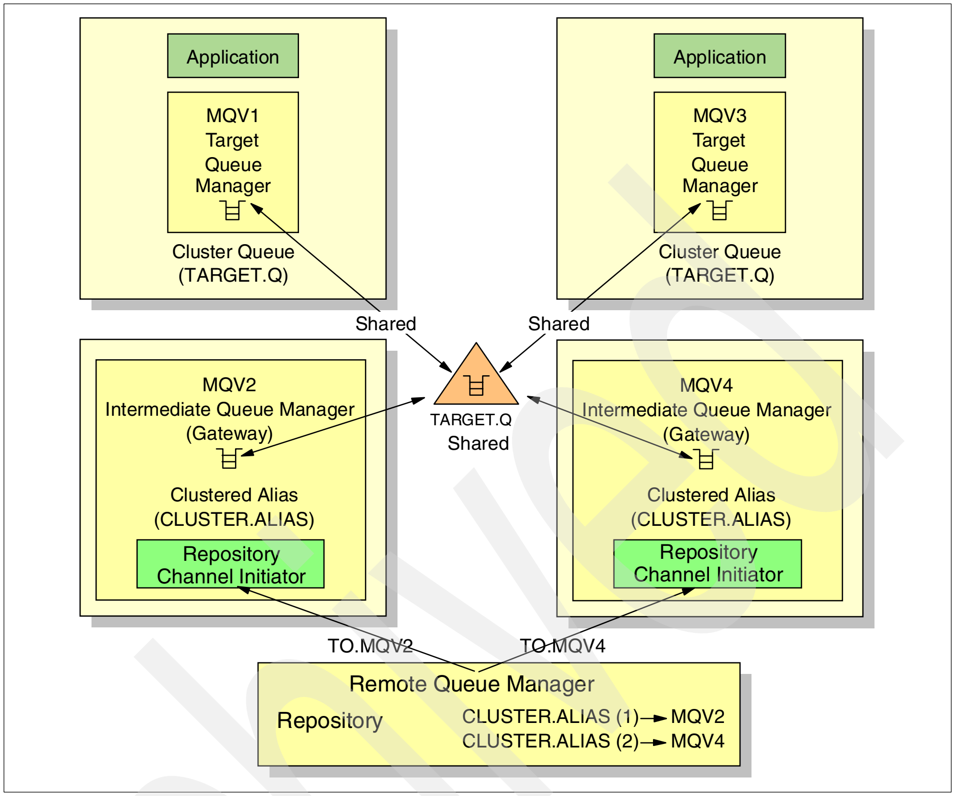


Figure 11-8 Clustering of the shared queue on the gateways only using aliases

We now have an alternative means for connecting the remote queue managers to our queue-sharing group. We have maintained our gateway approach and have also provided redundancy in the case of a gateway failure. There will be twice as many network connections compared with a Sysplex Distributor/shared channel setup, as every remote queue manager will be connected to both gateways. However, it may be a viable alternative for some customers.

12

CICS applications and queue sharing

In this chapter, we describe a CICS application that we developed to demonstrate the various workload management and failover scenarios described in the previous chapters. The application echoes the message that it gets from an input WebSphere MQ queue by placing it on an output queue. We called the CICS transaction associated with this application ECHO. When we ran workflow management and failover tests and studied different triggering options, we echoed the message in a CICS program that was triggered.

Our additional goal was to show how an Enterprise JavaBean (EJB) running in CICS can be invoked with WebSphere MQ triggering. We configured our CICS TS 2.2 regions as EJB servers. Refer to *Enterprise JavaBeans for z/OS and OS/390 CICS Transaction Server V2.2*, SG24-6284, for configuration information. The CICS program that gets the message from an input queue would not echo this message itself, but would invoke the CICS-provided HelloWorldEJB enterprise bean that echoes the message.

We also describe in this chapter a suite of Windows command scripts we used to generate the workload for CICS to process. These scripts use the sample program AMQSREQ that is supplied with WebSphere MQ on distributed platforms.

12.1 Application environment

The message driven CICS EJB application is a small example demonstrating how to utilize the capabilities of CICS session EJBs using WebSphere MQ triggering. A client application running in the network constructs a request message and places it on an application queue. The client application specifies the name of the reply-to queue in the message descriptor. The WebSphere MQ user application gets a message from the application queue and transfers its contents as a text string to the CICS EJB. The EJB simply returns the message to its caller, which will then put the message to a reply queue.

We wanted to benefit from the advantages of using shared queues. Therefore, we have installed the application back-end environment shown in Figure 12-1.

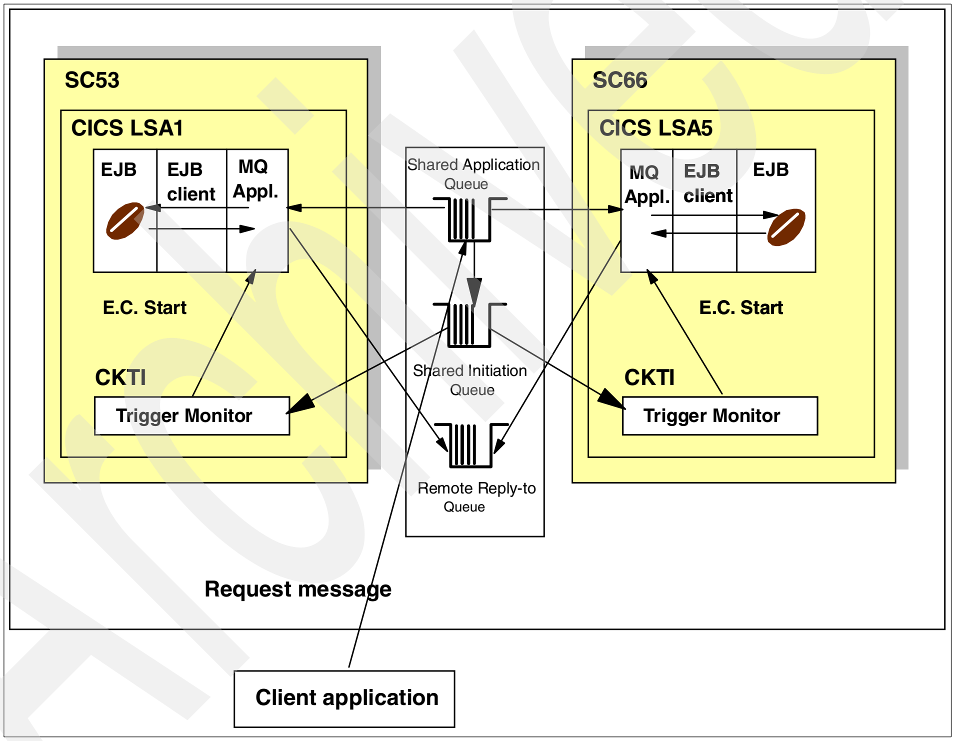


Figure 12-1 Back-end application environment

We use two z/OS images, SC53 and SC66. On each of the images, we run a WebSphere MQ 5.3 queue manager that is a member of queue-sharing group MQVG. The CICS application itself runs in CICS Transaction Server V2.2 regions LSA1 and LSA5. An alternative environment using non-shared application queues was tested in order to compare the triggering behavior with the shared queue environment. This environment is shown in Figure 12-2.

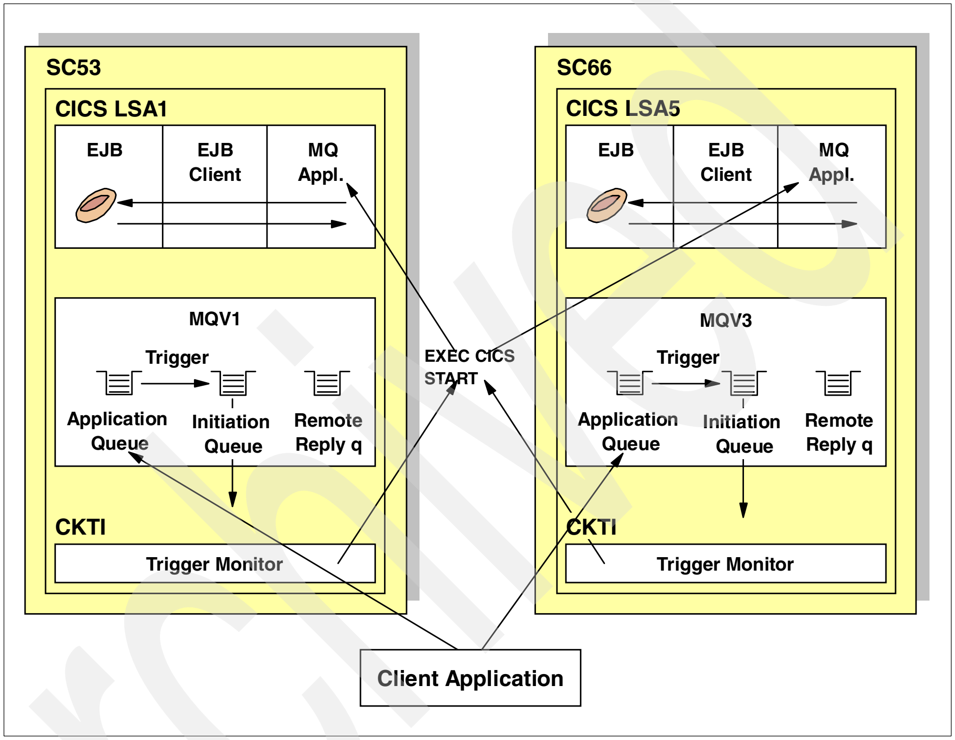


Figure 12-2 Back-end application environment using non-shared queues

12.1.1 Triggering considerations

To provide workload management functionality, we started one CICS trigger monitor (CKTI) instance per CICS Transaction Server region.

If there is only one CICS trigger monitor active in one of the CICS regions, we still could have used CICS-provided workload management facilities, such as dynamic routing for START commands. This can be done because the CICS trigger monitor is using an EXEC CICS START TRANSID command in order to start the application transaction. If the CICS application is to run in a non-shared queue environment, then one CICS trigger monitor instance is sufficient.

12.1.2 Queue definitions

We have defined a new set of application queue, initiation queue, remote reply-to queue, and process definition for the message driven CICS EJB application. They belong to queue-sharing group MQGV.

12.1.3 CICS sample EJB HelloWorldEJB

CICS Transaction Server 2.2 supplies the sample EJB HelloWorldEJB.jar file. Refer to the CICS Transaction Server V2.2 documentation and *Enterprise JavaBeans for z/OS and OS/390 CICS Transaction Server V2.2,* SG24-6284, to get full details on how to deploy the sample HelloWorldEJB.jar file.

12.1.4 Example source code and definitions

The following message driven CICS EJB application source files and sample definitions are available in the additional material provided with this redbook. They are packaged in the MDB.zip file.

- MQCICEJB.src - Assembler source of the WebSphere MQ user application. This is a CICS command-level application that issues an EXEC CICS LINK to the CICS EJB client JCICS program. It passes the COMMAREA that contains the message string.
- EJBCIC.java - JCICS Java program that acts as an EJB client. It gets passed to the COMMAREA containing the message text string and calls the CICS sample EJB.
- HelloWorldEJB.jar - The CICS-supplied sample EJB.
- CICSdefs.txt - CICS definitions that are needed to run the message driven CICS EJB application.
- WebSphere MQ definitions needed to run the message driven CICS EJB application mqobjects.txt.

12.2 Application flow and triggering considerations

In this section we discuss the following issues:

- We explain how we used WebSphere MQ triggering to start the CICS application. We describe how we used the CICS trigger monitor as well.
- We provide the definitions we made for triggering.
- We describe the front-end application that gets kicked off by the trigger monitor using an EXEC CICS START TRANSID(ECHO) command. It issues

the WebSphere MQ API commands and can optionally link to the CICS EJB client.

- Finally, we describe how we actually echo the WebSphere MQ message using the functionality of CICS EJBs.

Figure 12-3 illustrates the application flow.

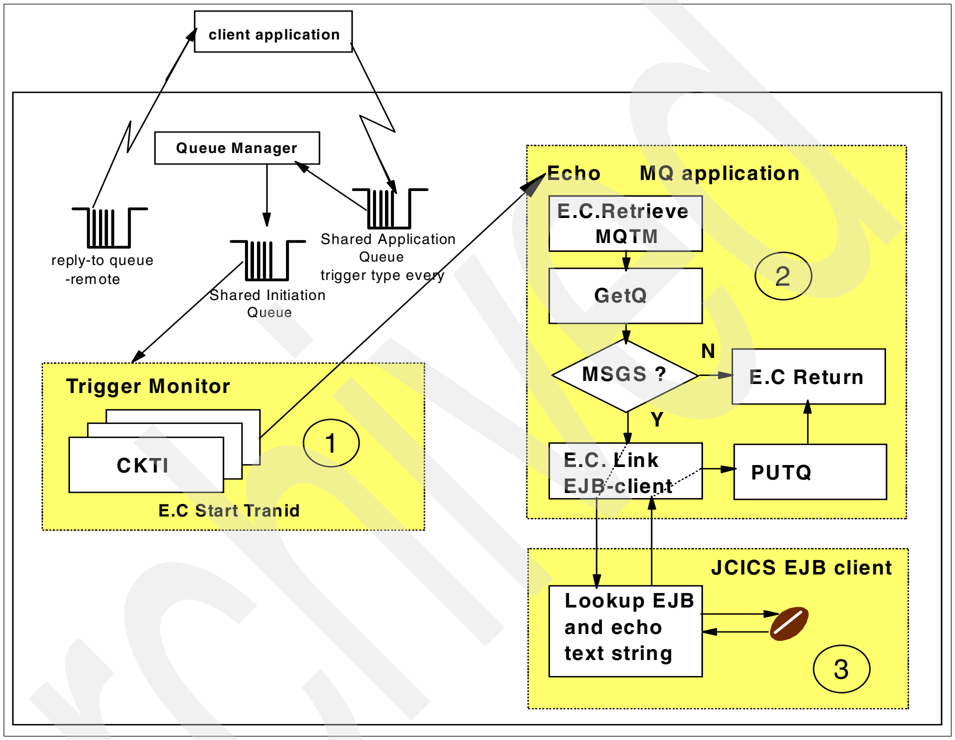


Figure 12-3 Message-driven CICS EJB application flow

12.2.1 WebSphere MQ triggering considerations

There are a number of triggering methods that can be utilized by a message-driven program running in a WebSphere MQ shared queues environment. They are TRIGTYPE=EVERY or TRIGTYPE=FIRST.

Each method has its advantages and disadvantages. The choice will be heavily influenced by the design of the program. Short-running transactions are usually associated with TRIGTYPE=EVERY, and long-running transactions are usually associated with TRIGYPE=FIRST. The implications on triggering for these application styles are discussed in the following sections.

Scenario # 1 Short-running transactions

For conventional short-running transactions, each processing a single message, specifying TRIGTYPE=EVERY is recommended. For each message put on to the application queue, a single trigger message is generated that will cause CKTI to start a new instance of the ECHO transaction by issuing the EXEC CICS START command. We also decided to control the workload distribution of ECHO by using two instances of CKTI, one on each CICS transaction server. In this scenario, the two CKTIs will share the trigger messages, resulting in a fairly even distribution of workload across the two CICS regions.

> **Note:** For TRIGTYPE=EVERY, the initiation queue must also be shared. This will ensure that only one trigger message is generated for each application message.
>
> If non-shared initiation queues are used, each CKTI will receive a trigger for the same message. Two ECHO transactions will be started but there is only one message. One of the ECHO transactions will terminate without being able to find a message. This unnecessary overhead can be avoided by simply ensuring the initiation queue is shared.

The ECHO transaction was designed to process single messages. The triggering method TRIGTYPE=EVERY was successfully used to distribute and process transactions across the two CICS regions in our application environment.

> **Important:** The only issue with TRIGTYPE=EVERY is that if an application abends, the message will be put back on the queue, but a trigger message is not regenerated. Application processes need to be put in place to ensure that this message is reprocessed appropriately. If the contents of the message itself are causing the abend, there are processes for "poisoned" messages that can be used. These processes are described in Chapter 5, "Handling program errors", in the *WebSphere MQ Application Programming Guide,* SC34-6064.

Scenario #2 Long-running transactions

Long-running or daemon-like transactions will process many messages. They typically read through and process messages on the application queue until it is empty. In this case, we do not need to generate triggers for every message. The only time a trigger is required is if there are no transactions currently active and a new message arrives. For this application type we would recommend using TRIGTYPE=FIRST. The first application message to arrive onto an empty queue will generate the trigger and start the transaction. In the setup shown in Figure 12-1 on page 226, two trigger messages (one for each queue manager with an active CKTI running) are generated, which will enable the transaction to be started in parallel on both CICS regions.

The behavior of these transactions will depend on the type of workload entering the system. If batches of messages are put to the application queue at regular (say hourly) intervals throughout the day, the program should be coded to process all the messages in the queue and then terminate. The next batch of messages would re-trigger the application.

If there is a sporadic workload where messages are arriving regularly throughout the day, but not continuously, the program should be coded to process all the available messages in the queue, and then wait a reasonable amount of time for the next message (say five minutes). The transaction would then be ready to process the next message when it arrives. If there are reasonably long periods of time during the day (lunch time) when no messages arrive, then the program could terminate to conserve CICS resources. Once again, the next message will retrigger the application.

A final option would be to have a true daemon application running continuously. In this case, the CKTI trigger monitor would be redundant. It would be easier to start the transaction as part of the CICS region startup.

Note: For TRIGTYPE=FIRST, the choice of a shared or non-shared initiation queue is not important. In both cases, a trigger message will be generated for each instance of CKTI.

Non-shared application queue considerations

We also distributed messages across two non-shared CICS application queues, one on each application queue manager. In this case the initiation queues were defined as non-shared on both queue managers. This message flow was used to see if the triggering behavior was consistent with the behavior when application queues are shared.

For TRIGTYPE=EVERY, the results were the same as for a shared queue. A single trigger was generated for each message arriving on the application queue, causing an instance of the ECHO transaction to be started.

For TRIGTYPE=FIRST, the results were also the same as for a shared queue. A trigger was generated when the first message was placed onto an empty queue. As we have two application queues, an ECHO transaction was started on both CICS regions, each reading messages from its own instance of the application queue.

Conclusions

The decision to use TRIGTYPE=EVERY or TRIGTYPE=FIRST will be determined by the design of the CICS application (short-running as opposed to daemon). Migrating from using multiple instances of a non-shared application queue to a shared application queue should cause no problems, since both trigger types function consistently with either setup.

12.2.2 Triggering definitions

The shared queues that we are using for our CICS application are illustrated in Example 12-1. We defined an application queue and associated it with an initiation queue and process definition.

Example 12-1 Shared application queue definition

```
DEFINE QLOCAL(CICS.MQVG.ECHO01)
...
INITQ(CICS.MQVG.INITQ01)
PROCESS(CICS.MQVG.ECHO01.PROCESS)
...
TRIGTYPE(FIRST)
```

The process definition specifies application-related data that is necessary to start the message driven CICS EJB application in CICS. Example 12-3 shows our process definition to start transaction ECHO, which is the transaction ID for WebSphere MQ user application MQCICEJB. Example 12-1 and Example 12-2 show how we have associated the application queue with the shared initiation queue that we are using.

Example 12-2 Shared initiation queue definition

```
DEFINE QLOCAL(CICS.MQVG.INITQ01)
...
```

The name of the reply-to queue that we used to reply the echo text string is derived from the message descriptor structure of the request message that the network client sends to the shared application queue.

Example 12-3 Process definition

```
DEFINE PROCESS (CICS.MQVG.ECHO01.PROCESS)
...
APPLTYPE (CICS)
APPLICID (ECHO)
...
```

Any network client, such as WebSphere MQ-supplied AMQSREQ or API Exerciser, can be used now to construct a message and place it on the application queue. The message descriptor of that message must contain the reply-to queue name that the CICS application will use to send the reply message to. The queue manager that is processing the trigger event checks to see if the conditions are met under which it has to generate a trigger message. We use TRIGTYPE=EVERY, so we create a trigger event every time a message arrives on the application queue. The queue manager creates a trigger message using the information held within the associated process definition and puts it on the initiation queue. One of the CICS trigger monitor CKTI instances retrieves the trigger message from the shared initiation queue. After that, the CICS trigger monitor starts the application using an EXEC CICS START TRANSID(ECHO) command.

12.2.3 WebSphere MQ user application MQCICEJB

Transaction ID ECHO is associated with WebSphere MQ user application MQCICEJB. The process flow of the assembler program is shown in Figure 12-3 on page 229. When MQCICEJB gets control, it first does an EXEC CICS RETRIEVE command in order to retrieve the trigger message structure. From the trigger message structure, we derive the application queue name. MQCICEJB does a MQGET request from the shared application queue and, if there is a message, stores the message in a data area. If there are no messages, we just return control to CICS.

If we have received a message from the application queue, we check the first character of the user data. If the value is “N”, we echo the message in this program and do not invoke the CICS enterprise bean.

If the value is not “N”, we issue an EXEC CICS LINK PROGRAM(EJBCIC) COMMAREA(CA) command to the JCICS EJB client program. When EJBCIC returns control, the COMMAREA contains the echo text string of the message as provided by the sample enterprise bean. The enterprise bean prefixes the original text string by “you said”.

MQCICEJB then derives the reply-to queue name from the message descriptor structure. The network client was responsible for making sure that there is a reply-to queue name as well as a reply-to-qmgr name in the message descriptor. MQCICEJB is using the reply-to queue name as well as the reply-to-qmgr name to send the echo reply message.

12.2.4 JCICS EJB client EJBCIC

EJB client applications written in traditional languages (Assembler, COBOL, PL1) cannot interact with an enterprise bean directly. The client interacts with the enterprise bean through intermediate objects that are created from classes generated by a deployment tool. Therefore, we need a Java client in order to provide access to the sample HelloWorldEJB.

We used a JCICS program to provide an EJB client. JCICS Java programs allow you to use most of the functions of the EXEC CICS API using the JCICS class library. MQCICEJB can therefore link to JCICS program EJBCIC and pass a COMMAREA. When EJBCIC gets control, it takes the string that was passed in the COMMAREA and uses it as an argument for the HelloWorldEJB's business method hello().

Example 12-4 shows how the JCICS EJBCIC client calls the HelloWorldEJB business method hello() with String argument `anewstring`. The message text string was copied from the COMMAREA. When the HelloWorldEJB enterprise bean returns the echoed message text string, we copy it back to the COMMAREA and return to MQCICEJB.

This sample shows how to get access to enterprise beans from programs written in CICS-supported languages, such as COBOL or Assembler.

Example 12-4 JCICS EJB client EJBCIC

```
public static void main(CommAreaHolder CAH)
  {
   String returnedTextstring = null;
     Task t            = null;
      byte MyCommArea[] = CAH.value;
      String anewstring = new String(CAH.value);
      try
      {
    CicsCallsEjb aCicsCallsEjb;
    aCicsCallsEjb = new CicsCallsEjb();



          t = Task.getTask();
          if (t == null)
          {
              System.err.println(" Can't get Task");
          }
          else
          {

              if (MyCommArea.length > 0)
              {
```

```
   System.out.println("input comarea" + anewstring);



   returnedTextstring =
aCicsCallsEjb.getHelloWorldAccessBean1fromJcics().hello(anewstring);



        CAH.value = returnedTextstring.getBytes();
                }
                else
       {
             t.out.println("Transaction invoked without a CommArea");
           }
       }
        }
        catch (Throwable th)
        {
            System.err.println("Failed to read COMMAREA");
        }
        finally
        {
            if (t != null)
            {
                System.out.println("Leaving main()");
            }
        }
    }
```

12.2.5 EJB client program flow

Like any other EJB client, the JCICS program EJBCIC needs to locate the enterprise bean first. It uses a JNDI lookup call to a naming service. If the enterprise beans JNDI name can be found, the JNDI call returns a reference to an object implementing the EJB's home interface. The JCICS client has been developed using VisualAge for Java V4.0, which generates access beans to allow clients to interact with EJBs. Example 12-5 shows how we used an access bean to look up the sample HelloWorldEJB.

Example 12-5 HelloWorldAccessBean

```
private HelloWorldAccessBean getHelloWorldAccessBean1fromJcics() {
   if (xyzHelloWorldAccessBean1 == null) {
      try {
      xyzHelloWorldAccessBean1 = new HelloWorldAccessBean();
      xyzHelloWorldAccessBean1.setInit_JNDIName("HelloWorld");
```

```
            try {

               String sRegionSysid = Region.getSYSID();
               String sRegionApplid = Region.getAPPLID();



xyzHelloWorldAccessBean1.setInit_JNDIName(sRegionSysid+"/"+sRegionApplid+"/Hell
oWorld");

xyzHelloWorldAccessBean1.setInit_GlobalNameServiceTypeName("com.sun.jndi.cosnam
ing.CNCtxFactory");

xyzHelloWorldAccessBean1.setInit_GlobalNameServiceURLName("iiop://9.24.105.29:9
00");}
            catch (Exception e)

            {
               System.out.println("some kind of error happened");
            }



         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end

      }
   }
   return xyzHelloWorldAccessBean1;
}
```

Basically what we need to locate the bean is the JNDI name, the location of the naming service, and the naming service type. The JNDI name, in our case, is prefixed by the CICS SYSID plus CICS APPLID. Our JNDI name for the sample HelloWorldEJB would therefore be LSA1/SCSCLSA1/HelloWorld. The naming service stores the name in its directory structure when CICS initiates a publish process for the enterprise bean.

Having received an object reference to the HelloWorld enterprise bean's home interface, the JCICS client invokes the create() method on the enterprise bean's home interface. The JCICS client uses the home interface to look up existing HelloWorld EJB instances or to create new ones. Invoking the create() method causes the container to drive the ejbCreate() method. An object that implements the HelloWorld EJB remote interface is created and the correct reference is returned to the JCICS client. When the JCICS client has a reference to an object implementing the HelloWorld EJB's remote interface, the business method hello() of the Helloworld EJB can be invoked.

The JCICS client has access to the business methods through the remote interface.

When the JCICS client drives the remove() method, the remote HelloWorld EJB object is removed.

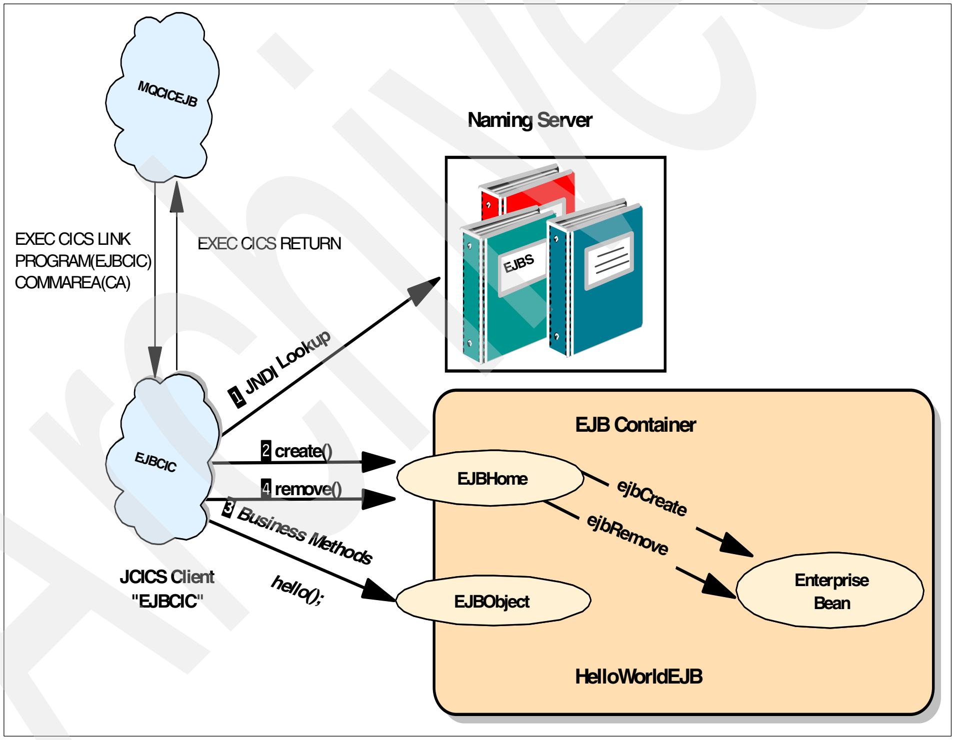


Figure 12-4 Sample EJB HelloWorldEJB control flow

12.3 Workload generation scripts

We created several scripts in order to generate some workload for the CICS ECHO transaction running in MQV1 and MQV3. These scripts deliver messages to either a shared application queue residing in the coupling facility or to cloned non-shared queues on the application queue managers MQV1 and MQV3. They were also used to verify the high availability configurations as described in Chapter 11, “High availability setup” on page 209. The scripts generate flows of request and reply messages that use various combinations of shared and clustered channels. The scenarios documented in Chapter 15, “Recovery scenarios” on page 295 make use of these scripts to simulate a system under heavy load. The following section explains in detail how these scripts work. Links to the sample scripts and WebSphere MQ object definitions can be found in Appendix B, “Additional material” on page 339. The WebSphere MQ object definitions can also be found in Example 12-7 on page 244.

Naming standard for scripts

All script names are of the form MQREQxyzz as shown in Example 12-6.

Example 12-6 Workload script name format

```
MQREQ  - common prefix for all scripts

x         Number of requests to generate.
x=1       single message,
x=5k      5000 messages,
x=wl      constant workload.
This script sends 5000 messages and then re-submits itself until terminated by
typing (Ctrl +C)
y          Method of delivery to gateway.
y=s        shared inbound channel(short reply)  ,
y=l        shared inbound channel(long reply),
y=c        cluster channel

zz       (Only used when y = C)
zz=ss     short request,short reply
zz=sl     short request,long reply,
zz=ls     long request,short reply
zz=ll    long request,long reply
```

“Short” refers to messages less than 63 KB, which can reside on a shared queue. “Long” refers to messages more than 63 KB, which need a traditional mover to deliver them to their final destination.

In all cases, the CICS ECHO transaction will respond to the queue as specified in the REPLYTOQ field of the message descriptor.

Detailed description of scripts

These scripts are set up to run with a queue manager that has the name %hostname%. This variable will be resolved to the local host name of the Windows server. If you wish to use a queue manager whose name does not follow this convention, then the scripts as supplied will need to be altered.

- **Message Flow #1**
  - MQREQ1S/MQREQ5KS/MQREQWLS
  - Request (Requests to CICS.MQVG.ECHO01 - shared application queue)
  - Replies (Replies are returned using a shared transmit queue and shared outbound channel)
  - This script uses a remote queue CICS.MQVG.ECHO01.SHAREDQ.REQ to target the shared application queue CICS.MQVG.ECHO01 via a single gateway. A shared inbound channel (%hostname% .MQVG) is used to deliver the request to the queue-sharing group. This channel will be connected to one of the two gateway queue managers MQV2 or MQV4 as determined by the Sysplex Distributor.
  - After all requests are sent, the script waits for the replies on local queue CICS.MQVG.ECHO01.SHAREDQ.SHORT.REPLY. The replies are delivered by a shared outbound channel MQVG.%hostname% using a shared transmit queue %hostname%. This channel will be started on the most suitable channel initiator in the QSG based on current workload (that is, number of active channels).
- **Message Flow #2**
  - MQREQ1L/MQREQ5KL/MQREQWLL
  - Request (Requests to CICS.MQVG.ECHO01 - shared application queue)
  - Replies (Replies are returned using a private transmit queue and conventional movers)
  - This script uses a remote queue CICS.MQVG.ECHO01.SHAREDQ.REQ to target the shared application queueCICS.MQVG.ECHO01 via a single gateway. A shared inbound channel (%hostname% .MQVG) is used to deliver the request to the queue-sharing group. This channel will be connected to one of the 2 gateway queue managers MQV2 or MQV4 as determined by the Sysplex Distributor.
  - After all requests are sent, the script waits for the replies on local queue CICS.MQVG.ECHO01.SHAREDQ.LONG.REPLY.

Note: The application is coded to look for the string “LONG” in the REPLYTOQ name. If found, it will use a REPLYTOQMGR alias (%hostname%.ALIAS) when opening the reply queue, instead of the REPLYTOQMGR as supplied in the message descriptor. This will resolve to a private transmit queue (MQV2 or MQV4), depending on which application queue manager the CICS application is connected to) and the reply will be sent back to one of the gateways using a traditional private channel (MQV3.MQV2, or MQV1.MQV4), rather than resolving to the shared transmit queue. When the reply arrives on the gateway, it will be resolved by another REPLYTOQMGR alias to private transmit queue %hostname%.XMITQ and private channel MQV2.%hostname%, which will send it back to the client.

- **Message Flow #3**
  - MQREQ1CSS/MQREQ5KCSS/MQREQWLCSS
  - Request (Requests to CICS.MQVG.ECHO01 - shared application queue).
  - Replies (Replies are returned using a shared transmit queue and shared outbound channel)
  - This script uses a clustered queue alias CICS.MQVG.ECHO01.SHORT.REQ to target the shared application queue CICS.MQVG.ECHO01 via both gateways. This alias is only advertised on the gateway queue managers to prevent connections being made directly from the client to the application queue managers. Cluster sender channels (TO.MQV2 and TO.MQV4) are used to deliver the request to the queue-sharing group.
  - After all requests are sent, the script waits for the replies on local queue CICS.MQVG.ECHO01.CLUSTER.SHORT.REPLY. The replies are delivered by a shared outbound channel MQVG.%hostname% using a shared transmit %hostname%. This channel will be started on the most suitable channel initiator in the QSG based on current workload (that is, number of active channels).
- **Message Flow #4**
  - MQREQ1CSL/MQREQ5KCSL/MQREQWLCSL
  - Request (Requests to CICS.MQVG.ECHO01- shared application queue).
  - Replies (Replies are returned using a private transmit queue and conventional movers)
  - This script uses a clustered queue alias CICS.MQVG.ECHO01.SHORT.REQ to target the shared application queue CICS.MQVG.ECHO01 via both gateways. This alias is only

advertised on the gateway queue managers to prevent connections being made directly from the client to the application queue managers. Cluster sender channels (TO.MQV2 and TO.MQV4) are used to deliver the request to the queue-sharing group.

– After all requests are sent, the script waits for the replies on local queue CICS.MQVG.ECHO01.CLUSTER.LONG.REPLY.

> **Note:** The application is coded to look for the string “LONG” in the REPLYTOQ name. If found, it will use a REPLYTOQMGR alias (%hostname%.ALIAS) when opening the reply queue, instead of the REPLYTOQMGR as supplied in the message descriptor. This will resolve to a private transmit queue (MQV2 or MQV4), depending on which application queue manager the CICS app is connected to) and the reply will be sent back to one of the gateways using a traditional private channel (MQV3.MQV2, or MQV1.MQV4), rather than resolving to the shared transmit queue. When the reply arrives on the gateway, it will be resolved by another REPLYTOQMGR alias to private transmit queue %hostname%.XMITQ and private channel MQV2.%hostname%, which will send it back to the client

- **Message Flow #5**
  - MQREQ1CLS/MQREQ5KCLS/MQREQWLCLS
  - Request (Requests to CICS.MQVG.ECHO01.LONG - non-shared application queue with instances on both application queue managers)
  - Replies (Replies are returned using a shared transmit queue and shared outbound channel)
  - This script uses a clustered queue alias CICS.MQVG.ECHO01.LONG.REQ to target the non-shared application queues CICS.MQVG.ECHO01.LONG through both gateways. This alias is only advertised on the gateway queue managers to prevent connections being made directly from the client to the application queue managers. Cluster sender channels (TO.MQV2 and TO.MQV4) are used to deliver the request to the queue-sharing group.
  - After all requests are sent, the script waits for the replies on local queue CICS.MQVG.ECHO01.CLUSTER.SHORT.REPLY.
- **Message Flow #6**
  - MQREQ1CLL/MQREQ5KCLL/MQREQWLCLL
  - Request (Requests to CICS.MQVG.ECHO01.LONG (non-shared application queue with instances on both application queue managers)

– Replies (Replies are returned using a private transmit queue and conventional movers)

– This script uses a clustered queue alias CICS.MQVG.ECHO01.LONG.REQ to target the non-shared application queues CICS.MQVG.ECHO01.LONG via both gateways. This alias is only advertised on the gateway queue managers to prevent connections being made directly from the client to the application queue managers. Cluster sender channels (TO.MQV2 and TO.MQV4) are used to deliver the requests to the queue-sharing group.

– After all requests are sent, the script waits for the replies on local queue CICS.MQVG.ECHO01.CLUSTER.LONG.REPLY.

> **Note:** The application is coded to look for the string "LONG" in the REPLYTOQ name. If found, it will use a REPLYTOQMGR alias (%hostname%.ALIAS) when opening the reply queue, instead of the REPLYTOQMGR as supplied in the message descriptor. This will resolve to a private transmit queue (MQV2 or MQV4), depending on which application queue manager the CICS application is connected to, and the reply will be sent back to one of the gateways using a traditional private channel (MQV3.MQV2, or MQV1.MQV4), rather than resolving to the shared transmit queue. When the reply arrives on the gateway, it will be resolved by another REPLYTOQMGR alias to private transmit queue %hostname%.XMITQ and private channel MQV2.%hostname%, which will send it back to the client.

The channel topologies used by these scripts are shown in Figure 12-5 on page 243, which shows the channels used to deliver messages and replies using shared application and transmission queues, and Figure 12-6 on page 244, which shows the channels used to deliver messages and replies using non-shared queues.

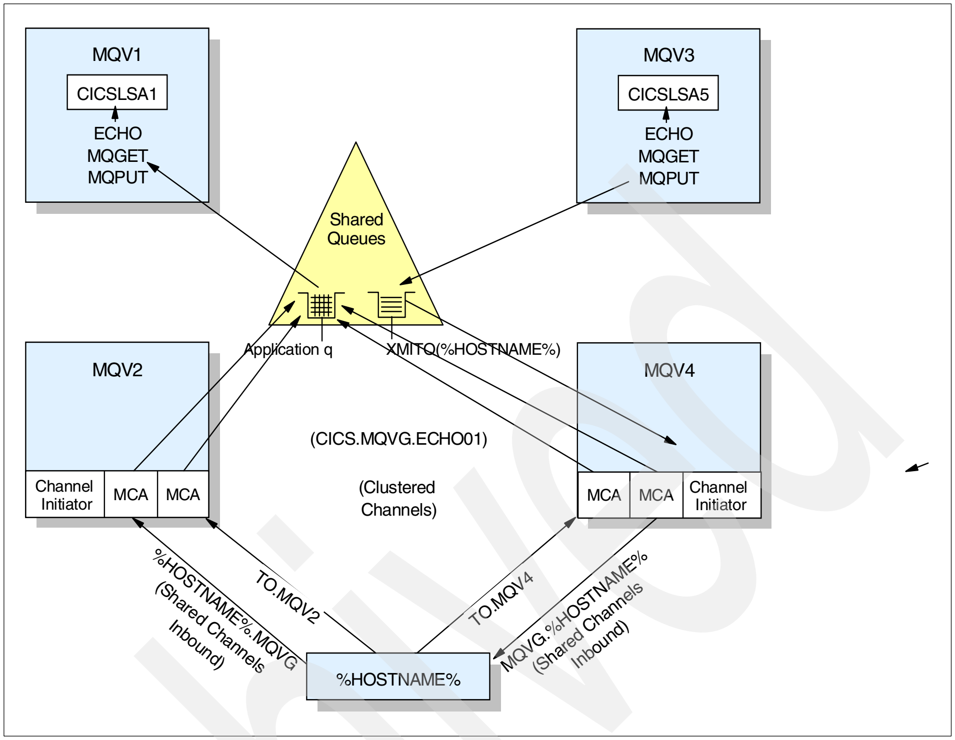


Figure 12-5 Channel topology (shared queues)

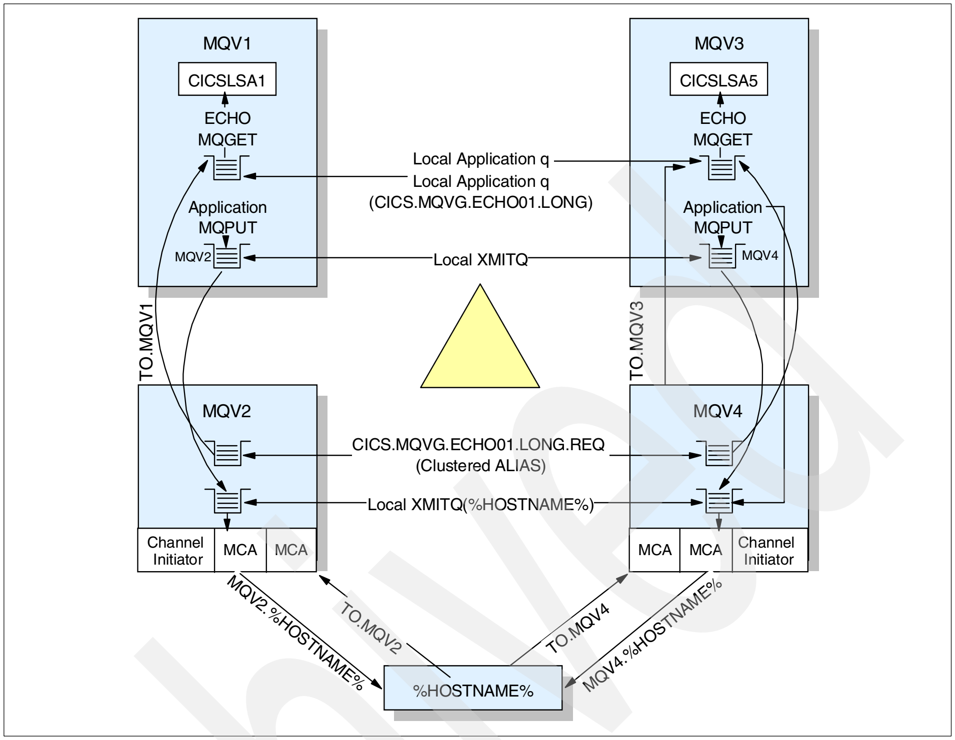


Figure 12-6 Channel topology (non-shared queues)

Example 12-7 MQ objects required to run workload scripts

```
Gateway queue manager MQV2
DEFINE NOREPLACE                                                   -
 QLOCAL('CICS.MQVG.ECHO01')                                        -
 QSGDISP(SHARED)                                                   -
 STGCLASS('DEFAULT')                                               -
 CFSTRUCT('APPLICATION1')                                          -
 DESCR('CICS shared application queue')
 PROCESS('CICS.MQVG.ECHO01.PROCESS')                               -
 TRIGGER                                                           -
 MAXMSGL(10000)                                                    -
 INITQ('CICS.MQVG.INITQ01')                                        -
 USAGE(NORMAL)                                                     -
 SHARE                                                             -
 DEFSOPT(SHARED)                                                   -
 TRIGTYPE(EVERY)                                                   -
 TRIGDPTH(1)                                                       -
 TRIGMPRI(0)                                                       -
 TRIGDATA(' ')
```

```
 DEFINE NOREPLACE                                                         -
 QLOCAL('CICS.MQVG.INITQ01')                                             -
 QSGDISP(SHARED)                                                         -
 STGCLASS('DEFAULT')                                                     -
 CFSTRUCT('APPLICATION1')                                                -
 DESCR('MQVG INITIATION QUEUE FOR BACKEND CICS REGIONS')                 -
 PUT(ENABLED)                                                            -
 DEFPRTY(0)                                                              -
 DEFPSIST(YES)                                                           -
 MAXDEPTH(99999999)

DEFINE NOREPLACE                                                         -
 PROCESS('CICS.MQVG.ECHO01.PROCESS')                                     -
 QSGDISP(GROUP)                                                           -
 DESCR('ECHO 01 APPLICATION PROCESS')                                    -
 APPLTYPE(CICS)                                                          -
 APPLICID('ECHO START')                                                  -
 USERDATA('WORK')                                                        -
 ENVRDATA(' ')
DEFINE NOREPLACE                                                         -
 QLOCAL('%hostname%')                                                    -
 QSGDISP(SHARED)                                                         -
 STGCLASS('DEFAULT')                                                     -
 CFSTRUCT('APPLICATION1')                                                -
 DESCR('Shared Xmit queue to %hostname%')                                -
 TRIGGER                                                                 -
 INITQ('SYSTEM.CHANNEL.INITQ')                                           -
 USAGE(XMITQ)                                                            -
 TRIGTYPE(FIRST)                                                         -
 TRIGDATA('MQVG.%hostname%')                                             -
DEFINE NOREPLACE
 QLOCAL('%hostname%.XMITQ')                                              -
 QSGDISP(QMGR)                                                           -
 DESCR('Xmit q to %hostname%')                                           -
 INITQ('SYSTEM.CHANNEL.INITQ')                                           -
 USAGE(XMITQ)                                                            -
 TRIGGER
 TRIGTYPE(FIRST)                                                         -
 TRIGDATA('MQV2.%hostname%')
DEFINE NOREPLACE                                                         -
 QREMOTE('%hostname%.alias')                                             -
 QSGDISP(QMGR)                                                           -
 DESCR('Queue manager alias for         %hostname%')                     -
 PUT(ENABLED)                                                            -
 RNAME(' ')                                                              -
 RQMNAME('%hostname%')                                                   -
 XMITQ('%hostname%.XMITQ')                                               -
DEFINE NOREPLACE
 CHANNEL('MQVG.%hostname%')                                              -
```

```
 CHLTYPE(SDR)                                                          -
 QSGDISP(GROUP)                                                        -
 XMITQ('%hostname%')                                                   -
 TRPTYPE(TCP)                                                          -
 CONNAME('9.24.105.29(1415)')                                          -
 DESCR('Shared sender to %hostname%')                                  -
DEFINE NOREPLACE
 CHANNEL('MQV2.%hostname%')                                            -
 CHLTYPE(SDR)                                                          -
 QSGDISP(QMGR)                                                         -
 XMITQ('%hostname%.XMITQ')                                             -
 TRPTYPE(TCP)                                                          -
 CONNAME('9.24.105.29(1415)')
DESCR('Sender to %hostname%')

Gateway queue manager MQV4

DEFINE NOREPLACE                                                       -
 QLOCAL('%hostname%.XMITQ')                                            -
 QSGDISP(QMGR)                                                         -
 DESCR('Xmit q to %hostname%')                                         -
 INITQ('SYSTEM.CHANNEL.INITQ')                                         -
 USAGE(XMITQ)                                                          -
 TRIGGER
 TRIGTYPE(FIRST)                                                       -
 TRIGDATA('MQV4.%hostname%')                                           -
DEFINE NOREPLACE
 QREMOTE('%hostname%.alias')                                           -
 QSGDISP(QMGR)                                                         -
 DESCR('Queue manager alias for         %hostname%')                   -
 PUT(ENABLED)                                                          -
 RNAME(' ')                                                            -
 RQMNAME('%hostname%')                                                 -
 XMITQ('%hostname%.XMITQ')
DEFINE NOREPLACE                                                       -
 CHANNEL('MQV4.%hostname%')                                            -
 CHLTYPE(SDR)                                                          -
 QSGDISP(QMGR)                                                         -
 XMITQ('%hostname%.XMITQ')                                             -
 TRPTYPE(TCP)                                                          -
 CONNAME('9.24.105.29(1415)')                                          -
 DESCR('Sender to %hostname%')

Application  queue manager MQV1

DEFINE NOREPLACE                                                       -
 QLOCAL('MQV4')                                                        -
 QSGDISP(QMGR)                                                         -
 DESCR('Xmitq to gateway MQV4')                                        -
```

```
 TRIGGER                                                          -
 INITQ('SYSTEM.CHANNEL.INITQ')                                    -
 USAGE(XMITQ)                                                     -
 NOSHARE                                                          -
 DEFSOPT(EXCL)                                                    -
 MSGDLVSQ(PRIORITY)
DEFINE NOREPLACE                                                  -
 QREMOTE('%hostname%.alias')                                      -
 QSGDISP(QMGR)                                                    -
 DESCR('Qmgr alias for %hostname%')                               -
 PUT(ENABLED)                                                     -
 DEFPRTY(0)                                                       -
 DEFPSIST(NO)                                                     -
 RNAME(' ')                                                       -
 RQMNAME('%hostname%.alias')                                      -
 XMITQ('MQV4')
DEFINE NOREPLACE                                                  -
 CHANNEL('MQV1.MQV4')                                             -
 CHLTYPE(SDR)                                                     -
 QSGDISP(QMGR)                                                    -
 XMITQ('MQV4')                                                    -
 TRPTYPE(TCP)                                                     -
 CONNAME('9.12.6.84(1500)')                                       -
 DESCR('Explicit channel to gateway MQV4 for CICS echo replies')  -
 DISCINT(0)

Application  queue manager MQV3

DEFINE NOREPLACE                                                  -
 QLOCAL('MQV2')                                                   -
 QSGDISP(QMGR)                                                    -
 DESCR('Xmitq to gateway MQV2')                                   -
 TRIGGER                                                          -
 INITQ('SYSTEM.CHANNEL.INITQ')                                    -
 USAGE(XMITQ)                                                     -
 NOSHARE                                                          -
 DEFSOPT(EXCL)                                                    -
 MSGDLVSQ(PRIORITY)
DEFINE NOREPLACE                                                  -
 QREMOTE('M23X2517.alias')                                        -
 QSGDISP(QMGR)                                                    -
 DESCR('Qmgr alias for M23X2517')                                 -
 PUT(ENABLED)                                                     -
 DEFPRTY(0)                                                       -
 DEFPSIST(NO)                                                     -
 RNAME(' ')                                                       -
 RQMNAME('M23X2517.alias')                                        -
 XMITQ('MQV2')
DEFINE NOREPLACE                                                  -
```

```
 CHANNEL('MQV3.MQV2')                                                   -
 CHLTYPE(SDR)                                                           -
 QSGDISP(QMGR)                                                          -
 XMITQ('MQV2')                                                          -
 TRPTYPE(TCP)                                                           -
 CONNAME('9.12.6.85(1500)')                                             -
 DESCR('Explicit channel to gateway MQV2 for CICS echo replies')        -
 DISCINT(0)

Remote queue manager %hostname%

DEFINE QLOCAL ('CICS.MQVG.ECHO01.CLUSTER.LONG.REPLY')
DEFINE QLOCAL ('CICS.MQVG.ECHO01.CLUSTER.SHORT.REPLY')
DEFINE QLOCAL ('CICS.MQVG.ECHO01.SHAREDQ.LONG.REPLY')
DEFINE QLOCAL ('CICS.MQVG.ECHO01.SHAREDQ.SHORT.REPLY')
DEFINE QLOCAL ('MQVG') +
       DESCR('Xmitq to MQVG - z/OS V5.3') +
       TRIGGER +
       TRIGTYPE(FIRST) +
       TRIGDPTH(1) +
       TRIGMPRI(0) +
       TRIGDATA('%hostname.MQVG') +
       PROCESS(' ') +
       INITQ('SYSTEM.CHANNEL.INITQ')
DEFINE QREMOTE ('CICS.MQVG.ECHO01.SHAREDQ.REQ') +
       XMITQ('MQVG') +
       RNAME('CICS.MQVG.ECHO01') +
       RQMNAME('MQVG')
DEFINE CHANNEL ('M23X2517.MQVG') CHLTYPE(SDR) +
       TRPTYPE(TCP) +
       BATCHINT(0) +
       BATCHSZ(1) +
       CONNAME('9.12.8.10(1501)') +
       CONVERT(YES) +
       XMITQ('MQVG')
DEFINE CHANNEL ('MQV2.%hostname%') CHLTYPE(RCVR) +
       TRPTYPE(TCP)
DEFINE CHANNEL ('MQV4.%hostname%') CHLTYPE(RCVR) +
       TRPTYPE(TCP)
DEFINE CHANNEL ('MQVG.%hostname%') CHLTYPE(RCVR) +
       TRPTYPE(TCP)
DEFINE CHANNEL ('TO.%hostname%') CHLTYPE(CLUSRCVR) +
       TRPTYPE(TCP) +
       CLUSTER('CLUSPLX1') +
       CONNAME('9.24.105.29(1415)') +
       CONVERT(NO)
DEFINE CHANNEL ('TO.MQV2') CHLTYPE(CLUSSDR) +
       TRPTYPE(TCP) +
       CLUSTER('CLUSPLX1') +
```

```
CONNAME('9.12.6.85(1500)') +
CONVERT(YES)
```

13

Other WebSphere MQ applications in a Parallel Sysplex environment

The high availability features of WebSphere MQ in a Parallel Sysplex environment can be used by a variety of applications, not only CICS applications. In this chapter, we overview the family of MQ-based IBM business integration products that can fully enjoy the z/OS strengths. We also discuss limitations on use of shared queues that apply to Information Management System (IMS).

13.1 WebSphere MQ family overview

WebSphere MQ (formerly known as MQSeries) provides support for applications with a number of application programming interfaces (Message Queuing Interface (MQI), Application Messaging Interface (AMI), Java Messaging Services (JMS)) in several programming languages and for a number of communication models (including point-to-point and publish/subscribe). WebSphere MQ also provides a number of connectors and gateways to other products such as Lotus Domino, Microsoft Exchange, SAP/R3, CICS, and IMS.

The WebSphere MQ family of products contains three WebSphere MQ Broker products, the MQSeries Workflow product, and a number of adapters and connectors.

13.1.1 WebSphere MQ Broker products

These products extend the messaging capabilities of WebSphere MQ by adding message routing and transformation capabilities driven by business rules.

Business rules can be defined by graphically developing *message flows,* which consist of a sequence of *nodes*. Message flows are processes that can encapsulate logic ranging from simple to extremely complex. They can be designed to perform a wide variety of functions including (but not limited to):

- Routing of messages to zero or more destinations based on the contents of the message or message header (both one-to-many and many-to-one messaging topologies are supported).
- Transformation of messages into different formats so that diverse applications can exchange messages that each of them can understand.
- Enrichment of the message content en-route (for example, by using a database lookup performed by the message broker).
- Storing information extracted from messages en-route to a database (using the message broker to perform this).
- Publishing messages (and using topic/content based criteria for subscribers to select which messages to receive).
- Interfacing with other transport mechanisms such as MQSeries Everyplace.
- Extending the base functionality with *plug-in* nodes (which can be developed by customers as well as by IBM and ISVs).

Message content can be handled in a number of message domains, including:

- XML domain, which handles self-defining (or generic) XML messages.

- Message Repository Manager (MRM) domain, which handles messages predefined in *message sets* in the message repository database, as:
  - XML (defined)
  - Tagged data
  - Custom Wire Format (similar to a COBOL copybook layout)
  - Delimited data
  - PDF (not Adobe, but a specialized financial data format).
- BLOB domain, which handles messages with no predefined structure.

There are currently three WebSphere MQ Broker products:

- WebSphere MQ Integrator (WMQI), formerly known as MQSeries Integrator, is the full function product that includes the New Era of Networks (NEON) component that provides connectivity to NEON adapters and backward compatibility with previous releases.
- WebSphere MQ Integrator Broker (WMQIB) does not contain the NEON component and it is, therefore, more attractively priced for the customer. Both WMQI and WMQIB are available on a variety of Windows and UNIX platforms as well as on z/OS and OS/390.
- WebSphere MQ Event Broker a new member of the WebSphere MQ famiy is a high speed publish/subscribe broker delivering real-time event-based information to people, applications and devices. WMQ Event Broker extends the solution through real-time event integration, with high performance topic and content publish and subscribe support and routing. It provides a simple step up from point-to-point integration to basic pub/sub capacity. Real-time alerts integrate with telemetry and mobile technologies, and it has an Internet protocol transport in addition to those in WebSphere MQ Integrator Broker. Currently, there is no 390 version of this product.

Figure 13-1 on page 254 illustrates the scalable degrees of function of WebSphere MQ Brokers.

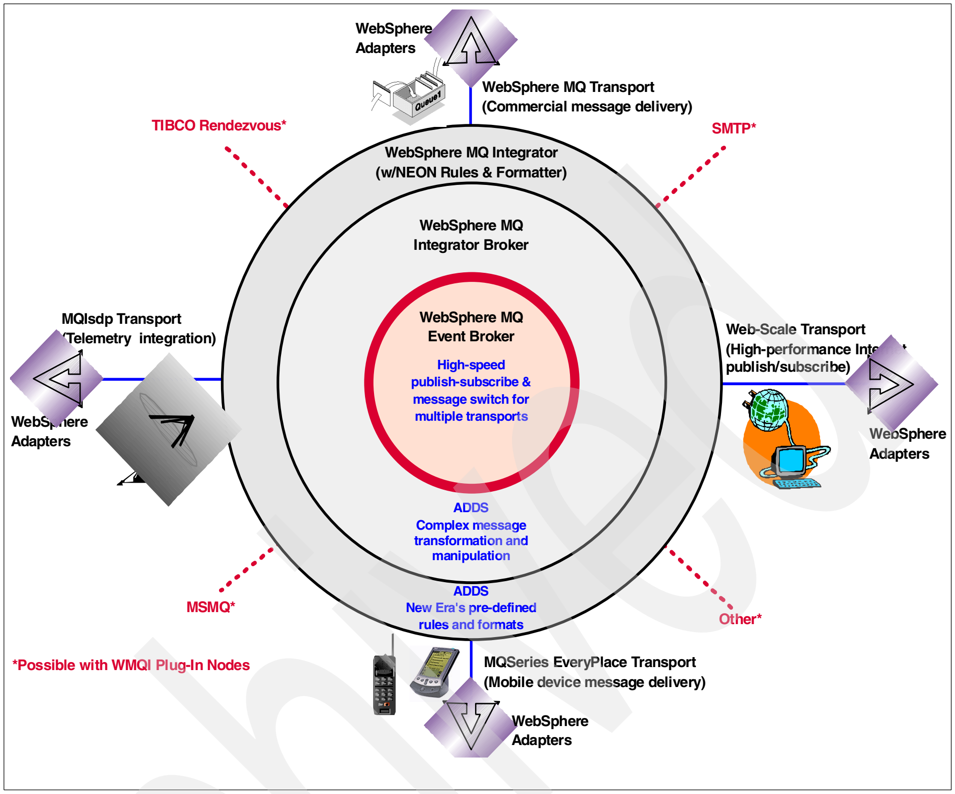


Figure 13-1 WebSphere MQ Brokers -- Scalable degrees of function

Table 13-1 provides feature comparison between members of WebSphere MQ Broker family.

Table 13-1 WebSphere MQ Broker feature comparison

Feature	WMQEB 2.1	WMQIB 2.1	WMQI 2.1
High Performance Matching Eng	X	X	X
Content Based Pub/Sub	X*	X	X
Subject Based Pub/Sub	X	X	X
HTTP Tunneling	X**		
SSL Authentication	X**		
SSL Encryption	X**		

Feature	WMQEB 2.1	WMQIB 2.1	WMQI 2.1
High Availability Topologies	X**		
IP Transport (JMS Interface)	X		
WMQ Transport	X	X	X
WMQE Transport	X	X	X
Telemetry Transport	X	X	X
Graphical Tool Interface	X	X	X
Admin Control Center	X	X	X
Database Support	X	X	X
Multi Broker Topologies	X	X	X
Content Based Routing	X	X	X
MRM Repository		X	X
Formatting		X	X
Complex Message Flows		X	X
New Era of Networks Technology			X
Multicast	X**		
Entry License Option		X	

Notes:

* XML only

** SupportPac

13.1.2 MQSeries Workflow

MQSeries Workflow (MQWF) is the third component of IBM's family of products for business integration. MQSeries Workflow is the process engine for IBM Business Process Management (BPM) software. MQWF executes process definitions captured during modeling and ensures that business functions are performed accurately and reliably using the transactional integrity of WebSphere MQ. MQWF facilitates the integration of services provided by IT and organizational infrastructures, integrating bought and built applications - even with customers, business partners, suppliers and employees. MQWF can transform traditional business models into e-business models.

13.2 WMQI and WMQIB for z/OS

WMQI and WMQIB for z/OS V2.1 extend the MQ message broker to the z/OS platform. These products are designed to allow customers to leverage existing hardware and existing OS/390 and z/OS administrative skills to provide a highly available, robust messaging environment while taking advantage of the strengths of the mainframe. Customers will choose a mainframe environment:

- When applications require highest availability
- When high levels of I/O are required
- When they desire to consolidate numerous smaller servers
- When the application benefits from proximity to mainframe-based data or application components

Advantages of WebSphere MQ Integrator for z/OS and WebSphere MQ Integrator Broker on z/OS are:

- Quality of Service
  - Scalability:
    - Parallel Sysplex exploitation
  - Availability:
    - Workload Manager exploitation
    - Automatic Restart Manager
    - Geographically distributed failover
    - Highest reliability server hardware with industry's most advanced self-healing and recovery capabilities
  - Data integrity
    - Integrated transaction support with other z/OS Work Managers
    - Transactional services based on RRS

        - Two-phase distributed transactions spanning CICS, IMS and DB2
        - Sysplex-wide data sharing
    - Data protection and security
        - Integrated security via System Authorization Facility
        - Security integration with RACF
        - Strongest inter process isolation and system security
    - Performance
        - Optimization of communication protocols
        - Asynchronous socket I/O parallelism
        - IIOP optimizations to exploit local communications within a sysplex cluster
- Proximity to enterprise data and applications
    - Short path length to enterprise data and applications
    - Gigabit I/O access to enterprise data and applications
    - Enablement of single unit of work (UOW) management of assets
- Administration and management
    - Processing administration and operations matured and refined over decades
    - Geared to production-level, high transaction volume, mission-critical processing
    - Practical 24x7 operation
    - Tools refined to require fewer people to manage large, complex environments
    - Tools refined to quickly determine, isolate, circumvent, and repair problems
    - Well-established problem and change management procedures
    - Ability to tailor many processing environments on one box for specific applications

13.2.1 WMQI and WMQIB applications

These applications can be existing WebSphere MQ-enabled applications that do not need to be aware of the broker. In this case, the message flows will set the correct configuration defined to handle the messages. Alternatively the application can prefix the user data of the message with a WebSphere MQ Integrator header (MQRFH or MQRFH2) to control the way the broker handles the message.

WebSphere MQ Integrator and WebSphere MQ Integrator Broker support two models for application communication:

- Point to Point - which can include one to many and many to one (that is, where the number of message producers and consumers are different)
- Publish/Subscribe - which includes and extends the existing WebSphere MQ publish/subscribe model

13.2.2 WMQI and WMQIB in a Parallel Sysplex

The high-availability features of WMQI and WMQIB in a Parallel Sysplex environment rely upon base WebSphere MQ support discussed in the previous chapters. A WMQI or a WMQIB broker is a WebSphere MQ application. It is connected to a specific queue manager and executes message flows. Our goal is to achieve high availability for these message flows.

Figure 13-2 depicts a configuration with multiple brokers in a Parallel Sysplex.

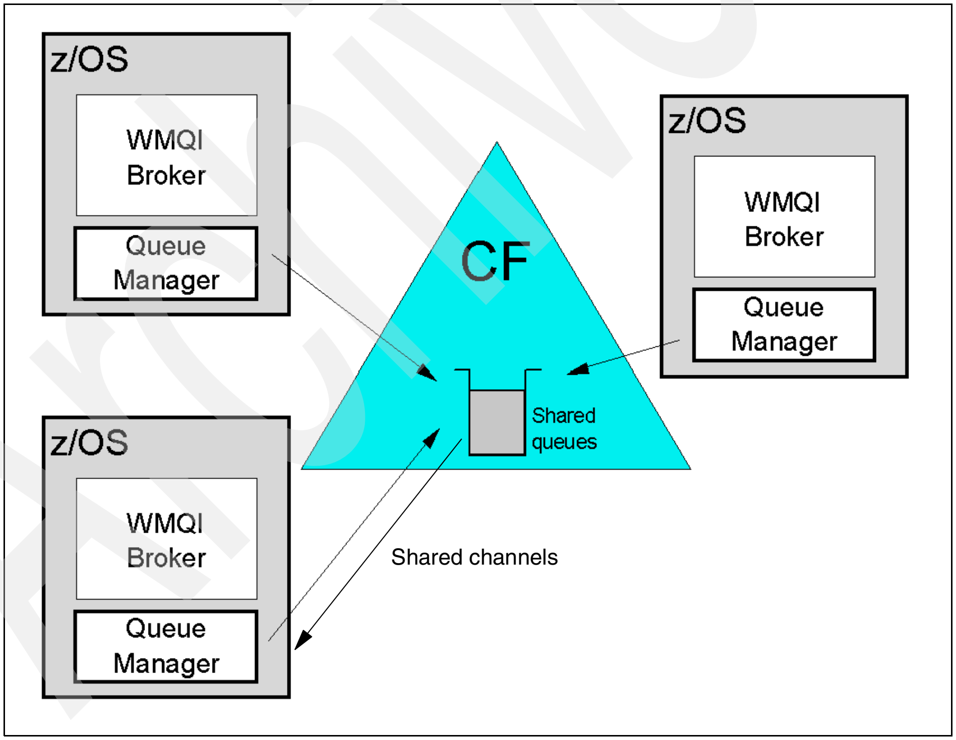


Figure 13-2 Multiple brokers in a Parallel Sysplex

Improved availability of WMQIB message flows can be achieved if they are deployed to several brokers. The queue managers these brokers are connected to share the queues involved in those message flows. That means that the data present in any of these shared queues is available to any broker. In case of a failure of one of the systems, unfinished units of work are backed out, so the backed-out messages are available for the rest of the brokers in the sysplex.

Refer to *WebSphere MQ Integrator for z/OS V2.1*, SG24-6528, for a discussion of possible configurations and considerations to be taken into account when you implement a high availability broker configuration in a Parallel Sysplex.

13.3 WebSphere MQ and IMS shared queue groups

The IMS also has a shared message queue capability. This feature was introduced in IMS Version 6. If enabled, the message queues are stored in dedicated coupling facility structures just like WebSphere MQ shared queues. If several IMS control regions are configured in a shared queues group, they can all access these queues and can benefit from the dynamic workload balancing that this environment can provide.

For a customer running a Parallel Sysplex environment, consisting of several cloned LPARS, it is quite likely that WebSphere MQ queue-sharing groups and IMS shared queue groups are coexisting. If WebSphere MQ is being used to deliver transactions to the IMS shared queues group, there will be an IMS bridge connecting each queue manager to its local IMS control region. This configuration is shown in Figure 13-3 on page 260.

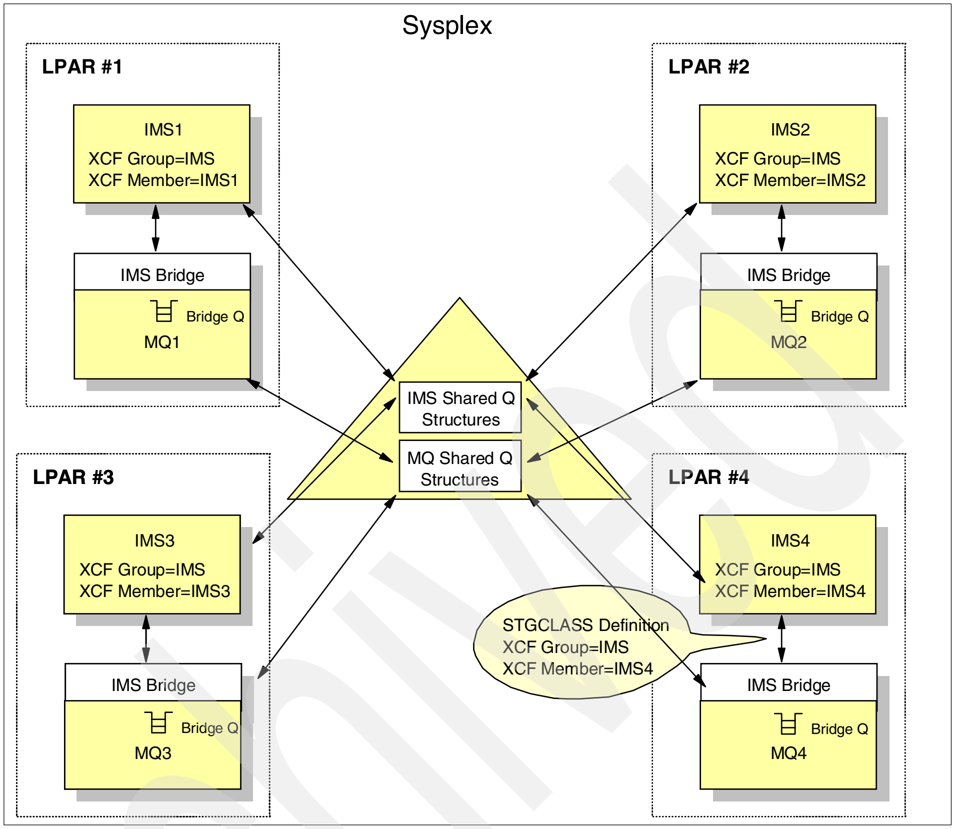


Figure 13-3 IMS and WebSphere MQ shared queues in a sysplex

The IMS Bridge uses the IMS transport protocol Open Transaction Manager Acess (OTMA) to connect the queue manager to IMS.

Underlying OTMA is the sysplex communication protocol XCF. Subsystems that need to communicate using XCF have to join a common XCF group and be identified by a unique XCF member name within that group.

A current restriction with the OTMA protocol is that a WebSphere MQ queue manager must define a connection to a specific IMS control region. This connection identifies an XCF group and Member name of an IMS control region.

The implications of this are apparent if one IMS control region needs to be removed from the queue-sharing group. Unless a coordinated change was made to stop delivering IMS messages through its associated WebSphere MQ queue manager, then messages would start building up on the IMS bridge queue. To avoid this situation, the associated WebSphere MQ queue manager must also be shut down.

An OTMA and WebSphere MQ design change request has been raised to allow the option for the IMS bridge to target a generic XCF member rather than a specific member of the group. XCF has a sysplex scope, so a WebSphere MQ queue manager can have an IMS bridge connection to another IMS region in the shared queues group on another LPAR. This change would allow an IMS system with a WebSphere MQ connection to be shut down and the queue manager could reconnect automatically to another member of the group. This scenario is shown in Figure 13-4 on page 262. WebSphere MQ queue manager MQ1 had an XCF connection to IMS region IMS1 running locally on LPAR #1. When IMS1 was shut down for maintenance, the IMS Bridge reconnects to another IMS image in the IMS shared queues group (IMS4) running on LPAR #4. The STGCLASS definition has specified a generic XCF member name of IMS*.

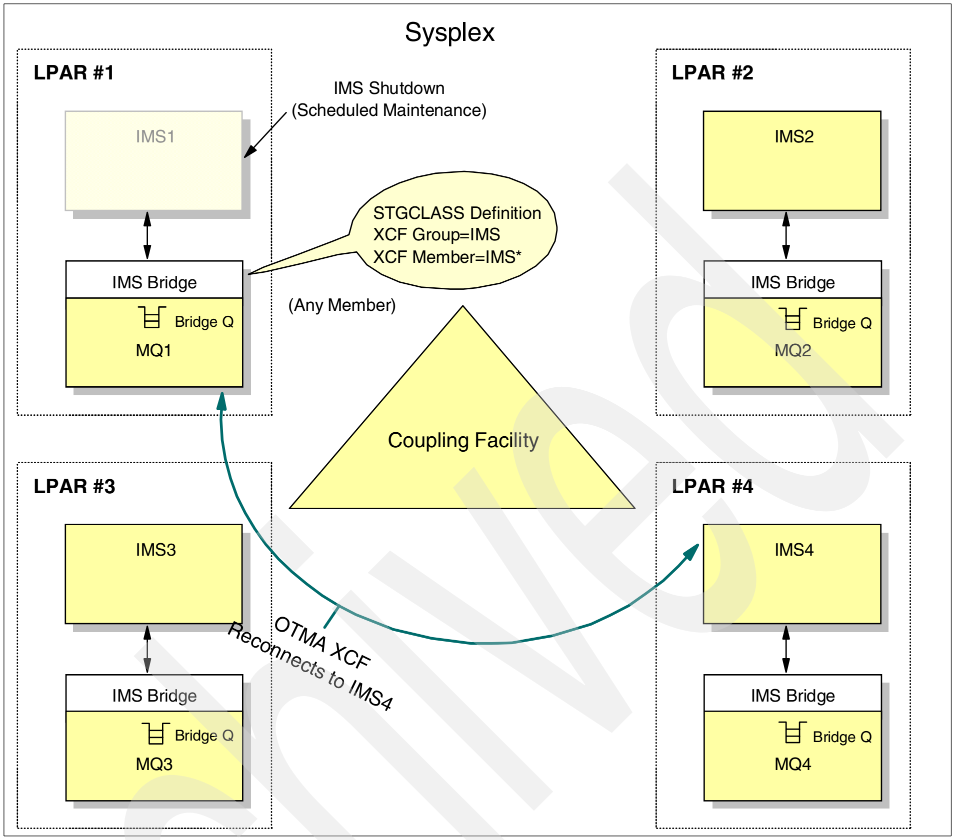


Figure 13-4 Proposed XCF connection using generic XCF member name

Until this design change is implemented, the IMS and WebSphere MQ queue manager shutdowns will need to be coordinated to prevent request queuing up in queue manager MQ1.

Part 4

Operations and recovery

In Part 4 of this redbook, we look at the operational impact of a queue-sharing environment and discuss some failure and recovery scenarios.

14

Operational impact of shared queue environment

There are new commands and changes to existing commands and administration screens implemented to support shared queues and the queue-sharing group.

The two main changes are the command scope parameter, which defines the routing of commands, and the object disposition parameter, which signifies where the *object definition* is kept, and how the object behaves.

You may be familiar with some of the changed screens and commands, since some of them were introduced with MQSeries for OS/390 V5.2. On the other hand, you may have missed some changes introduced since MQSeries for OS/390 V5.2, which we do not include in this chapter, because our main interest is to show impacts on a shared queue environment, and not to provide a complete description of operational changes.

14.1 Changes to the operations and control screens

Because of the introduction of the queue-sharing group, some of the WebSphere MQ screens are different. For example, the initial connect to the queue manager is different from previous versions. You can also now use the queue-sharing group name to request information or to make updates.

14.1.1 WebSphere MQ main operations and control menu

When you initially view the main WebSphere MQ operations and control screen, shown in Figure 14-1, you are not connected to any queue manager.

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

Complete fields. Then press Enter.

Action  . . . . . . . . . .       1. Display       4. Manage    6. Start
                                  2. Define like   5. Perform   7. Stop
                                  3. Alter

Object type . . . . . . . .               +
Name  . . . . . . . . . . .
Disposition . . . . . . . .    Q=Qmgr,C=Copy,P=Private,
                               G=Group,S=Shared,A=All

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

Figure 14-1 New WebSphere MQ main operations and control menu

You can connect to a queue manager and define the queue managers on which your requests should be executed by filling in the following three fields:

- Connect queue manager: This is the queue manager to which you actually connect.
- Target queue manager: With this parameter you specify on which queue manager you want to input your request.
- Action queue manager: On the action queue manager, the commands are actually executed.

You can choose a queue manager and enter the name of it into the connect queue manager name field. If you fill out only the connect name field and leave the fields for target queue manager and action queue manager blank, target and action queue managers default to the same name as you set in the connect name field, and a pop-up screen is presented to allow you to confirm this default.

If you leave the connect name field blank and press the Enter key, you will be connected to the default queue manager for batch applications, which is defined in CSQBDEFV adapter module. Chapter 2, “Customizing your queue managers” in *WebSphere MQ for z/OS System Setup Guide,* SC34-6052, contains a description of how to set up CSQBDEFV.

Note: If you do not have a customized version of CSQBDEFV, the shipped version is used, which has queue manager CSQ1 as default queue manager. If your queue manager has a different name, you receive the message:

```
CSQO010I Queue manager or group is not available
```

Other changes to the WebSphere MQ operations and control main menu include:

- Disposition Field

  This field relates to the QSGDISP parameter, which is introduced in Chapter 8, “Shared queue overview” on page 137. In addition to the four types of WebSphere MQ objects, the administration allows you to specify:

  – P for Private, which means that the object definitions have a disposition of either QMGR or COPY. So in either case a local definition exists. No group definition is displayed.

  – A for All, which means that all objects of the object type you selected are included.

  If the object type is MANAGER, SYSTEM, or SECURITY, the value in the disposition input field is ignored. Also for the two new parameters, CFSTRUCT and CFSTATUS, the disposition field is of no relevance. For all other objects you have to use a valid combination. For example, the use of disposition S is valid only with queues (QUEUE, QLOCAL, CLUSQ and QSTATUS).

- Option 2 is now *Define like* instead of *Define*

  This change does not only apply to the main menu, but WebSphere MQ now uses *Define like* in all administration screens instead of *Define*.

  For option 2 *Define like*, the input fields Name and Disposition belong to the object, from which you want to copy your definitions.

Therefore, if you specify an existing object name, but with a disposition different from the disposition defined for the existing object, you receive message CSQO051I, and the name you entered is assumed to be the name of the new object definition. A subsequent definition fails, if an object of this name already exists. Example 14-1 shows the error messages we receive when we try to define a queue CICS01.INITQ.

Example 14-1 Possible error scenario after CSQO051I message

```
CSQM095I -MQV1 CSQMAQLC QLOCAL(CICS01.INITQ) ALREADY EXISTS
CSQM090E -MQV1 CSQMAQLC FAILURE REASON CODE X'00D44003'
CSQ9023E -MQV1 CSQMAQLC ' DEFINE QLOCAL' ABNORMAL COMPLETION
```

- Option 4 (*Manage*) is new on the main menu.

  This option is also available on other menus, and is discussed later in this chapter.

14.1.2 New fields for disposition in list screens

WebSphere MQ objects for which you can specify the list menus have also changed. Now they provide information on the disposition. This change is shown in the List Channels screen in Figure 14-2.

```
                            List Channels - MQVG                    Row 1 of 8

Type action codes. Then press Enter.
 1=Display   2=Define like   3=Alter   4=Manage   5=Perform
 6=Start     7=Stop

    Name                    Type          Disposition    Status
<>  MQVG.M23WPF*            CHANNEL       ALL     MQVG
    MQVG.M23WPF22           SENDER        COPY    MQV1   INACTIVE
    MQVG.M23WPF22           SENDER        COPY    MQV2   INACTIVE
    MQVG.M23WPF22           SENDER        COPY    MQV4   INACTIVE
    MQVG.M23WPF22           SENDER        GROUP
    MQVG.M23WPF31           SENDER        COPY    MQV1   INACTIVE
    MQVG.M23WPF31           SENDER        COPY    MQV2   INACTIVE
    MQVG.M23WPF31           SENDER        COPY    MQV4   RUN
    MQVG.M23WPF31           SENDER        GROUP
                       ******** End of list ********
```

Figure 14-2 List Channels screen

In this screen, you can see a line marked with <>, which is called the *selector* field. This line shows you the selection criteria you entered or implied on the main menu.

For the disposition field, the actual value is shown, as a result of the value you entered for disposition on the main menu and your selection of the target queue manager. For example, if you leave the disposition field blank and your action queue manager is the same as your target queue manager, the disposition on this list screen displayed is ALL.

The target queue manager is shown in the screen title.

Since in our display the target queue manager is the queue-sharing group, you can see for all channels defined as group channels, additional lines for the local copies of all queue managers active in this queue-sharing group.

14.1.3 New list queues screens

The List Queues screens have changed. WebSphere MQ now offers three screens:

- CSQOLQAA, which shows the queue names, the type of the queue, and the disposition. An example of this screen is shown in Figure 14-3, which shows all our application queues for the ECHO application.

```
CSQOLQAA                      List Queues - MQVG                        Row 1 of 8

Type action codes. Then press Enter.
 1=Display   2=Define like   3=Alter   4=Manage


    Name                                                  Type      Disposition
<>  CICS.MQVG.ECHO*                                       QUEUE     ALL     MQVG
    CICS.MQVG.ECHO01                                      QLOCAL    SHARED
    CICS.MQVG.ECHO01.LONG                                 QLOCAL    QMGR    MQV1
    CICS.MQVG.ECHO01.LONG.REQ                             QALIAS    QMGR    MQV2
    CICS.MQVG.ECHO01.LONG.REQ                             QALIAS    QMGR    MQV4
    CICS.MQVG.ECHO01.SHAREDQ.LONG.REPLY                   QLOCAL    QMGR    MQV2
    CICS.MQVG.ECHO01.SHAREDQ.SHORT.REPLY                  QLOCAL    QMGR    MQV2
    CICS.MQVG.ECHO01.SHORT.REQ                            QALIAS    QMGR    MQV2
    CICS.MQVG.ECHO01.SHORT.REQ                            QALIAS    QMGR    MQV4
                     ******** End of list ********
```

Figure 14-3 List Queues screen

Pressing the function PF11displays the status of the queues.

- CSQOL1AA shows the status of the queues. In the first line of each queue display, you can see the queue name, disposition, and depth. The second line shows the output use, input use, and whether uncommitted messages exist. In our sample shown in Figure 14-4, we do not have any use counts, since the queue manager was restarted.

```
CSQOL1AA            List Local Queues - Status - MQVG             Row 1 of 6

Use '/' to select one or more open queues, then press Enter to display
application details, or press F11 to display all open queues.

    Queue name                                         Disposition   Depth
      Output use  Input use   Uncommitted messages
<>  CICS.MQVG.ECHO*                                    ALL     MQVG
    CICS.MQVG.ECHO01                                   SHARED  MQV1  2
    CICS.MQVG.ECHO01                                   SHARED  MQV2  2
    CICS.MQVG.ECHO01                                   SHARED  MQV4  2
    CICS.MQVG.ECHO01.LONG                              QMGR    MQV1  0
    CICS.MQVG.ECHO01.SHAREDQ.LONG.REPLY                QMGR    MQV2  0
    CICS.MQVG.ECHO01.SHAREDQ.SHORT.REPLY               QMGR    MQV2  0
                     ******** End of list ********
```

Figure 14-4 List Local Queues - Status screen

- CSQOLOAA shows all open queues, their disposition, and the access. A second line for the queues shows information on applications that are currently accessing the queue. Figure 14-5 shows an example of open queues.

```
                          List Open Queues - MQVG                 Row 1 of 5



    Queue name                                         Disposition   Access
      Application       ASID  Application information          User ID
<>  CICS.MQVG.ECHO*                                    ALL     MQVG
    CICS.MQVG.ECHO01                                   SHARED  MQV4  0 -  - -
      MQV4CHIN CHINIT    0043  M23WPH767.TO.MQVG               STC
                                9.24.104.132
    CICS.MQVG.ECHO01                                   SHARED  MQV3  - IS - -
      SCSCLSA5 CICS      0097  ECHO 0036762                    CICSUSER
                     ******** End of list ********
```

Figure 14-5 LIst Open Queues screen

14.1.4 Manage object displays

WebSphere MQ provides the function MANAGE for objects, with different possible actions. It incorporates the Delete action available in earlier releases. Let's take a closer look at the screens for managing queues.

Manage queues

If you enter the value 4 for MANAGE at the List Queues screen and press Enter, the screen in Figure 14-6 will be displayed.

```
CSQO41AA                     Manage a Local Queue

Select function type, then press Enter.


Function type . . . . . . . .       1. Delete
                                    2. Create a local copy on MQVG
                                    3. Refresh local copy
                                    4. Clear messages
                                    5. Move messages to another queue
                                    6. Add messages to another queue

Queue name  . . . . . . . . : CICS.MQVG.ECHO01
Queue type  . . . . . . . . : QLOCAL
Disposition . . . . . . . . : SHARED
Description . . . . . . . . : ECHO 01 APPLICATION INPUT QUEUE


To queue name (Move/Add)  . .
```

Figure 14-6 Manage queues screen

You have now several options:

- Selecting **Option 1. Delete** allows you to delete an object.
- Selecting **Option 2. Create** allows you to create a local copy of an object defined in a shared repository.
- Selecting **Option 3. Refresh** allows you to refresh the local copy with information from the base object. After this action the local copy has the same defintion as the base object.
- You now can empty a queue by selecting **Option 4. Clear messages**.
- Selecting **Option 5**, you are able to move messages to another queue. This can be helpful if you want to change attributes of the queue definition without losing data.

- Selecting **Option 6** allows you to add messages from the selected queue to another queue.

For option 5 and 6, you must also specify the name of the queue where you want to put your messages.

14.1.5 New functions for channels

In addition to the changes for managing channels, WebSphere MQ provides now two possibilities to stop a channel. Figure 14-7 shows the screen provided if you select to stop a channel.

```
CSQOHCAA                        Stop a Channel

Complete fields, then press Enter to stop channel.


Channel name  . . . . . . . : MQVG.M23WPF31
Channel type  . . . . . . . : SENDER
Description . . . . . . . . : Shared sender to M23WPF31


Disposition . . . . . . . . . P     P=Private on MQV4
                                    A=Shared on any queue manager

Stop mode . . . . . . . . . . 1     1. Quiesce   2. Force
Stop status . . . . . . . . . 1     1. Stopped   2. Inactive

Queue manager . . . . . . . .
Connection name . . . . . . .
```

Figure 14-7 Stop a Channel screen

In addition to the stop mode, which can be quiesce or force, you can now enter the required stop status. For stop status INACTIVE, this screen provides two fields in the last two lines for queue manager name or connection name. The two possibilities for stop status are:

- STOPPED. The resulting channel status is STOPPED, and you have to restart the channel manually. If you do not specify a queue manager or connection name, this is the default. If either the queue manager or the connection name is specified, you are not allowed to use this mode and you receive an error message. Example 14-2 on page 273 shows the error messages you may receive when using mode STOPPED and specifying the queue manager name.

Example 14-2 Channel stop error, when specifying mode STOPPED and a queue manager name

```
CSQM155I -MQV4 CSQMTCHL STATUS(STOPPED) NOT ALLOWED WITH QMNAME OR CONNAME
CSQM090E -MQV4 CSQMTCHL FAILURE REASON CODE X'00D44004'
CSQ9023E -MQV4 CSQMTCHL ' STOP CHANNEL' ABNORMAL COMPLETION
```

- INACTIVE. The resulting channel status is INACTIVE, which means the channel is for automatic restart (that is, for triggering). If you specify a queue manager name, this option is the default.

To see a detailed description of the channel status, press PF11.

14.2 Coupling facility list structures

WebSphere MQ uses coupling facility list structure to store messages and administration information, which have to be shared among queue managers in a queue-sharing group. One administrative structure is defined for the whole queue-sharing group, whereas application structures are defined according to the application requirements.

To support administration and information from the WebSphere MQ point of view for these structures, WebSphere MQ introduces a set of commands and some new administration screens to manage the new object type CFSTRUCT, which covers all application structures. This object can now be referenced in the queue definitions to associate a shared queue with a structure name.

The commands you can now use to manage your application structures include the following:

- ALTER CFSTRUCT: With this command you can alter your application structure backup and recovery parameters. You have to use this command if you want to upgrade to a higher WebSphere MQ CFLEVEL, as described in Chapter 10, “Migration to WebSphere MQ for z/OS Version 5.3” on page 193.
- DEFINE CFSTRUCT: Use this command to define the backup and recovery parameters for a coupling facility application structure.
- DELETE CFSTRUCT: With this command you delete a coupling facility application structure definition of WebSphere MQ. To delete the structure in your CFRM policy, you can use the SETXCF FORCE,STR,STRNAME=*strname* command, and you have to update your policy to reflect this change.
- DISPLAY CFSTRUCT: This command displays the WebSphere MQ information for your CFSTRUCT definition. To display the coupling facility structure information itself, use the command D XCF,STR,STRNM=*strname*.

- Commands:BACKUP CFSTRUCTBACKUP CFSTRUCT: Coupling facility list structures can be backed up using a WebSphere MQ backup command BACKUP CFSTRUCT. This puts a copy of the persistent messages currently within the CF structure onto the active log data set of the queue manager making the backup, and writes a record of the backup to DB2.
- RECOVER CFSTRUCT: In the event of a coupling facility failure, you can use a manual WebSphere MQ command RECOVER CFSTRUCT(*name1, name2,....nameN*) to the queue manager that will do the recovery. This queue manager locates the relevant fuzzy backups on the queue manager's logs using the data in DB2 and the bootstrap data sets. It replays these backups, and then positions each queue manager's recovery log stream to a time just before the earliest timestamp of any of the restored backups. It then uses timestamps in each shared queue's related log record to replay the shared queue operations in the correct time sequence across the queue-sharing group.
- DISPLAY CFSTATUS displays the current status of all structures (including the administrative structure). You can display three different types of status information:
  - SUMMARY: Gives an overview of the status information
  - CONNECT: Shows all members connected to the structure and in the case of a connection failure, failure information
  - BACKUP: Shows backup date and time, RBA information and the queue manager that did the backup

  Figure 14-8 on page 275 shows the summary display and the backup display.

```
-MQV1 DISPLAY CFSTATUS(A*) TYPE(SUMMARY)
CSQM293I -MQV1 CSQMDMSG 1 CFSTATUS FOUND MATCHING REQUEST CRITERIA
CSQM201I -MQV1 CSQMDMSG  DISPLAY CFSTATUS DETAILS 841
CFSTATUS(APPLICATION1)
TYPE(SUMMARY)
CFTYPE(APPL)
STATUS(ACTIVE)
SIZEMAX(20480)
SIZEUSED(14)
ENTSMAX(8351)
ENTSUSED(1207)
FAILTIME( )
FAILDATE( )
CSQMDMSG END CFSTATUS DETAILS
CSQ9022I -MQV1 CSQMDMSG ' DISPLAY CFSTATUS' NORMAL COMPLETION
-MQV1 DISPLAY CFSTATUS(A*) TYPE(BACKUP)
CSQM293I -MQV1 CSQMDMSG 1 CFSTATUS FOUND MATCHING REQUEST CRITERIA
CSQM201I -MQV1 CSQMDMSG  DISPLAY CFSTATUS DETAILS 852
CFSTATUS(APPLICATION1)
TYPE(BACKUP)
STATUS(ACTIVE)
QMNAME(MQV2)
BKUPTIME(11.34.04)
BKUPDATE(2002-08-26)
BKUPSIZE(0)
BKUPSRBA(000028A95E23)
BKUPERBA(000028A96218)
FAILTIME( )
FAILDATE( )
CSQMDMSG END CFSTATUS DETAILS
CSQ9022I -MQV1 CSQMDMSG ' DISPLAY CFSTATUS' NORMAL COMPLETION
```

Figure 14-8 Display CFSTATUS output

For both the CFSTRUCT objects and the CFSTATUS operation, screens are provided. Figure 14-9 on page 276 shows the List CF Structures screen. With option 1 you can display the summary information of the structure and from this screen you can request the status by pressing PF11.

```
                           List CF Structures - MQVG                Row 1 of 1

Type action codes. Then press Enter.
 1=Display   2=Define like   3=Alter   4=Manage


    Name                   Structure level
<>  *
    APPLICATION1           3
                     ******** End of list ********
```

Figure 14-9 List CF Structures screen

To display the status of all structures, you can enter `CFSTATUS` in the object field of the main menu and an asterisk (*) in the name field. Figure 14-10 shows the status display screen.

```
                   List CF Structures - Current Status - MQVG        Row 1 of 4



    CF structure name  Status     Failure time         Type
                     Size(KB)   %Used  Entries   In use
<>  *
    APPLICATION1       ACTIVE                          APPL
                     20480      7      8351      586
    CSQ_ADMIN          ACTIVE                          ADMIN
                     10240      5      10411     585
                     ******** End of list ********
```

Figure 14-10 CFSTRUCT status display screen

Pressing PF11 allows you to display the connection status information of your structures. In our environment we have structure APPLICATION1, to which all four queue managers of the queue-sharing group MQVG are connected. Figure 14-11 on page 277 shows the connection status of our application structure APPLICATION1.

```
            List CF Structures - Connection Status - MQVG       Row 1 of 8



    CF structure name  Status     Failure time         System
      Queue manager
<>  *
    APPLICATION1       ACTIVE                          SC53
      MQV1
    APPLICATION1       ACTIVE                          SC61
      MQV2
    APPLICATION1       ACTIVE                          SC66
      MQV3
    APPLICATION1       ACTIVE                          SC62
      MQV4
                  ******** End of list ********
```

Figure 14-11   Connection status display of application structure

14.3 Parameters in MQSC commands

There are csome new keywords that can be used with MQSC commands.

14.3.1 CMDSCOPE

One of the main changes to commands is the new keyword CMDSCOPE. This enables you to enter commands on one queue manager of a queue-sharing group and have that command executed on any or all of the queue managers within that group. The syntax of the command is:

- CMDSCOPE (qmgr)

  Activates the command on the queue manager that the command was entered.

- CMDSCOPE(target)

  Activates the command on the target queue manager, assuming the target queue manager is within the same queue-sharing group as the queue manager issuing the command.

- CMDSCOPE(*)

  Activates the command on all active queue managers in the queue-sharing group.

The default is for the command to be executed on the local queue manager. The CMDSCOPE option has been added to all MQSC commands except the following:

- DEFINE BUFFPOOL
- DEFINE PSID
- DISPLAY CMDSERV
- DISPLAY GROUP
- START CMDSERV
- START QMGR
- STOP CMDSERV

Should you try to add CMDSCOPE to one of these commands, you will see the error message shown in Example 14-3.

Example 14-3 Using CMDSCOPE erroniously

```
-MQV4 DISPLAY GROUP,CMDSCOPE(*)
CSQ9001E -MQV4 KEYWORD CMDSCOPE IS INVALID
CSQ9023E -MQV4 CSQ9SCND 'DISPLAY GROUP' ABNORMAL COMPLETION
```

When you enter a command and use CMDSCOPE, the results are displayed only on the queue manager that submitted the command. The responses are grouped within the queue manager they originated from. If the queue manager that issued the command is MQV4, and the queue-sharing group has additional members MQV1, MQV2 and MQV3 and you are going to display for instance all local queues starting with SYSTEM.ADMIN.CHANNEL, the resulting output is shown in Figure 14-12 on page 279.

```
-MQV4 DIS QUEUE(SYSTEM.ADMIN.CHANNEL*) CMDSCOPE(*)
CSQN137I -MQV4 'DIS QUEUE' command accepted for CMDSCOPE(*), sent to 4
CSQN121I -MQV4 'DIS QUEUE' command responses from MQV3
CSQM293I -MQV3 CSQMDMSG 1 QUEUE FOUND MATCHING REQUEST CRITERIA
CSQM201I -MQV3 CSQMDMSG  DIS QUEUE DETAILS 208
QUEUE(SYSTEM.ADMIN.CHANNEL.EVENT)
TYPE(QLOCAL)
QSGDISP(QMGR)
CSQMDMSG END QUEUE DETAILS
CSQ9022I -MQV3 CSQMDMSG ' DIS QUEUE' NORMAL COMPLETION
CSQN121I -MQV4 'DIS QUEUE' command responses from MQV2
CSQM293I -MQV2 CSQMDMSG 1 QUEUE FOUND MATCHING REQUEST CRITERIA
CSQM201I -MQV2 CSQMDMSG  DIS QUEUE DETAILS 212
QUEUE(SYSTEM.ADMIN.CHANNEL.EVENT)
TYPE(QLOCAL)
QSGDISP(QMGR)
CSQMDMSG END QUEUE DETAILS
CSQ9022I -MQV2 CSQMDMSG ' DIS QUEUE' NORMAL COMPLETION
CSQN121I -MQV4 'DIS QUEUE' command responses from MQV1
CSQM293I -MQV1 CSQMDMSG 1 QUEUE FOUND MATCHING REQUEST CRITERIA
CSQM201I -MQV1 CSQMDMSG  DIS QUEUE DETAILS 216
QUEUE(SYSTEM.ADMIN.CHANNEL.EVENT)
TYPE(QLOCAL)
QSGDISP(QMGR)
CSQMDMSG END QUEUE DETAILS
CSQ9022I -MQV1 CSQMDMSG ' DIS QUEUE' NORMAL COMPLETION
CSQN121I -MQV4 'DIS QUEUE' command responses from MQV4
CSQM293I -MQV4 CSQMDMSG 1 QUEUE FOUND MATCHING REQUEST CRITERIA
CSQM201I -MQV4 CSQMDMSG  DIS QUEUE DETAILS 220
QUEUE(SYSTEM.ADMIN.CHANNEL.EVENT)
TYPE(QLOCAL)
QSGDISP(QMGR)
CSQMDMSG END QUEUE DETAILS
CSQ9022I -MQV4 CSQMDMSG ' DIS QUEUE' NORMAL COMPLETION
CSQN122I -MQV4 'DIS QUEUE' command for CMDSCOPE(*) normal completion
```

Figure 14-12 Sample of using DISPLAY command with CMDSCOPE(*)

You can see in the message flow that first the command is sent to all queue managers. Then the result from each queue manager is displayed. Each queue manager result starts with message CSQN121I. Then the actual display of the queue follows, ending with message CSQ9022I. After the output of the last queue manager, message CSQN122I follows to show the end of the complete display.

14.3.2 QSGDISP

Another main change to commands is the keyword QSGDISP. It specifies the disposition of the object to which you are applying the command (that is, where it is defined and how it behaves). You can choose whether you want to share the object definition with other queue managers (a global definition), or whether the object definition is to be used by one queue manager only (a private definition). This is called the *object disposition*.

- Global definition

  If your queue manager belongs to a queue-sharing group, you can choose to share any object definitions you make with the other members of the group. This means that an object has to be defined once only, reducing the total number of definitions required for the whole system. Global object definitions are held in a shared repository (a DB2 shared database), and are available to all the queue managers in the queue-sharing group. These objects have a disposition of GROUP or SHARED.

- Private definition

  If you want to create an object definition that is required by one queue manager only, or if your queue manager is not a member of a queue-sharing group, you can create object definitions that are not shared with other members of a queue-sharing group. Private object definitions are held on page set zero of the defining queue manager. These objects have a disposition of QMGR or COPY.

The QSGDISP parameter has been added to the ALTER, DELETE, DEFINE, and DISPLAY commands for:

- Channels
- Namelists
- Processes
- Queues
- Storage classes
- Authinfo

There are seven different QSG dispositions:

1. QSGDISP(QMGR)

   If you define an object with disposition QMGR, then the definition will be held on page set zero of the defining queue manager. Messages sent to that queue are available to the queue manager that defined the queue only. This is a private definition and can be used for all of the above-mentioned objects. You can use this disposition with the ALTER, DELETE, DEFINE, and DISPLAY commands.

2. QSGDISP(PRIVATE)

   This disposition can only be used with the ALTER or DISPLAY command for the objects mentioned above. The objects reside on the page set of the queue manager that executes the command, and were defined with QSGDISP(QMGR) or QSGDISP(COPY). Any object residing in the shared repository is unaffected. QSGDISP(PRIVATE) is not permitted for DELETE or DEFINE commends.

3. QSGDISP(COPY)

   If an object is defined with disposition COPY, then it is defined on the page set of the queue manager that executes the command using the QSGDISP(GROUP) object of the same name as the LIKE object. This can be used with the ALTER, DELETE, DEFINE and DISPLAY commands for queues, channels, authinfo, namelists, processes, and storage classes.

4. QSGDISP(GROUP)

   If you define an object with disposition GROUP, then the object definition is held in the shared repository and a local copy of the definition is held on page set zero of each queue manager in the group. This can be used with the ALTER, DELETE, DEFINE and DISPLAY commands for queues, channels, authinfo, namelists, processes, and storage classes.

5. QSGDISP(SHARED)

   The definition of an object defined with disposition SHARED is held in the shared queue repository and is available to all queue managers in the queue-sharing group because it is stored in the coupling facility. This definition can only be used with the ALTER, DELETE, DEFINE, and DISPLAY commands for queues.

6. QSGDISP(ALL)

   For the DISPLAY command, you can also specify a QSGDISP(ALL). It displays information for objects defined with QSGDISP(QMGR) or QSGDISP(COPY). If there is a shared queue manager environment, and the command is being executed on the queue manager where it was issued, this option also displays information for objects defined with QSGDISP(GROUP). If QSGDISP(ALL) is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

7. QSGDISP(LIVE)

   Like the QSGDISP(ALL), this parameter can only be used with the DISPLAY command. It is the default value and displays information for objects defined with QSGDISP(QMGR), QSGDISP(COPY), or QSGDISP(SHARED). And as with QSGDISP(ALL), if it is specified in a shared queue manager environment, the command might give duplicated names (with different dispositions).

14.3.3 CFSTRUCT

This parameter is used only in conjunction with managing queues. CFSTRUCT associates a shared queue with the application structure where you want messages for this queue to be stored.

Only local queues and model queues can be defined with the parameter CFSTRUCT.

If you define a new local queue with QSGDISP(SHARED) and you use the queue-sharing group name for your CMDSCOPE value, you receive the following error message:

```
CSQN132E -MQV1 CMDSCOPE(MQVG) not allowed with disposition SHARED
```

The *system action* for this message states that the command is not processed, but we found that after issuing this command via the screens, a shared local queue definition exists.

> **Note:** If you define a shared local queue with a structure name that is not defined to this queue-sharing group, a subsequent open to this queue fails with reason code `2349 (MQRC_CF_STRUC_ERROR)`.

There are some important rules when you use the commands ALTER QLOCAL, ALTER QMODEL, DEFINE QLOCAL with REPLACE, and DEFINE QMODEL with REPLACE:

- CFSTRUCT cannot change on a local queue with QSGDISP(SHARED). To change this attribute of the queue, you have to delete and redefine the queue with the new CFSTRUCT value.
- On a model queue with DEFTYPE(SHAREDYN), CFSTRUCT cannot be blank. The resulting messages are shown in Example 14-4.

Example 14-4 Error receiving for model queue definition with DEFTYPE(SHAREDYN) and no CFSTRUCT value

```
CSQM145I -MQV1 CSQMAQLC CFSTRUCT VALUE REQUIRED FOR SHARED QUEUE
CSQM090E -MQV1 CSQMAQLC FAILURE REASON CODE X'00D44004'
CSQ9023E -MQV1 CSQMAQLC ' DEFINE QMODEL' ABNORMAL COMPLETION
```

- On a local queue with a QSGDISP other than SHARED, or a model queue with a DEFTYPE other than SHAREDYN, the value of CFSTRUCT does not matter. The structure name is stored in the object definition, but the object is not defined as shared object (no DB2 table entry for this queue is generated).

- For DEFINE QLOCAL with NOREPLACE and DEFINE QMODEL with NOREPLACE, the following rules apply:
  - On a local queue with QSGDISP(SHARED) or a model queue with a DEFTYPE(SHAREDYN), CFSTRUCT cannot be blank.
  - On a local queue with a QSGDISP other than SHARED, or a model queue with a DEFTYPE other than SHAREDYN, the value of CFSTRUCT does not matter.

14.4 Additional new commands

In this section we introduce some new commands that are useful for managing your queue managers and help you to keep planned outage times to a minimum.

14.4.1 Refresh the early code

With WebSphere MQ V5.3, you do not need to IPL your system for a change of the early code, as was necessary with earlier releases. To enable this functionality, which is only provided with WebSphere MQ V5.3, you need at least one IPL to activate the WebSphere MQ V5.3 early code for the first time before migrating from earlier releases.

Once you have the early code of WebSphere MQ V5.3 activated, you can replace your early code dynamically. Assuming that maintenance has to be applied to your WebSphere MQ V5.3 code, and you want to use different data sets for the maintenance, follow these steps:

1. Install the maintenance to your new data sets.
2. Stop your queue manager(s) using the WebSphere MQ command STOP QMGR.
3. Use the commands SETPROG LPA, SETPROG APF and SETPROG LNKLST to add the new LPA modules, add your new data sets to the APF authorization list, and update your linklist concatenation.
4. Refresh the early code, using the WebSphere MQ command REFRESH QMGR TYPE(EARLY).

> **Note:** Refreshing the early code works only if your early code resides in the LPA. Therefore, if you want to use this function, you have to add your modules to the LPA.

If the early code modules do not reside in the LPA, the command to refresh the early code fails. Figure 14-5 shows the error messages and the following dump.

Example 14-5 Early code refresh error

```
-MQGV REFRESH QMGR TYPE(EARLY)
CSQ3105E -MQGV    CSQ3UR00 - UNABLE TO LOAD EARLY PROCESSING PROGRAM
CSQ3EPX. MQGV IS NOT AVAILABLE
IEA794I SVC DUMP HAS CAPTURED: 620
DUMPID=001 REQUESTED BY JOB (MQRES1  )
DUMP TITLE=MQGV,ABN=5C6-00F30105,U=        ,C=     .530.SSSC-CS
           Q3INI ,M=CSQ3UR00,PSW=077C1000904F7906,ASID=0041
CSQ3104I -MQGV    CSQ3ECOX - TERMINATION COMPLETE
CSQ3100I -MQGV    CSQ3ECOX - SUBSYSTEM MQGV READY FOR START COMMAND
```

5. Change the WebSphere MQ data sets in your queue manager procedure and the channel initiator procedure to the data sets containing the service.
6. Restart the queue manager, using the WebSphere MQ command START QMGR.

You can check the status of your early code by checking the queue manager startup log and looking for the following lines:

```
CSQ3111I -MQV1 CSQYSCMD - EARLY PROCESSING PROGRAM IS V5.3 LEVEL
001-000
```

14.4.2 Dynamic change of system parameter values

WebSphere MQ now provides three commands to change dynamically certain parameter values that you defined via the system parameter module CSQZPARM macros CSQ6ARVP, CSQ6LOGP and CSQ6SYSP. You will find a complete list of parameters you can change dynamically in *WebSphere MQ Script (MQSC) Command Reference,* SC34-6055.

Note: All dynamic changes are valid only during the time the queue manager remains active. At a subsequent restart, the queue manager loads the values from CSQZPARM.

An additional command for each type of system parameter (ARCHIVE, LOG, SYSTEM) to display the actual setting is also provided. These DISPLAY commands allow you only to specify the CMDSCOPE parameter.

SET ARCHIVE

This command is used to change certain archive parameters specified in macro CSQ6ARVP.

The changes, you requested with the command SET ARCHIVE take effect at the next archive process.

SET LOG

With the SET LOG command, you can change certain log parameters specified in macro CSQ6LOGP.

Any changes for the parameter WRTHRSH take effect immediately. The changes for the parameter MAXARCH take effect at the next scheduled offload task.

SET SYSTEM

Use this command to change parameters specified in macro CSQ6SYSP.

Except for STATIME and TRACTBL, all parameter changes take effect immediately. The change for parameter STATIME takes effect when the current statistic gathering interval has expired, or when the new interval is shorter than the remaining time of the old interval. The values for parameter TRACTBL are picked up for any new START TRACE command. Old traces continue using the existing trace tables.

If you enlarge the value of parameter CTHREAD, the change takes effect immediately. If you reduce the value, and currently more threads are active than the new maximum thread limit, the reduction takes effect when current active threads stop.

To show the effect of the thread reduction, we first made a display of active threads, and found that only the channel initiator of our queue manager had active 20 threads. We then reduced the thread limits to values, even smaller than the number of current active threads, and the threads remained active. Example 14-6 shows the beginning of the messages after issuing the SET SYSTEM command.

Example 14-6 SET SYSTEM output for changed parameters

```
-MQV1 SET SYSTEM CTHREAD(10) IDBACK(9) IDFORE(1)
CSQJ337I -MQV1 SYSTEM parameters set
CSQJ322I -MQV1 DISPLAY SYSTEM report ...
Parameter   Initial value          SET value
----------- ---------------------- ----------------------
CTHREAD     300                    10
IDBACK      40                     9
IDFORE      100                    1
```

To test the new thread limits, we started our CICS trigger monitor transaction. The connection is deferred, because there are no more free threads available. The queue manager issues the following message:

```
CSQV428I -MQV1 CURRENT THREAD LIMIT OF 10 EXCEEDED.
CREATE THREAD FOR JOB CICSLSA1 DEFERRED
```

The transaction CKTI is delayed until free threads are available again.

After stopping and restarting the queue manager, the thread values again show only the initial values.

14.5 Utilities

There are new utilities and enhancements to existing utilities.

14.5.1 CSQ5PQSG

This utility allows you to manage your queue-sharing group and your queue manager definitions in your DB2 tables. CSQ5PQSG has the following parameters:

- ADD QSG: This parameter adds a queue-sharing group. An entry in the CSQ.ADMIN_B_QSG is created for the queue-sharing group. The function-specific parameters you have to specify are the queue-sharing group name, the DB2 data-sharing group name, and the DB2 subsystem identifier.
- REMOVE QSG: This parameter removes a queue-sharing group and deletes the entry in the table CSQ.ADMIN_B_QSG. Again you have to specify the queue-sharing group name, the data-sharing group name, and the DB2 subsystem identifier.
- MIGRATE DSG: This parameter is used only for migrating the DB2 tables from MQSeries for OS/390 V5.2 to WebSphere MQ for z/OS V5.3. Function-specific parameters are the data-sharing group name and the DB2 subsystem identifier. Do not use this parameter except when you migrate DB2 definitions with CSQ45ATB job.
- ADD QMGR: With this parameter you can add a queue manager to an existing queue-sharing group. An entry in the DB2 table CSQ.ADMIN_B_QMGR is created. For this parameter you have to specify the queue manager name, the queue-sharing group name, the data-sharing group name and the DB2 subsystem identifier.
- REMOVE QMGR: Specifying this parameter, you can remove a queue manager from the queue-sharing group. The same function-specific

parameters as for ADD QMGR are needed. With this parameter you can remove a queue manager only if it performed a normal shutdown.

- FORCE QMGR: If for some reason a queue manager abended, and you do not want to or cannot restart it, but you want to remove it from the queue-sharing group, you need to specify the FORCE QMGR parameter with the same function-specific parameters as you need for REMOVE QMGR.

Example 14-7 shows the parameters to add a queue-sharing group MQVG, a queue manager MQV1, and to remove this queue manager and queue-sharing group again.

Example 14-7 CSQ5PQSG parameters for QSG and QMGR add and remove

```
ADD QSG,MQVG,DB7VU,D7V1
ADD QMGR,MQV1,MQVG,DB7VU,D7V1

REMOVE QMGR,MQV1,MQVG,DB7VU,D7V1
REMOVE QSG,MQVG,DB7VU,D7V1
```

In addition to updating the DB2 tables, this utility also creates your XCF groups and adds XCF members to that group, depending on whether you are adding a queue-sharing group or a queue manager. The XCF group name has two parts in its name:

- The first part is always CSQG.
- The second part is the name of your queue-sharing group. If you are using a queue-sharing group name that is less than four characters long, you have to add the @ sign at the end of your name when displaying the group.

To display the XCF group and member, you can use the z/OS command D XCF,GROUP, as shown in Figure 14-13 on page 288.

```
D XCF,GROUP,CSQGMQVG,ALL
IXC333I  22.08.21  DISPLAY XCF 525
   INFORMATION FOR GROUP CSQGMQVG
   MEMBER NAME:        SYSTEM:     JOB ID:    STATUS:
   MQV1                SC53        MQV1MSTR   ACTIVE
   MQV2                SC61        MQV2MSTR   ACTIVE
   MQV3                SC66        MQV3MSTR   ACTIVE
   MQV4                SC62        MQV4MSTR   ACTIVE

 INFORMATION FOR GROUP CSQGMQVG MEMBER MQV3

 MEMTOKEN: 0A00001C 00530003      ASID: 03EF

 SIGNALLING SERVICE
   MSGO ACCEPTED:          0  NOBUFFER:          0

   MSGI RECEIVED:          0  PENDINGQ:          0
   MSGI XFER CNT:          0  XFERTIME:        N/A

 GROUP SERVICE
   EVNT RECEIVED:          0  PENDINGQ:          0
```

Figure 14-13 Display XCF group and members for a queue-sharing group

14.5.2 The dead-letter queue handler utility CSQUDLQH

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.

The CSQUDLQH utility program runs as a z/OS batch program. You need to specify the name of the dead-letter queue that you want to process and the queue manager on which it resides. In addition the following DD statements are required:

- SYSOUT - to name the data set for print output. You can specify the logical record length (LRECL) and block size (BLKSIZE) for this output data set.
- SYSIN - to identify the input data set that contains the rules table and specifies what the dead-letter queue handler is to do.

You can also use the PARM statement to overwrite the values specified in the SYSIN data set. Figure 14-14 on page 290 shows an example.

The following keywords can be included in a control-data entry in a DLQ handler rules table:

- INPUTQ (*QueueName*)

  Specifies the name of the DLQ that you want to process:

  – If you specify a queue name in the PARM parameter of the EXEC statement, this overrides any INPUTQ value in the rules table.

  – 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.

  – 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.

  – If you specify a queue manager name in the PARM parameter of the EXEC statement, this overrides any INPUTQM value in the rules table.

  – 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.

  – 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)*

  – 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 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 999999.

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.

There are also a lot of pattern-matching keywords, such as DESTQ, DESTQM, MSGTYPE and so on, which are those against which messages on the dead-letter queue are matched. And remember there are ACTION keywords such as DISCARD, IGNORE, RETRY, and so on.

Detailed information about the utility, the rules table, and its parameters can be found in the *WebSphere MQ for z/OS System Administration Guide*, SC34-6053.

Figure 14-14 shows a sample JCL where queue manager MQV4 and dead-letter queue MQV4.DEAD.QUEUE are used because of the specifications in the PARM statement.

```
//MQRES2   JOB
//READQ    EXEC PGM=CSQUDLQH,
//             PARM= MQV4.DEAD.QUEUE MQV4
//STEPLIB  DD  DISP=SHR,DSN=MQ530.SCSQAUTH
//         DD  DISP=SHR,DSN=MQ530.SCSQLOAD
//         DD  DISP=SHR,DSN=MQ530.SCSQANLE
//SYSOUT   DD  SYSOUT=*
//SYSIN    DD  *
  INPUTQM(MQV2) INPUTQ(MQV2.DEAD.QUEUE)
  ACTION(RETRY)
/*
```

Figure 14-14 Sample JCL to invoke the CSQUDLQH utility

14.5.3 The log print utility CSQ1LOGP

This utility can be used to print information contained in the:

- Bootstrap data set (BSDS)
- Active log data set (without BSDS)
- Archive log data set (without BSDS)

It runs as a batch program and the JCL to invoke CSQ1LOGP requires the following DD statements:

- 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.

- SYSSUMRY

  If a summary report is requested, the output is written to this data set. The logical record length (LRECL) is 131.

And one of the following:

- 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.

With WebSphere MQ 5.3, you are now able to run CSQ1LOGP against active logs even if your queue manager is up and running, provided that the active log data set and BSDS are defined using at least SHAREOPTIONS(2 3).

You can specify the following input control parameters:

- LRSNSTART(*hexadecimal-constant*) and LRSNEND*(hexadecimal-constant)*, which specify the logical record sequence number from which to begin to scan and the last record to be scanned.
- RBASTART(*hexadecimal-constant*) and RBAEND(*hexadecimal-constant*), which specify the log RBA from which to begin processing and the last valid RBA to be processed.
- PAGESET(*decimal-integer*), to identify the page set from which you want the information printed.
- URID(*hexadecimal-constant*) specifies the unit of recovery identifier.
- RM(*resource_manager*) specifies a particular resource manager, such as buffer manager or data manager.
- Summary(*YES* | *NO* | *ONLY*) specifies if a summary report should be produced.

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.

JCL and job output

Figure 14-15 shows a sample JCL to invoke CSQ1LOGP. The job output will show all entries of the recovery manager on page set 1 from log data set MQV1.LOGCOPY1.DS02. No summary report will be produced.

Figure 14-16 on page 293 shows the produced job output.

```
//MQRES2LP JOB
//PRTLOG   EXEC PGM=CSQ1LOGP
//STEPLIB  DD  DISP=SHR,DSN=MQ530.SCSQANLE
//         DD  DISP=SHR,DSN=MQ530.SCSQLOAD
//ACTIVE1  DD  DISP=SHR,DSN=MQV1.LOGCOPY1.DS02
//SYSPRINT DD  SYSOUT=*
//SYSSUMRY DD  SYSOUT=*
//SYSIN    DD  *
  PAGESET(1) SUMMARY(NO) RM(RECOVERY)
/*
```

Figure 14-15 JCL to invoke CS1LOGP

```
CSQ1102I SEARCH CRITERIA

SUMMARY(NO)
ALL URIDS
RM(RECOVERY)
PAGESET(00000001)

============================================================================
====

CSQ1212I FIRST LOG RBA ENCOUNTERED = 0000278D0000

0002913E000  RM(RECOVERY)  TYPE( RESTART - START CURRENT STATUS REBUILD )
**** 002A0000 00100003 03800000 00000000 00000000 00005320 B81E3BC6 95600001
0000  8000B81E 3BC6955F FF25

0002914F000  RM(RECOVERY)  TYPE( RESTART - START CURRENT STATUS REBUILD )
**** 002A0000 00100003 03800000 00000000 00000000 00005320 B81E40DA 3CBE0001
0000  8000B81E 40DA3CBE 2B84
```

Figure 14-16 Job output for CSQ1LOGP

For more information about CSQ1LOGP, refer to *WebSphere MQ for z/OS System Administration Guide Version 5 Release 3*, SC34-6053.

14.5.4 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 no further formatting is needed, and no performance benefit accrues.

This utility can only be used to format active logs before starting the queue manager that will use the logs. If you use it to format a log data set after the queue manager has started, the data will get lost.

The following DD statements should be provided:

- SYSPRINT: Required to specify a data set or print spool class for print output.
- SYSUT1: Required to identify the log data set to be preformatted.

15

Recovery scenarios

The following high-availability queue-sharing group configuration was designed to establish an application infrastructure that had no single point of failure.

Each component is implemented with a backup component to ensure that if an outage at one point occurs, a backup component would be able to take over the workload, thereby reducing application exposure to system component failures.

In this chapter we document the practical testing procedures that were carried out by this residency team to prove that the high-availability scenario does in fact work. We include an outline of all testing steps and the output that was produced by these steps.

15.1 Test environment description

During discussions on the best practice to establish a no single point of failure architecture during this residency, the scenario in Figure 15-1 was designed and tested to be recommended to IBM customers for implementation.

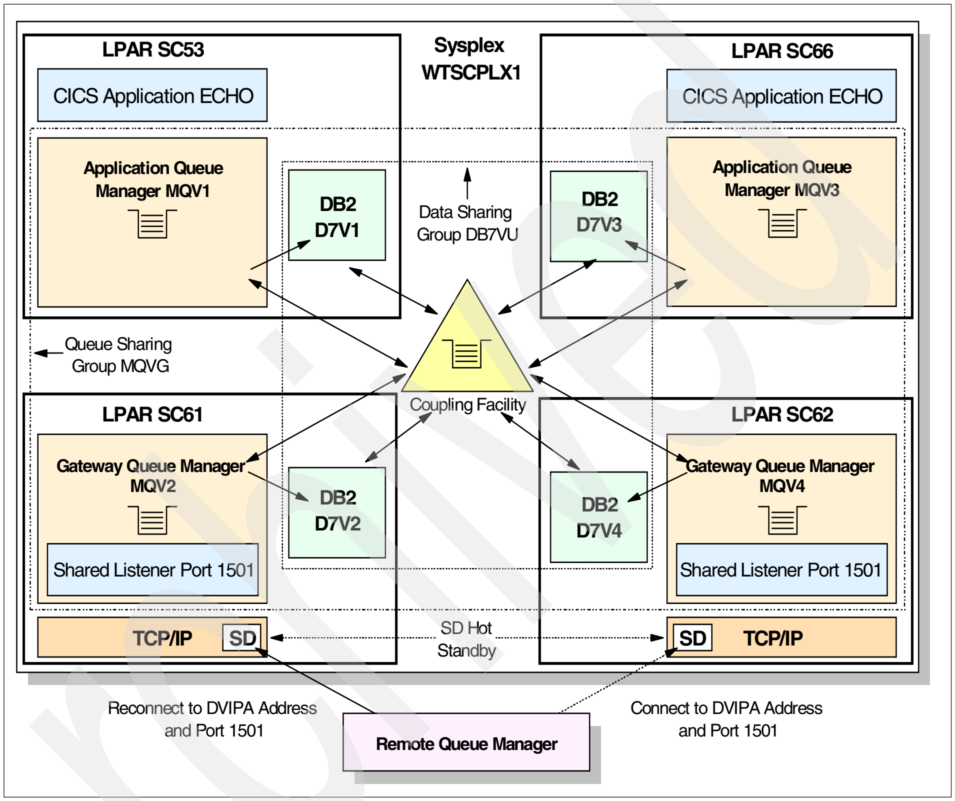


Figure 15-1 Application Implementation scenario

The test environment setup contains four queue managers (MQV1, MQV2, MQV3, and MQV4), which are all members of the queue-sharing group MQVG. Each queue manager connects to a local DB2 subsystem (D7V1, D7V2, D7V3, and D7V4), which are all members of a data-sharing group DB7VU. On LPAR SC61 and SC62 we have implemented Sysplex Distributor, and we want to use the queue managers running on these two LPARs to take on the role of gateway queue managers. On the other two LPARs, all applications are connected to the queue managers MQV1 and MQV3.

For information regarding the WebSphere MQ setup, refer to Chapter 9, “Creating a shared queue environment” on page 159.

Test scenarios

We considered several tests to show the functionality of our queue-sharing group setup:

1. Outage of the primary gateway queue manager MQV4. In this scenario, we want to show the impact of one missing gateway queue manager and to show a shared channel reconnecting to the second gateway queue manager.
2. Outage of the TCP/IP subsystem, which is configured to be the primary Sysplex Distributor running on the LPAR SC61. Our objectives for this test are:
   - Check system log for SD VIPA takeover messages
   - Check MQ log for shared channel reconnections
3. Outage of one of the application queue managers. The objective is to ensure that the second application queue manager resumes processing for our applications.
4. Outage of one CICS application region. The objective is to observe queue manager logs for error messages and the effect on message processing.
5. Outage of a DB2 subsystem. With this test, we want to check the impact of losing the DB2 connection:
   - Check the queue manager and channel initiator logs
   - Test the queue manager functionality without DB2 connection
6. Outage of the complete z/OS image (one single LPAR) of a gateway queue manager. Our objectives are:
   - Observe the effect on the remaining queue managers
   - Observe the recovery procedure, when restarting the z/OS image
7. Outage of a coupling facility where our WebSphere MQ structures reside. In this case, we want to:
   - Observe the effect on all queue managers in the queue-sharing group
   - Control the recovery of the WebSphere MQ structures
   - Observe WebSphere MQ message recovery of shared messages
   - Observe restart of coupling facility

15.2 Outage of primary gateway queue manager MQV4

To simulate the loss of one of the gateway queue managers, we canceled queue manager MQV4 running on LPAR SC62. The primary objective was to demonstrate how the fallback gateway queue manager MQV2 would recover the shared outbound channel from the queue-sharing group to the remote queue manager. We also wanted to verify that channel retry processing on the remote queue manager would reconnect to the queue-sharing group via MQV2. The workload generation script MQREQ5KS (Message flow #1) was used to simulate a production load for this test. Refer to 12.3, “Workload generation scripts” on page 238.

The following steps were followed to verify that recovery processing was successful:

1. The workload script MQREQ5KS was started to generate a workload.
2. We confirmed that the shared inbound and outbound channels were both connected to gateway queue manager MQV4. The remote queue manager used in this test was called M23WPH767. Figure 15-2 shows the shared inbound channel M23WPH767.TO.MQVG running on queue manager MQV4.

```
                              List Channels - MQV4                    Row 1

 Type action codes. Then press Enter.
  1=Display   2=Define like   3=Alter   4=Manage   5=Perform
  6=Start     7=Stop

     Name                 Type        Disposition   Status
 <>  M23WPH767.TO.MQVG     CHANNEL       COPY    MQVG
     M23WPH767.TO.MQVG     RECEIVER      COPY    MQV1  INACTIVE
     M23WPH767.TO.MQVG     RECEIVER      COPY    MQV2  INACTIVE
     M23WPH767.TO.MQVG     RECEIVER      COPY    MQV3  INACTIVE
     M23WPH767.TO.MQVG     RECEIVER      COPY    MQV4  RUN
                   ******** End of list ********
```

Figure 15-2 Shared inbound channels on MQV4.

Figure 15-3 on page 299 shows the shared outbound channel M23WPH767.TO.MQVG running on queue manager MQV4.

```
                              List Channels - MQV4                    Row
1 of 4

 Type action codes. Then press Enter.
  1=Display   2=Define like   3=Alter   4=Manage   5=Perform
  6=Start     7=Stop

     Name                Type         Disposition   Status
 <>  MQVG.M23WPH767        CHANNEL      COPY    MQVG
     MQVG.M23WPH767        SENDER       COPY    MQV1  INACTIVE
     MQVG.M23WPH767        SENDER       COPY    MQV2  INACTIVE
     MQVG.M23WPH767        SENDER       COPY    MQV3  INACTIVE
     MQVG.M23WPH767        SENDER       COPY    MQV4  RUN
                    ******** End of list ********
```

Figure 15-3 Shared outbound channels on MQV4

3. We then cancelled the gateway queue manager MQV4.
4. The joblogs for the fallback gateway were analyzed to verify that shared channel recovery had taken place on MQV2 and that our remote queue manager M23WPH767 was able to resynchronize its sender channel with the queue-sharing group via MQV2.

   Figure 15-4 on page 300 shows the joblog of the MQV2 master address space showing that shared channel recovery was successfully completed for the shared outbound channel MQVG.M23WPH767. Messages from the CHIN joblog are also included to show the successful channel restart for the shared outbound and inbound channels.

```
Shared channel recovery from MQV2MSTR joblog

18.18.37 STC26223  CSQM052I -MQV2 CSQMPCRT Shared channel recovery
                    completed for MQV4, 10 channels found, 8 FIXSHARED, 2
recovered


NQV2CHIN joblog showing shared outbound and inbound channels starting

18.18.37 STC26224  +CSQX500I -MQV2 CSQXRCTL Channel MQVG.M23WPH767 started

18.19:37 STC26224  +CSQX500I -MQV2 CSQXRCTL Channel M23WPH767.TO.MQVG
started
```

Figure 15-4 Successful channel recovery message (evidence from joblogs)

Figure 15-5 and Figure 15-6 on page 301 show the display channel commands showing that both shared channels are now active on the fallback gateway MQV2.

```
                              List Channels - MQV2                    Row
 1 of 4

  Type action codes. Then press Enter.
   1=Display  2=Define like  3=Alter  4=Manage  5=Perform
   6=Start    7=Stop

     Name                Type        Disposition  Status
  <>  M23WPH767.TO.MQVG    CHANNEL      COPY   MQVG
     M23WPH767.TO.MQVG    RECEIVER     COPY   MQV1  INACTIVE
     M23WPH767.TO.MQVG    RECEIVER     COPY   MQV2  RUN
     M23WPH767.TO.MQVG    RECEIVER     COPY   MQV3  INACTIVE
     M23WPH767.TO.MQVG    RECEIVER     COPY   MQV4  INACTIVE
                   ******** End of list ********
```

Figure 15-5 Display Channel commands showing inbound channels on MQV2

```
                              List Channels - MQV2
Row 1 of 4

 Type action codes. Then press Enter.
  1=Display   2=Define like   3=Alter   4=Manage   5=Perform
  6=Start     7=Stop

     Name                Type        Disposition   Status
 <>  MQVG.M23WPH767       CHANNEL      COPY    MQVG
    MQVG.M23WPH767       SENDER       COPY    MQV1  INACTIVE
    MQVG.M23WPH767       SENDER       COPY    MQV2  RUN
    MQVG.M23WPH767       SENDER       COPY    MQV3  INACTIVE
    MQVG.M23WPH767       SENDER       COPY    MQV4  INACTIVE
                  ******** End of list ********
```

Figure 15-6 Display Channel commands recovered outbound channels on MQV2

For this scenario, the shared outbound recovery took 23 seconds to be recovered. The shared inbound channel was restarted by a successful retry after 1 minute and 33 seconds. This time can be reduced by re-configuring the retry and heartbeat parameters on the sender channel definition.

15.3 Outage of the primary TCP/IP Sysplex Distributor

To simulate a severe TCP/IP problem where the TCP/IP region of one LPAR fails, we canceled the TCP/IP region of SC61, our DVIPA owning region. Before we canceled TCP/IP, we checked the connection routing table using the z/OS command D TCPIP,TCPIPA,N,VCRT for the number of connections currently going to SC61. As you can see in Figure 15-7 on page 302, there are two connections made for XCF address 10.1.100.61, which is the XCF address for LPAR SC61, and one connection is made with XCF address 10.1.100.62, which is LPAR SC62.

```
D TCPIP,TCPIPA,N,VCRT
EZZ2500I NETSTAT CS V2R10 TCPIPA 622
DYNAMIC VIPA CONNECTION ROUTING TABLE:
DEST IPADDR      DPORT  SRC IPADDR       SPORT  DESTXCF ADDR
9.12.8.10        01501  9.24.104.127     01360  10.1.100.61
9.12.8.10        01501  9.24.104.142     01209  10.1.100.61
9.12.8.10        01501  9.24.105.29      01294  10.1.100.62
3 OF 3 RECORDS DISPLAYED
```

Figure 15-7 Display of TCP/IP routing table

After we canceled TCP/IP, we first had a look at the messages issued for our DVIPA. As shown in Figure 15-8, the DVIPA is taken over by the backup system.

```
13:40:08.49 P TCPIPA
13:40:08.68 EZZ3205I SNMP SUBAGENT: SHUTDOWN IN PROGRESS
13:40:08.71 EZZ0673I AUTOLOG TASK: SHUTDOWN IN PROGRESS
13:40:08.78 EZZ6008I TELNET STOPPING
..........
13:40:09.00 EZZ8301I VIPA 9.12.8.10 TAKEN OVER FROM TCPIPA ON SC61
13.40.09.20 EZZ4323I CONNECTION TO 10.1.100.61 CLEARED FOR DEVICE SC61M
```

Figure 15-8 Cancel TCP/IP

Then we had a look at the messages printed to the JES message log of channel initiator MQV2, which is also on LPAR SC61. At the same time that TCP/IP goes down, the channel connection ends abnormally with a TCP/IP return code of 0000007A - I/O error (see Figure 15-9). On MQV4 we see a channel started message for the same channel that got the I/O error just a few seconds later (again Figure 15-9).

```
MQV2:
13.40.09 STC25520  +CSQX208E -MQV2 CSQXRESP Error receiving data,
                     channel M23WPF31.TO.MQVG,
                     connection 9.24.104.127
                     (queue manager ????)
                     TRPTYPE=TCP RC=0000007A
13.40.09 STC25520  +CSQX599E -MQV2 CSQXRESP Channel M23WPF31.TO.MQVG ended
abnormally

MQV4:
13.40.49 STC25528  +CSQX500I -MQV4 CSQXRESP Channel M23WPF31.TO.MQVG started
```

Figure 15-9 Messages issued for MQV2 and MQV4 at the time of TCP/IP outage

The display of the connection routing table in Figure 15-10 shows that all connections with SC61 are now handled by SC62.

```
D TCPIP,TCPIPA,N,VCRT
RESPONSE=SC62
 EZZ2500I NETSTAT CS V2R10 TCPIPA 935
 DYNAMIC VIPA CONNECTION ROUTING TABLE:
 DEST IPADDR      DPORT  SRC IPADDR       SPORT  DESTXCF ADDR
 9.12.8.10        01501  9.24.104.127     01796  10.1.100.62
 9.12.8.10        01501  9.24.104.142     01306  10.1.100.62
 9.12.8.10        01501  9.24.105.29      01428  10.1.100.62
 3 OF 3 RECORDS DISPLAYED
```

Figure 15-10 D TCPIP,TCPIPA

No manual intervention was necessary and apart from the fact that MQV2 cannot send or receive any messages at the moment because TCP/IP is not available, at least all messages destined for the shared queues within the queue-sharing group are not affected by the TCP/IP outage of SC61, because they can still be processed by MQV4 on SC62.

15.4 Outage of one of the application queue managers

To simulate the loss of one of the application queue managers, we canceled queue manager MQV3 running on LPAR SC66. The primary objective was to demonstrate how the remaining application queue manager MQV1 running on SC53 would take over the remaining workload. We also wanted to show that the remaining queue managers in the queue-sharing group would handle the recovery of any in-flight units of work started by the failed queue manager (peer recovery). The workload generation script MQREQ5KS (Message flow #1) was used to simulate a production load for this test. Refer to 12.3, "Workload generation scripts" on page 238 for details.

The following steps were followed to verify that recovery processing was successful:

1. The workload script `MQREQ5KS` was started to generate a workload.
2. After observing that both CICS regions were successfully processing transactions, we canceled the application queue manager MQV3.
3. The joblogs for the remaining queue managers in the QSG were analyzed to see if any peer recovery was performed. Peer recovery allows another queue manager in the QSG to handle the backout of any UOWs involving shared queue messages that may have been started by MQV3. Figure 15-11 on

page 304 shows that queue manager MQV1 has successfully completed a UOW recovery involving a coupling facility structure called APPLICATION1.

```
14.07.19 STC24733  CSQE008I -MQV1 Recovery event from MQV3 received for
                   structure APPLICATION1
14.07.19 STC24733  CSQE011I -MQV1 Recovery phase 1 started for structure
                   APPLICATION1 connection name CSQEMQVGMQV303
14.07.20 STC24733  CSQE013I -MQV1 Recovery phase 1 completed for
                   structure APPLICATION1 connection name CSQEMQVGMQV303
14.07.22 STC24733  CSQE008I -MQV1 Recovery event from MQV3 received for
                   structure APPLICATION1
14.07.22 STC24733  CSQE012I -MQV1 Recovery phase 2 started for structure
                   APPLICATION1 connection name CSQEMQVGMQV303
14.07.23 STC24733  CSQE014I -MQV1 Recovery phase 2 completed for
                   structure APPLICATION1 connection name CSQEMQVGMQV303
```

Figure 15-11 Peer recovery performed by queue manger MQV1

4. We monitored the progress of the workload script MQREQ5KS to check that all the remaining requests were processed by queue manager MQV1 and its associated CICS region CICSLSA1.

15.5 Outage of one CICS application region

This section deals with losing one of the CICS Transaction Server regions while our workload is running. We provided a failover scenario by using two CICS Transaction Server regions that run a CICS adapter and trigger monitor instances on each of them. We also used a shared application and initiation queue that are part of the queue-sharing group configuration described earlier in this chapter.

The impact of losing one of our CICS Transaction Server regions should be minimum. We avoided single point of failure scenarios and expected the workload to continue using the remaining CICS Transaction Server environment, CICS adapter, and trigger monitor.

15.5.1 Failover scenario

In order to simulate an outage of a CICS Transaction Server region, we simply cancel the region that causes an uncontrolled shutdown of CICS. When we cancel the region, CICS cannot do any shutdown processing. Therefore the CICS adapter and the trigger monitor as well as all active ECHO transactions just abend immediately. The subsequent CICS startup must therefore be START=AUTO, in order to allow an emergency startup that will resolve any uncommitted UOWs.

We have associated a TRANCLASS definition with the TRANSACTION definition for ECHO, and we defined a maximum of eight ECHO tasks that can run concurrently in one CICS region. We discovered that usually two or three ECHO transactions are running concurrently in CICS when the workload is active. Therefore we expect that we may lose trigger messages and that there might be uncommitted units of work (UOW) left for ECHO tasks when the CICS region will come down abnormally.

We performed the following steps to simulate an outage of one of the CICS Transaction Server regions:

1. Start the WebSphere MQ client workload. The workload sends 5000 messages to the shared application queue. Since the queue is defined as TRIGTYPE=EVERY, a trigger message is generated by queue manager MQV1 and MQV3 alternatively. The ECHO transaction gets distributed to both CICS Transaction Server regions.
2. While the workload is running, we cancel CICS Transaction Server region CICSLSA5, which is running on SC66. We keep the joblogs of queue manager MQV3 and CICS to check if there are any conspicuous messages.
3. We bring up the CICS Transaction Server region again using SIT parameter START=AUTO. The CICS emergency startup joblog may show any evidence that the CICS recovery manager drives backout or commit processing for UOWs that were in flight when CICS came down abnormally.
4. Check queue statistics of both queue managers MQV1 and MQV3 in order to check that there are no messages left on the shared application queue. The shared application queue should be checked for the number of trigger messages that have been generated.

15.5.2 Impact

We started the workload and canceled region CICSLSA5. Then we verified that the workload continued processing on CICS region CICSLSA1 running on SC53. It turned out that the workload immediately went to the remaining CICS region.

Example 15-1 CICSLSA5 joblog

```
   808            SYSTEM COMPLETION CODE=122
   808             TIME=13.39.14  SEQ=00796  CPU=0000  ASID=0097
   808             PSW AT TIME OF ERROR  078D1000   9084C69C  ILC 2  INTC 01
   808               ACTIVE LOAD MODULE           ADDRESS=10800000  OFFSET=0004C69C
   808               NAME=DFHSIP
   808               DATA AT PSW  1084C696 - 41101000  0A0147F0  57745820
   808               GR 0: 00000001   1: 1098F0D4
   808                  2: 1098F048   3: 9084B0F6
   808                  4: 1094E020   5: 9084C0F6
   808                  6: 9084D0F6   7: 10990060
   808                  8: 109900B4   9: 1084D958
   808                  A: 00000000   B: 1094DC80
   808                  C: 00000000   D: 1094E020
   808                  E: 9084C86A   F: 807F9078
   808             END OF SYMPTOM DUMP
13.39.14 STC25613  +DFHKE0201 SCSCLSA5 ABOUT TO TAKE SDUMP. DUMPCODE: KERNDUMP, DUMPID: 0/0000. (MODULE DFHKESTX).
13.39.16 STC25613  +DFHKE1799 SCSCLSA5 TERMINATION OF CICS IS COMPLETE.
13.39.16 STC25613  IEF450I CICSLSA5 CICSLSA5 - ABEND=S122 U0000 REASON=00000000  842
   842                  TIME=13.39.16
13.39.16 STC25613  -                                         --TIMINGS (MINS.)--            ----PAGING COUNTS---
13.39.16 STC25613  -JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  CLOCK   SERV  PG   PAGE   SWAP    VIO SWAPS
13.39.16 STC25613  -CICSLSA5 CICSTS   CICSLSA5 *S122  13703   5.41    .09 1280.5  1853M   0      0      0      0     0
13.39.16 STC25613  -CICSLSA5 ENDED.  NAME-                     TOTAL CPU TIME=  5.41  TOTAL ELAPSED TIME=1280.5
```

The joblog of CICSLSA5 in Example 15-1 did not show any WebSphere MQ related messages. It just shows the termination process as a result of the cancel command. We collected the queue manager joblog of MQV3 as well. Example 15-2 shows the partial joblog of MQV3 when the CICS outage occurred.

Example 15-2 Joblog of MQV3

```
   827                  CONNECTION-ID=SCSCLSA5 THREAD-XREF=
13.39.15 STC24732  CSQ3201E -MQV3 ABNORMAL EOT IN PROGRESS FOR USER=. 828
   828                  CONNECTION-ID=SCSCLSA5 THREAD-XREF=
13.39.15 STC24732  CSQ3201E -MQV3 ABNORMAL EOT IN PROGRESS FOR USER=STC. 829
   829             CONNECTION-ID=SCSCLSA5 THREAD-XREF=
13.39.15 STC24732  CSQ3201E -MQV3 ABNORMAL EOT IN PROGRESS FOR USER=STC. 830
   830             CONNECTION-ID=SCSCLSA5 THREAD-XREF=
13.39.15 STC24732  CSQ3201E -MQV3 ABNORMAL EOT IN PROGRESS FOR USER=STC. 831
   831             CONNECTION-ID=SCSCLSA5 THREAD-XREF=
13.39.15 STC24732  CSQ3201E -MQV3 ABNORMAL EOT IN PROGRESS FOR USER=STC. 832
   832             CONNECTION-ID=SCSCLSA5 THREAD-XREF=
13.39.15 STC24732  CSQ3201E -MQV3 ABNORMAL EOT IN PROGRESS FOR USER=STC. 833
   833             CONNECTION-ID=SCSCLSA5 THREAD-XREF=
13.39.15 STC24732  CSQ3201E -MQV3 ABNORMAL EOT IN PROGRESS FOR USER=STC. 834
   834             CONNECTION-ID=SCSCLSA5 THREAD-XREF=
```

When CICS was canceled, the main connection between WebSphere MQ and the CICS adapter terminated as well. The CICS adapter has attached eight additional subtasks to remove work from the main CICS TCB. Each of the additional subtasks issue a connect call to WebSphere MQ. A CICS system is using nine of the connections specified in the CTHREAD system parameter. The queue manager joblog shown in Example 15-2 on page 306 shows the termination messages of these connections.

When the workload finished, we checked the queue statistics for queue managers MQV3 and MQV1. We realized that there were no messages left on the shared application queue. We obviously have not lost any trigger messages and all the messages have been processed. All reply messages came back as well.

Example 15-3 Queue statistics MQV1

```
MQV1                   MSGS IN          MSGS OUT
CICS.MQVG.ECHO01       0                4138
CICS.MQVG.INITQ01      4120             4143
```

Example 15-4 Queue statistics MQV3

```
MQV3                   MSGS IN          MSGS OUT
CICS.MQVG.ECHO01       0                862
CICS.MQVG.INITQ01      892              869
```

The statistics in Example 15-3 and Example 15-4 illustrate that we have processed 5000 messages. The fact that we do not have any missing triggers or messages left on the application queue explains that there are no WebSphere MQ related UOWs to be backed out during the subsequent emergency restart of CICS.

15.6 Outage of a DB2 subsystem

In this section, we not only want to examine the outage of a DB2 subsystem because of a DB2 failure, but also want to see what happens if the DB2 subsystem a queue manager is connected to is stopped. We do not show the DB2 recovery, but only examine the impact on the queue manager.

15.6.1 Planned outage of a DB2 subsystem at the gateway

We have a look at the situation when DB2 is stopped normally and how the queue manager connected to this DB2 subsystems reacts. Depending on the connection setup you specified in the QSGDATA parameter, we have two different scenarios: connection to a specific DB2 subsystem on the one hand, and using the DB2 group attachment for your connection on the other hand.

Connection to a specific DB2 subsystem

The queue manager is informed that DB2 is no longer available. Example 15-5 shows these messages.

Example 15-5 DB2 stopped messages in queue manager address space

```
CSQ5016I -MQV4 CSQ5MONR Connection to D7V4 quiescing, DB2 terminating
CSQ5019I -MQV4 CSQ5DISC Disconnected from DB2 D7V4
CSQ5003A -MQV4 CSQ5CONN Connection to DB2 using D7V4
pending, no active DB2
```

In the channel initiator address space, you see message CSQX483E, telling you that DB2 is not available.

Since the DB2 connection is no longer available, no more updates to DB2 are possible. This can occur in several situations:

- Sending commands to queue manager, which require access to DB2 tables. An example for this would be a display of shared queues requested through ISPF application on the queue manager, where the connection is broken. You then receive the following message:

  ```
  CSQM294I -MQV4 CSQMDRTS CANNOT GET INFORMATION FROM DB2
  ```

- For group definitions done on other queue managers in the queue-sharing group, the local copy of the queue manager where DB2 is down cannot be created. On the queue manager where you issue the command, you get error messages. In Figure 15-12 on page 309, we show the error messages we get when defining a group sender channel.

```
CSQM122I -MQV2 CSQMACHL ' DEFINE CHANNEL' COMPLETED FOR QSGDISP(GROUP)

CSQN138I -MQV2 'DEFINE CHANNEL' command generated for CMDSCOPE(*),
sent to 4
CSQM125I -MQV1 CSQMACHL CHANNEL(TEST.DB2.DOWN) QSGDISP(GROUP) WAS NOT
FOUND
CSQM090E -MQV1 CSQMACHL FAILURE REASON CODE X'00D44002'
CSQ9023E -MQV1 CSQMACHL ' DEFINE CHANNEL' ABNORMAL COMPLETION
CSQ9022I -MQV2 CSQMACHL ' DEFINE CHANNEL' NORMAL COMPLETION
CSQ9022I -MQV3 CSQMACHL ' DEFINE CHANNEL' NORMAL COMPLETION
CSQ9022I -MQV4 CSQMACHL ' DEFINE CHANNEL' NORMAL COMPLETION
```

Figure 15-12 Errors defining group definitions

The DEFINE CHANNEL completes normally on three queue managers. In this test scenario, DB2 D7V1 was stopped, and queue manager MQV1 had no connection to a DB2 subsystem anymore.

After restarting the DB2 subsystem D7V1 we check the channel information on MQV1. No copy definition of the group channel definition exists on MQV1. See Figure 15-13.

```
                             List Channels - MQVG                  Row 1 of 4

Type action codes. Then press Enter.
 1=Display   2=Define like   3=Alter   4=Manage   5=Perform
 6=Start     7=Stop

    Name                     Type           Disposition   Status
<>  TEST*                    CHANNEL        ALL     MQVG
    TEST.DB2.DOWN            SENDER         COPY    MQV2  INACTIVE
    TEST.DB2.DOWN            SENDER         COPY    MQV3  INACTIVE
    TEST.DB2.DOWN            SENDER         COPY    MQV4  INACTIVE
    TEST.DB2.DOWN            SENDER         GROUP
                       ******** End of list ********
```

Figure 15-13 Channel display after failure with message CSQM090E

Selecting option 4 (Manage) at the group definition, you can then on the next display create the local copy of your group channel definition.

We also experienced some abends during the time DB2 connection was not available. Example 15-6 on page 310 shows the dump produced.

Example 15-6 Dump produced when DB2 is not active

```
IEA794I SVC DUMP HAS CAPTURED:
DUMPID=015 REQUESTED BY JOB (MQV4MSTR)
DUMP TITLE=MQV4,ABN=0C6-00000006,U=MQRES1  ,C=F1000.530.MMC -CS
           QM1DQQ,M=CSQGFRCV,LOC=CSQMLPLM.CSQM1DQQ+01C6
IEA794I SVC DUMP HAS CAPTURED:
DUMPID=016 REQUESTED BY JOB (MQV4MSTR)
DUMP TITLE=MQV4,ABN=0C4-00000011,U=MQRES1  ,C=F1000.530.MMC -CS
           QM1DQT,M=CSQGFRCV,LOC=CSQMLPLM.CSQM1DQT+017A
```

Although shared channel connections that are currently active are not influenced, you cannot start or stop a shared channel anymore.

If you have defined a shared receiver channel and you start the corresponding sender channel after DB2 subsystem comes down, you may get the following message:

```
CSQX599E -MQV4 CSQXRESP Channel M23WPM44.TO.MQVG ended abnormally
```

This is because to start a shared channel, the queue managers need to update DB2 shared channel status tables. The Sysplex Distributor still distributes connection requests to both gateway queue managers. Then the queue manager that has no DB2 connection may receive the channel start request, which it cannot process. The channel remains in a retry status. This might prevent you from restarting the channel on another queue manager in the queue-sharing group.

To avoid this problem, you can stop the group listener on the affected queue manager. No new shared channel request is sent to this queue manager, but the currently active channel connections remain active.

In all cases, the problems are not necessarily seen at the same time that the DB2 connection is lost. When the DB2 subsystem is available again, the queue manager reconnects to the DB2 subsystem and normal processing continues.

Connection to DB2 group attachment

If your queue-sharing group parameter (QSGDATA) in the system parameter module of your queue manager points to a group attachment, and not to a local DB2 subsystem, there are two possible ways to avoid an outage of the queue manager that is affected by stopping your DB2 subsystem:

1. First, you are able to restart the queue manager on any other z/OS image where a DB2 subsystem of the same data-sharing group as the one you want to stop runs, and where you have set up your TCP/IP Sysplex Distributor. In our environment, this would be LPAR SC61.
2. If you have a second DB2 subsystem running on the same LPAR that also

belongs to the same data-sharing group, no action is required. WebSphere MQ queue managers are able to switch the connection to the available DB2 subsystem. Example 15-7 shows the almost immediate reconnection after one DB2 subsystem stops.

Example 15-7 Queue manager DB2 reconnection with group attachment

```
12.11.06 STC24733  CSQ5016I -MQV1 CSQ5MONR Connection to D7V3 quiescing, DB2
                                                          terminating
12.11.06 STC24733  CSQ5019I -MQV1 CSQ5DISC Disconnected from DB2 D7V3
12.11.06 STC24733  CSQ5001I -MQV1 CSQ5CONN Connected to DB2 D7V1
12.13.52 STC24733  CSQ5016I -MQV1 CSQ5MONR Connection to D7V1 quiescing, DB2
                                                          terminating
12.13.52 STC24733  CSQ5019I -MQV1 CSQ5DISC Disconnected from DB2 D7V1
12.13.52 STC24733  CSQ5003A -MQV1 CSQ5CONN Connection to DB2 using D7VG
                   pending, no active DB2
12.17.46 STC24733  CSQ5001I -MQV1 CSQ5CONN Connected to DB2 D7V1
```

To test this scenario we reconfigured MQV1 to start with a group attachment name for DB2NAME in the QSGDATA parameter. Then we stopped the DB2 subsystem D7V3 on SC66 and restarted it on SC53. In Example 15-7 we already have stopped D7V1 and queue manager MQV1 has a connection to DB2 subsystem D7V3. We stop D7V3, and the queue manager connects to D7V1.

The second part of this example shows the disconnection messages if all DB2 subsystems belonging to the data-sharing group on the LPAR where MQV1 is up and running are stopped. In message CSQ5003A you can see that WebSphere MQ uses the group attachment of DB2 to connect to a DB2 subsystem.

15.6.2 Failure of a DB2 subsystem

To simulate an outage of a DB2 subsystem, we use the following command:

F *irlmproc*,ABEND

to request the IRLM of our DB2 subsystem to abend. This also forces DB2 to abend. We tested two scenarios:

- Outage of an DB2 at the application side
- Outage of an DB2 at the gateway side

In both cases the queue manager connected to the abending DB2 gets error messages, shown in Example 15-8.

Example 15-8 Connection loss, due to DB2 abend

```
CSQ5014I -MQV1 CSQ5MONR Connection to D7V1 lost, DB2
terminated abnormally
CSQ5003A -MQV1 CSQ5CONN Connection to DB2 using D7VG
```

```
pending, no active DB2
CSQ5001I -MQV1 CSQ5CONN Connected to DB2 D7V1
```

In contrast with a normal stop of DB2, you can now see that the connection to DB2 is lost because DB2 terminated abnormally, and therefore DB2 tasks of the queue manager MQV1 may not have been completed.

Table locks might be discovered

Because of the lost connection, there may be some unresolved locks of DB2 tables, which result in message CSQ5023E, shown in Example 15-9. The table locked during our recovery test is CSQ.OBJ_B_QUEUE and contains the queue object definitions for shared queues.

Example 15-9 DB2 table lock, after DB2 subsystem outage

```
CSQ5023E -MQV1 CSQ5UOBJ SQL error, table
CSQ.OBJ_B_QUEUE locked by another DB2
```

At the moment the locks are resolved, the table can again be accessed by all queue managers.

15.7 Outage of the z/OS image SC61

Let's say you plan an outage of a z/OS image, for example to upgrade to a new version of your operating system. You might want to restart your applications and subsystems on different LPARs, depending on how critical your applications are and how long your planned outage of the LPAR lasts.

In this case, you are able to shut down your queue managers and other subsystems without the need for recovery processes.

If you plan to restart your queue managers on different z/OS images, you should consider the following:

- Specifying a DB2 group attachment name, instead of a specific DB2 subsystem name, allows you to restart your queue manager on another system in the sysplex where any of the DB2 members of the data-sharing group runs.
- If you are using TCP/IP Sysplex Distributor support, you have to take care that the VIPA address is available on each z/OS system where you want to support queue managers with shared listener ports.
- Restarting queue managers that are serving applications requires that you restart the applications on the same system.

- The local listener address might not use the VIPA address. Therefore you should consider working with DNS services to allow a dynamic resolution of a host name to an IP address. You also should take care that you have configured your local listener ports properly.

15.7.1 Simulated hardware failure of LPAR SC61

To simulate a hardware outage of a whole LPAR where one of our gateway queue managers are active, we deactivate the LPAR through the hardware console.

In this scenario, we are not only confronted with WebSphere MQ recovery, but also with recovery of system components, for example RRS and System Logger, as well as the TCP/IP Sysplex Distributor function and the local DB2 subsystem to which the queue manager on this LPAR is connected.

After the failure of LPAR SC61, the remaining systems and subsystems that use sysplex functions detect errors and write messages to the system log. One of these errors that appears at the very beginning is the loss of the PATHIN devices for XCF communication.

As soon as it is known in the sysplex that the system SC61 is no longer available, a message to confirm the downfall of the system is issued. We confirm this message. Example 15-10 shows the messages and our reply.

Example 15-10 System down message flow

```
210 IXC102A XCF IS WAITING FOR SYSTEM SC61 DEACTIVATION. REPLY DOWN
WHEN MVS ON SC61 HAS BEEN SYSTEM RESET
R 210,DOWN
IEE600I REPLY TO 210 IS;DOWN
```

After this message a cleanup of coupling facility structures, for example the list structures used to communicate with other systems, is done. Other components that start cleanup include JES2, System Logger, and UNIX System Services.

The TCP/IP stack on our second gateway LPAR takes over the role of sysplex distribution management. See also 15.3, “Outage of the primary TCP/IP Sysplex Distributor” on page 301, for a description of the Sysplex Distributor recovery.

The DB2 subsystem D7V1 is disconnected from the data-sharing group. After the disconnect, a cleanup of all lock structures of DB2 is done.

Since the queue manager MQV2 is no longer available, sending messages through active channels to this queue manager fails. Example 15-11 on page 314 shows the failure messages for channel M23WPF31.TO.MQVG.

Example 15-11 Failure message for a channel after system outage of SC61

```
+CSQX609E -MQV4 CSQXRESP Resources in recovery, 062
 channel M23WPF31.TO.MQVG,
 MQCC=2 MQRC=2350
+CSQX599E -MQV4 CSQXRESP Channel M23WPF31.TO.MQVG ended abnormally
+CSQX514E -MQV4 CSQXRESP Channel M23X2517.MQVG is active on MQV4
```

The reason code 2350 (MQRC_CONN_TAG_NOT_USABLE) means that an MQCONNX specifying one of the MQCON_*_CONN_TAG_* options failed, because the connection tag specified by *ConnTag* in MQCNO is being used by the queue manager for recovery processing, and this processing is delayed pending recovery of the coupling facility.

After the channel recovery for shared channels has completed, the queue manager that actually did the recovery sends messages to the console and the channel restarts successfully, as shown in Example 15-12.

Example 15-12 Channel recovery complete messages

```
CSQM052I -MQV3 CSQMPCRT Shared channel recovery 610
completed for MQV2, 3 channels found, 0 FIXSHARED, 3 recovered
+CSQX500I -MQV4 CSQXRESP Channel M23WPF31.TO.MQVG started
```

In the message you can see the term FIXSHARED. Channels started with CHLDISP(FIXSHARED) are tied to the specific queue manager. That means if the channel initiator on that queue manager stops for any reason, the channels are not recovered by another queue manager in the group.

All channels that are configured local on MQV2 cannot restart until MQV2 is available again.

Another recovery action that has to be done by all queue managers in the queue-sharing group is to recover the application structures. Each active queue manager of the queue-sharing group receives a recovery event for the failed structure. The recovery is started on each queue manager after a successful completion message CSQE013I is issued. Example 15-13 shows the messages issued by queue manager MQV3. MQV1 and MQV4 produce similar output.

Example 15-13 Application structure recovery event and recovery

```
CSQE008I -MQV3 Recovery event from MQV2 received for 735
structure APPLICATION1
CSQE011I -MQV3 Recovery phase 1 started for structure 736
APPLICATION1 connection name CSQEMQVGMQV202
CSQE013I -MQV3 Recovery phase 1 completed for 737
structure APPLICATION1 connection name CSQEMQVGMQV202
```

In addition we see message IXL041E for all queue managers. The explanation for this message is that an XES process is in a possible hang condition because the indicated connector did not provide a required response to an event. Example 15-14 shows the message flow for queue manager MQV1.

Example 15-14 IXL014E during LPAR failure

```
IXL041E CONNECTOR NAME: CSQEMQVGMQV101, JOBNAME: MQV1MSTR, ASID: 03EB 396
HAS NOT RESPONDED TO THE DISCONNECTED/FAILED CONNECTION EVENT FOR
SUBJECT CONNECTION: CSQEMQVGMQV202.
DISCONNECT/FAILURE PROCESSING FOR STRUCTURE MQVGAPPLICATION1
CANNOT CONTINUE.
MONITORING FOR RESPONSE STARTED: 08/26/2002 14:22:06.
DIAG: 0000 0000 00000000
CSQE008I -MQV1 Recovery event from MQV2 received for 397
structure APPLICATION1
CSQE012I -MQV1 Recovery phase 2 started for structure 398
APPLICATION1 connection name CSQEMQVGMQV202
CSQE014I -MQV1 Recovery phase 2 completed for 400
structure APPLICATION1 connection name CSQEMQVGMQV202
IXL043I CONNECTOR NAME: CSQEMQVGMQV101, JOBNAME: MQV1MSTR, ASID: 03EB  399
HAS PROVIDED THE REQUIRED RESPONSE. THE REQUIRED RESPONSE
FOR THE DISCONNECTED/FAILED CONNECTION EVENT
FOR SUBJECT CONNECTION CSQEMQVGMQV202,
STRUCTURE MQVGAPPLICATION1 IS NO LONGER EXPECTED.
```

We see, that the queue manager MQV1 again gets a recovery event, and afterwards the message IXL043I indicates that the previoulsy missing response was provided. This also happens for queue managers MQV3 and MQV4.

The test applications that we had started before the deactivation of the LPAR, and which were using channel connections to shared channel receivers on MQV2, could restart the channels after the channel recovery process is done by MQV4.

Both application queue managers kept running and our CICS applications were not influenced by the problem. The reply messages were successfully sent back to the clients.

15.7.2 ARM processing

If you are using ARM to restart several subsystems, then these subsystems are candidates to be restarted on different z/OS images, according to the ARM policy you defined.

Since we had some configuration inconsistencies, ARM restarted D7V2 and MQV2 on LPARs we did not expect. We had no problem with D7V2, and the recovery of the DB2 subsystem worked as expected. We only had to restart DB2 subsystem on SC61.

Because ARM restarted MQV2 on a system where the old early code was still installed, MQV2 failed with an abend. Example 15-15 shows the dump message of MQV2.

Example 15-15 Wrong queue manager restart

```
CSQY010E -MQV2    CSQYASCP LOAD MODULE CSQ3EPX  IS NOT AT THE CORRECT
RELEASE LEVEL
IEA995I SYMPTOM DUMP OUTPUT 830
SYSTEM COMPLETION CODE=5C6
```

To avoid these problems we change our ARM policy to:

- Make a group for the DB2 subsystem and the queue manager.
- Restrict the start of the queue manager and the DB2 subsystem to a specific list of MVS systems.

15.7.3 Restart of SC61 - return to normal processing

After solving the hardware problems (in our case these were imaginary hardware problems, so we actually had no need of any action), system SC61 should be restarted. We experienced no problems at IPL of SC61.

Since we do not use any system automation tool to start all subsystems in the correct order, we need to restart our subsystems by ourselves. You might want to use a systems automation tool to process a complete IPL and restart all subsystems, but then you should take care that your queue manager is started after the DB2 subsystem and TCP/IP are available.

In our scenario, neither DB2 nor TCP/IP were available, so in addition to the recovery messages of MQV2, we had error messages due to the missing DB2 subsystem and problems when starting the listeners, because TCP/IP was not available.

The queue manager started task output shows information about the recovery. Since the BSDSs and page sets were not closed correctly, message IEC161I is displayed for those data sets. After opening the BSDSs, the queue manager issued message CSQ5003A to indicate that the required DB2 subsystem is not available. Example 15-16 on page 317 shows the bootstrap and log information, and message CSQ5003A.

Example 15-16 DB2 connection pending

```
CSQJ127I -MQV2 SYSTEM TIME STAMP FOR BSDS=2002-08-26 11:54:06.91
CSQJ001I -MQV2 CURRENT COPY 1 ACTIVE LOG DATA SET IS 295
DSNAME=MQV2.LOGCOPY1.DS02, STARTRBA=0000278D0000 ENDRBA=00002BF1FFFF
CSQJ099I -MQV2 LOG RECORDING TO COMMENCE WITH 296
STARTRBA=000028B04000
CSQ5003A -MQV2 CSQ5CONN Connection to DB2 using D7V2 297
pending, no active DB2
```

The next action was to restart DB2 subsystem D7V2. At the moment D7V2 is available, the queue manager MQV2 issues CSQ5001I to show the connection to DB2, and continues the restart processing. Example 15-17 shows the messages for the initiated restart.

Example 15-17 Restart messages for MQV2

```
CSQR001I -MQV2 RESTART INITIATED
CSQR003I -MQV2 RESTART - PRIOR CHECKPOINT RBA=000028AB1000
CSQR004I -MQV2 RESTART - UR COUNTS - 380
IN COMMIT=1, INDOUBT=0, INFLIGHT=0, IN BACKOUT=0
CSQR007I -MQV2 UR STATUS 381
T  CON-ID       THREAD-XREF        S    URID          TIME
- -------- ------------------------ ------------- -------------------
S MQV2     201.SCAVNG01             C000028B03A3C 2002-08-26 14:18:35
```

Once a commit UR is detected, it is resolved in the following recovery processes. The queue manager connects to the administrative structure and to the application structure and starts the recovery phases for the application structure. Example 15-18 shows the recovery messages.

Example 15-18 Recovery phases of MQV2

```
CSQE005I -MQV2 Structure CSQ_ADMIN connected as
CSQEMQVGMQV202, version=B822FAFBC5798EC3 00010035
IXL014I IXLCONN REQUEST FOR STRUCTURE MQVGCSQ_ADMIN
WAS SUCCESSFUL.  JOBNAME: MQV2MSTR ASID: 007E
CONNECTOR NAME: CSQEMQVGMQV202 CFNAME: CF05
CSQE021I -MQV2 Structure CSQ_ADMIN connection as
CSQEMQVGMQV202 warning, RC=00000004 reason=02010407 codes=00000000
00000000 00000000
IXL014I IXLCONN REQUEST FOR STRUCTURE MQVGAPPLICATION1
WAS SUCCESSFUL.  JOBNAME: MQV2MSTR ASID: 007E
CONNECTOR NAME: CSQEMQVGMQV202 CFNAME: CF05
CONNECTOR NAME: CSQEMQVGMQV202 CFNAME: CF05
CSQE005I -MQV2 Structure APPLICATION1 connected as
CSQEMQVGMQV202, version=B822FB0CA8358FC0 0004001E
CSQE011I -MQV2 Recovery phase 1 started for structure
APPLICATION1 connection name CSQEMQVGMQV202
```

```
CSQE013I -MQV2 Recovery phase 1 completed for
structure APPLICATION1 connection name CSQEMQVGMQV202
CSQE012I -MQV2 Recovery phase 2 started for structure
APPLICATION1 connection name CSQEMQVGMQV202
CSQE014I -MQV2 Recovery phase 2 completed for
structure APPLICATION1 connection name CSQEMQVGMQV202
CSQE006I -MQV2 Structure APPLICATION1 connection name
CSQEMQVGMQV202 disconnected
CSQR030I -MQV2 Forward recovery log range
from RBA=000028AB1000 to RBA=000028B03B30
CSQR005I -MQV2 RESTART - FORWARD RECOVERY COMPLETE -
IN COMMIT=0, INDOUBT=0
CSQR032I -MQV2 Backward recovery log range
from RBA=000028B03B30 to RBA=000028B03B30
CSQR006I -MQV2 RESTART - BACKWARD RECOVERY COMPLETE -
INFLIGHT=0, IN BACKOUT=0
CSQR002I -MQV2 RESTART COMPLETED
```

We do not care about the warning message CSQE021I when the queue manager reconnects to the administrative structure. Return code 4 means that the connection was completed, and bytes 3 and 4 of the reason code ('0407'X) mean that additional information is put to the CONAFLAGS field. CONAFLAGS contains flags that indicate one or more of the following: connector has been reconnected, rebuild in progress, rebuild stop in progress, alter in progress, or a user syncpoint event is set.

At the restart of the channel initiator, we encountered another problem: our listeners could not be started. Example 15-19 shows the messages CSQX220E and CSQX234I, showing the communications network is not available. To correct this error, we only had to restart TCP/IP.

Example 15-19 Listener start failure

```
+CSQX220E -MQV2 CSQXLSTT Communications network not available,
 channel ????,
 TRPTYPE=TCP
+CSQX234I -MQV2 CSQXLSTT Listener stopped, TRPTYPE=TCP INDISP=GROUP
```

15.8 Outage of a coupling facility

In our last disaster test we deactivated the coupling facility LPAR to simulate a coupling facility outage.

At startup, when the queue manager connects to the ADMIN structure, the message shown in Figure 15-14 on page 319 is issued, which shows us that we are connected to CF06.

```
XL014I IXLCONN REQUEST FOR STRUCTURE MQVGCSQ_ADMIN WAS SUCCESSFUL.  JOBNAME:
MQV1MSTR ASID: 03F6
CONNECTOR NAME: CSQEMQVGMQV104 CFNAME: CF06
```

Figure 15-14   Connection to Coupling Facility CF06

The MVS command to display coupling facility structure information D XCF,STR,STRNM=MQVGAPPLICATION1 shows us that in the preference list we have CF05 and CF06 defined and DUPLEX is disabled.

```
D XCF,STR,STRNM=MQVGAPPLICATION1
IXC360I  16.23.42  DISPLAY XCF 443
STRNAME: MQVGAPPLICATION1
 STATUS: ALLOCATED
  POLICY SIZE    : 81920 K
  POLICY INITSIZE: 20480 K
  POLICY MINSIZE : 0 K
  FULLTHRESHOLD  : 85
  ALLOWAUTOALT   : NO
  REBUILD PERCENT: 5
  DUPLEX         : DISABLED
  PREFERENCE LIST: CF06     CF05
  ENFORCEORDER   : NO
  EXCLUSION LIST IS EMPTY

 ACTIVE STRUCTURE
 ----------------
  ALLOCATION TIME: 08/22/2002 11:54:44
  CFNAME         : CF06
  COUPLING FACILITY: 009672.IBM.02.000000050822
                     PARTITION: F   CPCID: 00
  ACTUAL SIZE    : 20480 K
  STORAGE INCREMENT SIZE: 256 K
   PHYSICAL VERSION: B822FB0C A8358FC0
   LOGICAL  VERSION: B822FB0C A8358FC0
   SYSTEM-MANAGED PROCESS LEVEL: 9
   XCF GRPNAME    : IXCLO098
   DISPOSITION    : KEEP
   ACCESS TIME    : NOLIMIT
   MAX CONNECTIONS: 32
   # CONNECTIONS  : 4

  CONNECTION NAME  ID VERSION  SYSNAME  JOBNAME  ASID STATE
  ---------------- -- -------- -------- -------- ---- ----------------
  CSQEMQVGMQV101   01 00010055 SC53     MQV1MSTR 03EB ACTIVE
  CSQEMQVGMQV202   02 00020040 SC61     MQV2MSTR 03EB ACTIVE
  CSQEMQVGMQV303   04 00040021 SC66     MQV3MSTR 03FD ACTIVE
  CSQEMQVGMQV404   03 00030023 SC62     MQV4MSTR 03F6 ACTIVE
```

Figure 15-15 Output of D XCF,STR,STRNM=MQVGAPPLICATION1 command

When we deactivated the coupling facility, all queue managers from our queue-sharing group abended, as we would expect, with abend code 6C6, as shown in Figure 15-16 on page 321.

```
 CSQE007I -MQV1 EEPLLOSSCONN event received for
 structure APPLICATION1 connection name CSQEMQVGMQV101
 CSQE007I -MQV1 EEPLLOSSCONN event received for
 structure CSQ_ADMIN connection name CSQEMQVGMQV101
 CSQE006I -MQV1 Structure CSQ_ADMIN connection name
 CSQEMQVGMQV101 disconnected
 CSQE016E -MQV1 Structure CSQ_ADMIN connection name
 CSQEMQVGMQV101 disconnected, RC=00000008 reason=0204080A
*CSQV086E -MQV1    QUEUE MANAGER ABNORMAL TERMINATION REASON=00C510AB
 IEF450I MQV1MSTR MQV1MSTR - ABEND=S6C6 U0000 REASON=00000000
         TIME=11.54.39
```

Figure 15-16 Abend 6C6 after coupling facility outage

The message CSQE016E delivers a return and reason code. The explanation for the return and reason codes can be found in the *z/OS: MVS Programming Sysplex Services Reference:* SA22-7618-02.

In our case we have a return code 00000008 and reason code 0204080A, which means there was a program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons:

1. The user with the connection identifier represented by the token has disconnected from the structure.
2. The connector's task (the task that issued IXLCONN) ended.
3. The specified token is not the token that was returned from IXLCONN.
4. The request was issued from an address space other than the address space in which IXLCONN was issued.
5. The connect token was invalidated during rebuild.
6. The connect token was invalidated by XES.

Because we deactivated our coupling facility LPAR to simulate a CF outage, reason number 6 is true, because in our case the connect token was invalidated by XES and the queue-sharing group could not continue to use the structure. The queue managers abend with 6c6 and they will have to do recovery and cleanup after restart as specified in action number 6.

Action number 6 tells the application (in this case the queue manager) to discontinue use of the structure, and perform recovery and cleanup for the structure.

Because all four queue managers are registered to ARM, the restart request was submitted within seconds. Because CF06 is not active any more, this time CF05 is taken to allocate the structures.

The messages shown in Figure 15-17 on page 323 are written to the queue manager's joblog. Message CSQE031I indicates that some functions are not yet available because the indicated queue manager has not completed building its data for the administration structure yet. Message CSQE018I tells us that queue manager MQV1 starts to build its own data for the administration structure, and message CSQE019I indicates that MQV1 has finished building its own data for the administration structure.

The important message is CSQE035E because this message tells us that we have to initiate manual recovery. The explanation for this message is that the queue manager attempted to use structure *struc-name*, but it is in a failed state. The failure occurred previously; it was not caused by the current use of the structure. Processing can continue, but queues that use this structure will not be accessible.

```
CSQE005I -MQV1 Structure CSQ_ADMIN connected as
CSQEMQVGMQV101, version=B822FAFBC5798EC3 0004001D
IXL014I IXLCONN REQUEST FOR STRUCTURE MQVGCSQ_ADMIN
WAS SUCCESSFUL.  JOBNAME: MQV1MSTR ASID: 03EB
CONNECTOR NAME: CSQEMQVGMQV101 CFNAME: CF05
CSQE031I -MQV1 Admin structure data from MQV2 incomplete
CSQE031I -MQV1 Admin structure data from MQV3 incomplete
CSQE031I -MQV1 Admin structure data from MQV4 incomplete
CSQE018I -MQV1 Admin structure data building started
CSQE019I -MQV1 Admin structure data building completed
CSQR030I -MQV1 Forward recovery log range
from RBA=0000291C6000 to RBA=00002920A3EE
CSQR005I -MQV1 RESTART - FORWARD RECOVERY COMPLETE -
IN COMMIT=0, INDOUBT=0
CSQR032I -MQV1 Backward recovery log range
from RBA=00002920A3EE to RBA=00002920A3EE
CSQR006I -MQV1 RESTART - BACKWARD RECOVERY COMPLETE -
INFLIGHT=0, IN BACKOUT=0
CSQR002I -MQV1 RESTART COMPLETED
CSQP018I -MQV1 CSQPBCKW CHECKPOINT STARTED FOR ALL BUFFER POOLS
CSQP019I -MQV1 CSQP1DWP CHECKPOINT COMPLETED FOR
-MQV1 DISPLAY THREAD(*) TYPE(INDOUBT)
BUFFER POOL 2, 2 PAGES WRITTEN
CSQP019I -MQV1 CSQP1DWP CHECKPOINT COMPLETED FOR
BUFFER POOL 1, 2 PAGES WRITTEN
.
.
.
 CSQE035E -MQV1 CSQECONN Structure APPLICATION1 in
 failed state, recovery needed
```

Figure 15-17 Queue manager restart

To fix the problem reported with message CSQE035E, the WebSphere MQ command RECOVER CFSTRUCT(APPLICATION1) must be issued. If the command execution is successful, it results in the messages shown in Figure 15-18 on page 324.

```
-MQV4 RECOVER CFSTRUCT(APPLICATION1)
CSQE005I -MQV4 Structure APPLICATION1 connected as
CSQEMQVGMQV404, version=B822FB0CA8358FC0 00010054
IXL014I IXLCONN REQUEST FOR STRUCTURE MQVGAPPLICATION1
WAS SUCCESSFUL.  JOBNAME: MQV4MSTR ASID: 03F6
CONNECTOR NAME: CSQEMQVGMQV404 CFNAME: CF05
CSQ9022I -MQV4 CSQERCFS ' RECOVER CFSTRUCT' NORMAL COMPLETION
CSQE132I -MQV4 Structure recovery started, using log
range from LRSN=B822F66D9821B905 to LRSN=B822FB1D0B35D242
CSQE134I -MQV4 Structure recovery reading log completed
CSQE130I -MQV4 Recovery of structure APPLICATION1
started, using MQV2 log range from RBA=000028A95E23 to
RBA=000028A96218
CSQE131I -MQV4 CSQERCF2 Recovery of structure APPLICATION1 completed
CSQE006I -MQV4 Structure APPLICATION1 connection name
CSQEMQVGMQV404 disconnected
```

Figure 15-18 Recover CFSTRUCT

At our first attempt to recreate the coupling facility failure, the structure recovery lasted a long time because our coupling facility backup was older than three weeks and we forgot to issue the WebSphere MQ BACKUP CFSTRUCT command before deactivating the coupling facility. The time it takes to recover a coupling facility structure depends on the amount of recovery log data that must be replayed, which in turn depends on the frequency of the backups. *WebSphere MQ for z/OS Concepts and Planning Guide,* GC34-6051, recommends backing up the coupling facility structures at least every hour. To learn more about recovering coupling facility structures, refer to Chapter 18, “Planning for backup and recovery”, in the *WebSphere MQ for z/OS Concepts and Planning Guide,* GC34-6051.

15.9 Conclusions

As a result of running these tests, we proved that our queue-sharing group reduces the time of outages of planned as well as unexpected failures of various components.

In a majority of cases, our configuration allows automatic recovery without manual intervention. There are still issues with DB2 failures and coupling facility recovery, which need special attention:

- To prevent queue manager failure after a coupling facility outage, you should consider using the new coupling facility duplexing function.

- To prevent problems with DB2 failures, consider using DB2 group attachement instead of a specific DB2 subsystem ID. This allows you to use a second DB2 subsystem on the same LPAR as a backup.

Since in many recovery scenarios channel connections have to be recovered, we recommend that you review your setup of retry intervals and retry attempts.

A

The front-end CICS program MQCICEJB

In this appendix, we provide the source code for the front-end CICS application program MQCICSEJB that is associated with the transaction code ECHO. For the full description of the application logic flow, refer to Chapter 12, “CICS applications and queue sharing” on page 225. For information about how to download the Java components of the sample aplication, refer to Appendix B, “Additional material” on page 339.

Example: A-1 WebSphere MQ CICS program MQCICEJB

```
//*                                                            /*JCTRL*/
//         EXEC PROC=CTSPKTAL,OUTC=K,
//         INDEX='CICSTS22.CICS'
//TRN.STEPLIB  DD
//SYSPRINT DD SYSOUT=*
//TRN.SYSIN DD  *
*
DFHEISTG DSECT
HEXTAB   DS  CL256
*
PFXDATA  DS    OCL74
PFX      DS    CL10
BUFFER   DS    CL64
BUFFER_LEN EQU *-BUFFER
CA       DS     OCL35
```

```
CASTR    DS     CL35
*
MSGOPEN  DS    0CL80
MSTR1    DS    CL20
MSTR2    DS    F
MSTR3    DS    F
MOO_MSG  DS     CL79                         Current output message
MOO_MSG4 DS     0CL79                        General error message
MOO_4_OP DS     CL08                         Operation identifier
         DS     CL25
MOO_4_CC DS     CL08                         Completion Code
         DS     CL09
MOO_4_RC DS     CL08                         Reason
         DS     CL21
*
*********************************************************************
OBJDESC  CMQODA LIST=YES                Working object descriptor
WMD      CMQMDA DSECT=NO,LIST=YES       Message Descriptor Structure
WGMO     CMQGMOA DSECT=NO,LIST=YES      Get Message Options Structure
WPMO     CMQPMOA DSECT=NO,LIST=YES      Put Message Options Structure
MQTM     CMQTMA DSECT=NO,LIST=YES       Trigger Message Structure
*********************************************************************
*
SELECTORCOUNT   DS F                         Number of selectors
INTATTRCOUNT    DS F                         Number of int attributes
CHARATTRLENGTH  DS F                         char attributes length
CHARATTRS       DS C                         Area for char attributes
*
REG11    DS   F
OPTIONS  DS   F                              Command options
HCONN    DS   F                              Handle of connection
HOBJ     DS   F                              Handle of object
COMPCODE DS   F                              Completion code
REASON   DS   F                              Reason code
DATALEN  DS   F
BUFFLEN  DS   F
SELECTOR DS   2F                             Array of selectors
INTATTRS DS   2F                             Array of int attributes
*
ACTION   DS   CL08                           Action on queue
OBJECT   DS   CL48                           Name of queue
DWORD    DS   D                              Work field
UPDTYPE  DS   CL11                           Data store
*
CALLLIST CALL ,(0,0,0,0,0,0,0,0,0,0,0),VL,MF=L
*
********************************************************************
*                   PROGRAM EXECUTION STARTS HERE                  *
********************************************************************
```

```
         EJECT
*******************************************************************
*  Code start                                                     *
*******************************************************************
*                                                                 *
*  Macro DFHEIENT is used to obtain working storage.              *
*  In our example a single register for code (R3) and             *
*   storage (R13) is sufficient                                   *
*                                                                 *
*******************************************************************
MQCICEJB DFHEIENT CODEREG=(R3),DATAREG=(R13)
MQCICEJB AMODE 31
MQCICEJB RMODE ANY
         B     MAIN
         DC    C'ITSO WebSphere MQ SAMPLE PROGRAM  '
         DC    C'NAME : '
         DC    C'MQCICEJB'
         DC    C' DATE AND TIME ASSEMBLED : '
         DC    C'&SYSDATC',C','
         DC    C'&SYSTIME '
         DC    C'&SYSPARM '
*
         SPACE 4
*******************************************************************
*  SECTION NAME : MAIN                                            *
*                                                                 *
*  FUNCTION     : Controls flow of program                        *
*                                                                 *
*  CALLED BY    : MQCICEJB CSECT                                  *
*                                                                 *
*  CALLS        : INITIAL, PROCESS, ENDPROG                       *
*                                                                 *
*******************************************************************
MAIN     DS    0H
         BAS   R5,INITIAL
         BAS   R5,PROCESS
         BAS   R5,ENDPROG
*
PROGEND  DS    0H
         EXEC CICS RETURN
*
         EJECT
*******************************************************************
*  SECTION NAME : INITIAL                                         *
*                                                                 *
*  FUNCTION     : Performs initialization                         *
*                                                                 *
*  CALLED BY    : MAIN                                            *
*                                                                 *
```

```
*  CALLS        : NONE                                              *
*                                                                   *
*  RETURN       : To Register 5                                     *
*                                                                   *
*********************************************************************
INITIAL  DS    OH
         XC   HEXTAB,HEXTAB        make sure HEXTAB is zero
         MVC  HEXTAB+64(1),=X'40'  say offset x'40' is special
         MVC  BUFFER,=CL64' '      set buffer to blanks
         MVC  PFX(10),=CL10'FRED SAID:' initialize prefix of buffer
*                                                                   *
*        Connection handle                                          *
*                                                                   *
         LA   RO,MQHC_DEF_HCONN             Set default value
         ST   RO,HCONN                      Place in field
*                                                                   *
******************************************************************* *
*  uncomment DELETEQ commands to trace debug information          * *
******************************************************************* *
         EXEC CICS HANDLE CONDITION LENGERR(CONTIN) QIDERR(QERR)
*        EXEC CICS DELETEQ TS QUEUE('ERROR') NOHANDLE
*        EXEC CICS DELETEQ TS QUEUE('OBJD') NOHANDLE
*        EXEC CICS DELETEQ TS QUEUE('RTOQ') NOHANDLE
*        EXEC CICS DELETEQ TS QUEUE('MSGS') NOHANDLE
*        EXEC CICS DELETEQ TS QUEUE('MQTM') NOHANDLE
*        EXEC CICS DELETEQ TS QUEUE('FREDS') NOHANDLE
*        EXEC CICS DELETEQ TS QUEUE('M') NOHANDLE
*
QERR     DS    OH
*                                                                   *
         MVI   MOO_MSG,X'40'              Move in first character
         MVC   MOO_MSG+1(L'MOO_MSG-1),MOO_MSG  and initialize
         MVC   MOO_MSG4,MO1_MSG4
*                                                                   *
*        OBJECT DESCRIPTOR FIELDS                                   *
*                                                                   *
 MVC   OBJDESC_AREA(256),CONST_MQOD_AREA
 MVC   OBJDESC_AREA+256(L'CONST_MQOD_AREA-256),CONST_MQOD_AREA+256
*                                           Initialize from constant
         MVC   CASTR(35),=CL35' '     initialize commarea and...
         MVC   OBJECT(48),=CL48' '     object area
*
         BR    R5                         Return to MAIN process
*
         EJECT
*
PROCESS  DS   OH
CMDGET   DS   OH
*******************************************************************
```

```
***     retrieve trigger message structure                        ***
*******************************************************************
         EXEC CICS RETRIEVE INTO(MQTM) LENGTH(MQTMLEN)
CONTIN   DS   0H
*       remove comment to trace trigger message structure
*        EXEC CICS WRITEQ TS QUEUE('MQTM') FROM(MQTM) LENGTH(MQTMLEN)
         MVC  OBJECT,MQTM_QNAME   get queue name from structure...
         BAS  R6,GETMSG            and get the message
         CLI  BUFFER,C'N'  if first character of message buffer...
         BE   NOBEAN        ... is 'N', don't use enterprise bean
         MVC  PFX,=CL10'BEAN SAID:' if bean is used, update prefix
         MVC  CASTR(35),BUFFER      and move buffer to commarea
*******************************************************************
***     lINK to JCICS client to access the bean                   ***
*******************************************************************
         EXEC CICS LINK PROGRAM('EJBCIC') COMMAREA(CA)
*
         MVC  BUFFER(35),CASTR  move commarea back to buffer...
CMDPUT   DS   0H
         BAS  R6,PUTMSG      ... and reply the message
         BR   R5                      back to main flow
*
NOBEAN   DS   0H
*        uncomment writeq below to trace buffer
*        EXEC CICS WRITEQ TS QUEUE('FREDS') FROM(PFXDATA)
         B    CMDPUT
*
*
*********************************************************************
***       reply message to reply_to queue                         ***
*********************************************************************
PUTMSG   DS   0H
*********************************************************************
***      reinitialize object descriptor and get replyto queue.   ***
***      if replyto queue name ends with 'LONG',                 ***
***      we want to extent the replytoqmgr name by '.alias' ***
*********************************************************************
         MVC   OBJDESC_AREA(256),CONST_MQOD_AREA
  MVC   OBJDESC_AREA+256(L'CONST_MQOD_AREA-256),CONST_MQOD_AREA+256
         MVC  OBJECT(48),WMD_REPLYTOQ
         CLC  OBJECT+25(4),=CL4'LONG'  replytiq = ......LONG?
         BNE  NOEXC                  ...no, continue
ADDALIAS DS   0H
         LA   1,WMD_REPLYTOQMGR      ...yes, append .alias
         TRT  0(48,1),HEXTAB           to replyqmgr
         MVC  0(6,1),=C'.alias'
NOEXC    DS   0H
         MVC  OBJDESC_OBJECTQMGRNAME(48),WMD_REPLYTOQMGR
CONTINUE DS   0H
```

```
*
         LA   R0,MQOT_Q                     Set object type
         ST   R0,OBJDESC_OBJECTTYPE         to queue
         MVC  OBJDESC_OBJECTNAME,OBJECT     Move in queue name
*
*        uncomment commands below to trace OBJD and RTOQ
*
*        EXEC CICS WRITEQ TS QUEUE('OBJD') FROM(OBJDESC) LENGTH(LO)
*        EXEC CICS WRITEQ TS QUEUE('RTOQ') FROM(OBJECT) LENGTH(LL)
*
*
         LA   R0,MQOO_OUTPUT                set open for output
         ST   R0,OPTIONS                    and..
*
         BAS  R7,OPEN                     ..Open the queue
*
         LA   R0,MQCC_OK                    Load expected compcode
         C    R0,COMPCODE                   Was open successful
         BNER R6                            No ... return to caller
*
MQPUT    DS   0H
***************************************************************
*** initialize message descriptor and do put               ***
***************************************************************
         ST   R11,REG11
         LA   R8,MQMD
         LA   R9,MQMD_LENGTH
         LA   R10,WMD
         LA   R11,WMD_LENGTH
         MVCL R10,R8
         MVC  WMD_FORMAT,MQFMT_STRING
         MVC  WPMO_AREA,MQPMO_AREA
         LA   R9,BUFFER_LEN
         ST   R9,BUFFLEN
         L    R11,REG11
***************************************************************
***************************************************************
         CALL MQPUT,                                                  C
               (HCONN,                                                C
               HOBJ,                                                  C
               WMD,                                                   C
               WPMO,                                                  C
               BUFFLEN,                                               C
               PFXDATA,                                               C
               COMPCODE,                                              C
               REASON),                                               C
               VL,MF=(E,CALLLIST)
*
         LA   R0,MQCC_OK                    Load expected compcode
```

```
         C    R0,COMPCODE                 Was put successful?
         BNE  ERRPUT                      no...assemble message
*
*        uncomment command below to trace message
*
*        EXEC CICS WRITEQ TS QUEUE('MSGS') FROM(BUFFER)
         B    ENDPUT
*
ERRPUT   DS   0H
         MVC  MSTR1(20),=C'PUT ERROR'
         MVC  MSTR2,COMPCODE
         MVC  MSTR3,REASON
*
*        uncomment the command below to trace message
*
*        EXEC CICS WRITEQ TS QUEUE('ERROR') FROM(MSGOPEN) LENGTH(L)
ENDPUT   DS   0H
         BAS  R7,CLOSE                    Close the queue
         BR   R6                          Return to caller
*
         EJECT
*********************************************************************
*** get a message from the given queue name                      ***
*********************************************************************
GETMSG   DS   0H
         LA   R0,MQOT_Q                   Set object type
         ST   R0,OBJDESC_OBJECTTYPE       to queue
         MVC  OBJDESC_OBJECTNAME,OBJECT   Move in queue name
*
*        uncomment the command below to trace the objd
*        EXEC CICS WRITEQ TS QUEUE('OBJD') FROM(OBJDESC) LENGTH(LO)
*
*
         LA   R0,MQOO_INPUT_AS_Q_DEF     set input options
         ST   R0,OPTIONS                 ..and
*
         BAS  R7,OPEN                    ..Open the queue
*
         LA   R0,MQCC_OK                  Load expected compcode
         C    R0,COMPCODE                 Was open successful
         BNER R6                          No ... return to caller
*
MQGET    DS   0H
****************************************************************
*** initialize message descriptor and do get                ***
****************************************************************
         ST   R11,REG11
         LA   R8,MQMD
         LA   R9,MQMD_LENGTH
```

```
         LA    R10,WMD
         LA    R11,WMD_LENGTH
         MVCL  R10,R8
         MVC   WGMO_AREA,MQGMO_AREA
         L     R9,=AL4(MQGMO_WAIT)
         ST    R9,WGMO_OPTIONS
         MVC   WGMO_WAITINTERVAL,TWOSECS
         LA    R9,BUFFER_LEN
         ST    R9,BUFFLEN
         L     R11,REG11
         CALL  MQGET,                                                  C
               (HCONN,                                                 C
               HOBJ,                                                   C
               WMD,                                                    C
               WGMO,                                                   C
               BUFFLEN,                                                C
               BUFFER,                                                 C
               DATALEN,                                                C
               COMPCODE,                                               C
               REASON),                                                C
               VL,MF=(E,CALLLIST)
*
         LA    R0,MQCC_OK                   Load expected compcode
         C     R0,COMPCODE                  Was get successful?
         BNE   PREND
*
*        uncomment command below to trace replytoq
*        EXEC CICS WRITEQ TS QUEUE('RTOQ') FROM(WMD_REPLYTOQ)          X
                 LENGTH(LEN)
*
ENDGET   DS    0H
         BAS   R7,CLOSE                     Close the queue
         BR    R6                           RETURN TO CALLER
*
PREND    DS    0H
         B     PROGEND
*********************************************************************
* SUBROUTINES                                                       *
*********************************************************************
*                                                                   *
*********************************************************************
*  SECTION NAME : OPEN                                              *
*                                                                   *
*  FUNCTION     : To open the object                                *
*                                                                   *
*  CALLED BY    : GET   , PUT                                       *
*                                                                   *
*  CALLS        : CODES                                             *
*                                                                   *
```

```
*  RETURN       : To Register 7                                      *
*                                                                    *
**********************************************************************
OPEN     DS   0H                            Open the object
         CALL MQOPEN,                                                    C
               (HCONN,                                                   C
               OBJDESC,                                                  C
               OPTIONS,                                                  C
               HOBJ,                                                     C
               COMPCODE,                                                 C
               REASON),                                                  C
               MF=(E,CALLLIST),VL
*
         LA   R0,MQCC_OK                    Load expected code
         C    R0,COMPCODE                   Code as expected?
         BE   ENDOPEN                       Yes ... branch to exit
         ST   R0,MSTR2
         MVC  MSTR1(20),=C'OPEN ERROR'
*
*        uncomment command below to trace errors
*        EXEC CICS WRITEQ TS QUEUE('ERROR') FROM(MSGOPEN)
*
OPENEVAL DS   0H                            Evaluate error
         L    R0,=AL4(MQRC_Q_MGR_NOT_AVAILABLE) Load
         C    R0,REASON                     Special error?
         BE   OPENCONN                      Yes .. display MSG6
         L    R0,=AL4(MQRC_CONNECTION_BROKEN) Load
         C    R0,REASON                     Special error?
         BE   OPENCONN                      Yes .. display MSG6
         L    R0,=AL4(MQRC_UNKNOWN_OBJECT_NAME) Load
         C    R0,REASON                     Special error?
         BE   OPENUNKN                      Yes .. display MSG2
         L    R0,=AL4(MQRC_NOT_AUTHORIZED)  Load
         C    R0,REASON                     Special error?
         BE   OPENAUTH                      Yes .. display MSG3
         B    OPENGENL                      General error
*
OPENCONN DS   0H
         MVC  M00_MSG,M01_MSG6              Move in connection err msg
         B    ENDOPEN                       Branch to section exit
*
OPENUNKN DS   0H
         MVC  M00_MSG,M01_MSG2              Move in unknown q error msg
         B    ENDOPEN                       Branch to section exit
*
OPENAUTH DS   0H
         MVC  M00_MSG,M01_MSG3              Move in authorization error
         B    ENDOPEN                       Branch to section exit
*
```

```
OPENGENL DS   0H
         MVC  MOO_4_OP,OP_OPEN              Error ... construct message
         BAS  R4,CODES                      Translate codes
         MVC  MOO_MSG,MOO_MSG4              Move in general error msg
         B    ENDOPEN                       Branch to section exit
*
ENDOPEN  DS   0H
*
*        uncomment command below to trace errors
*        EXEC CICS WRITEQ TS QUEUE('ERROR') FROM(MOO_MSG) LENGTH(L)
         BR   R7                            Return to caller
*
         EJECT
*********************************************************************
*  SECTION NAME : CLOSE                                             *
*                                                                   *
*  FUNCTION     : To close the object                               *
*                                                                   *
*  CALLED BY    : UPDATE, INQUIRE                                   *
*                                                                   *
*  CALLS        : CODES                                             *
*                                                                   *
*  RETURN       : To Register 7                                     *
*                                                                   *
*********************************************************************
CLOSE    DS   0H                            Close the object
         LA   R0,MQCO_NONE                  Normal close
         ST   R0,OPTIONS                    options here
*
         CALL MQCLOSE,                                                 C
               (HCONN,                                                 C
               HOBJ,                                                   C
               OPTIONS,                                                C
               COMPCODE,                                               C
               REASON),                                                C
               MF=(E,CALLLIST),VL
*
         LA   R0,MQCC_OK                    Load expected code
         C    R0,COMPCODE                   Code as expected?
         BE   ENDCLOSE                      Yes ... branch to exit
*
         MVC  MOO_4_OP,OP_CLOSE             Error ... construct message
         BAS  R4,CODES                      Translate codes
         MVC  MOO_MSG,MOO_MSG4              Move in message
         B    ENDCLOSE                      Branch to section exit
*
ENDCLOSE DS   0H
         BR   R7                            Return to caller
*
```

```
         EJECT
*******************************************************************
*  SECTION NAME : CODES                                           *
*                                                                 *
*  FUNCTION     : Translates COMPCODE and REASON to a format      *
*                 for displaying                                  *
*                                                                 *
*  CALLED BY    : UPDATE, INQUIRE, OPEN, CLOSE                    *
*                                                                 *
*  CALLS        : None                                            *
*                                                                 *
*  RETURN       : To Register 4                                   *
*                                                                 *
*******************************************************************
CODES    DS   0H
         L    R0,COMPCODE                  Load compcode value
         CVD  R0,DWORD                     Binary to packed decimal
         UNPK M00_4_CC,DWORD+4(4)          Packed to zoned decimal
         MVZ  M00_4_CC+7(1),M00_4_CC+6     Make it display
*
         L    R0,REASON                    Load reason value
         CVD  R0,DWORD                     Binary to packed decimal
         UNPK M00_4_RC,DWORD+4(4)          Packed to zoned decimal
         MVZ  M00_4_RC+7(1),M00_4_RC+6     Make it display
         BR   R4                           Return to caller
*
ENDPROG  DS   0H
         MVI  M00_MSG,X'40'               Move in first character
         MVC  M00_MSG+1(L'M00_MSG-1),M00_MSG  clear the message field
         MVC  M00_MSG,M01_MSG7            Set termination message
*
*        uncomment command below to trace messages
*
*        EXEC CICS WRITEQ TS QUEUE('M') FROM(M00_MSG)
*
         BR    R5                         Return to caller
*
*******************************************************************
***     CONSTANTS                                                 *
*******************************************************************
         COPY  DFHAID                      PF keys etc
         COPY  DFHBMSCA                    BMS MAP command
*
         DFHREGS                           Register equates
*
         CMQA  LIST=NO                     Equates for MQ constants
         CMQGMOA LIST=NO
         CMQPMOA LIST=NO
MDAINIT  DS    0H
```

```
         CMQMDA  LIST=YES
*
         COPY  CSQ4AMSG                    Get the messages
*
CONST_MQOD     CMQODA LIST=NO
*
QPREFIX  DC    CL8'CSQ4SAMP'               Queue start restriction
SENDLEN  DC    Y(L'MOO_MSG)                Length of message
*
OP_SET   DC    CL08'MQSET'                 MQI Call
OP_INQ   DC    CL08'MQINQ'                 MQI Call
OP_OPEN  DC    CL08'MQOPEN'                MQI Call
OP_CLOSE DC    CL08'MQCLOSE'               MQI Call
*
CALLOW   DC    CL08'ALLOW'                 Action verb
CINQUI   DC    CL08'INQUIRE'               Action verb
CINHIB   DC    CL08'INHIBIT'               Action verb
CGETMSG  DC    CL08'GETMSG'                ACTION VERB
CPUTMSG  DC    CL08'PUTMSG'                ACTION VERB
*
CALLOWED DC    CL11'ALLOWED '              Queue status
CINHIBIT DC    CL11'INHIBITED '            Queue status
*
MAPSETID DC    CL08'CSQ4ACM '              Name of the mapset
MAPID    DC    CL08'CSQ4AC1 '              Name of the map
MAPIDHLP DC    CL08'CSQ4AC2 '              Name of the help map
*
*
MQTMLEN  DC    H'200'
LEN      DC    H'96'
L        DC    H'79'
LO       DC    H'288'
LL       DC    H'48'
TWOSECS  DC    F'2000'
MQFMT_STRING             DC C'MQSTR   '
*
*
         LTORG
*
         END   MQCICEJB
//LKED.SYSIN  DD  *
    MODE RMODE(ANY),AMODE(31)
    INCLUDE   SYSLIB(CSQCSTUB)
              NAME  MQCICEJB(R)
/*
//
```

B

Additional material

This redbook refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

`ftp://www.redbooks.ibm.com/redbooks/SG246864`

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the redbook form number, SG246864.

Using the Web material

The additional Web material that accompanies this redbook is packaged in one file, called SG246864.zip, and includes the following folders:

File name	*Description*
MQ_Application	CICS application program used by ECHO transaction and MQ object definitions and script
JCICS_Client	The JCICS EJB client program and a CICS 3270 program used to debug the client

System requirements for downloading the Web material

The following system configuration is recommended:

Hard disk space:	1 MB minimum
Operating System:	Windows NT or Windows 2000
Processor:	Any
Memory:	Any

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder. Then follow the instructions in the readme.txt file.

Glossary

Access Control List (ACL). The list of principals that have explicit permission (to publish, to subscribe to, and to request persistent delivery of a publication message) against a topic in the topic tree. The ACLs define the implementation of topic-based security.

Adapter. An interface between WebSphere MQ for z/OS and TSO, IMS, CICS, or batch address spaces. An adapter is an attachment facility that enables applications to access WebSphere MQ services.

Administrator commands. WebSphere MQ commands used to manage WebSphere MQ objects, such as queues, processes, and namelists.

AMI. See **Application messaging interface**.

API. See **Application programming interface**.

Application messaging interface. The programming interface provided by MQSeries that defines a high-level interface to message queuing services.

Application programming interface. An interface provided by a software product that enables programs to request services.

Asynchronous messaging. A method of communication between programs in which programs place messages on message queues. With asynchronous messaging, the sending program proceeds with its own processing without waiting for a reply to its message.

BLOB. Binary Large Object, a block of bytes of data (for example, the body of a message) that has no discernible meaning, but is treated as one solid entity that cannot be interpreted.

Bootstrap data set (BSDS). A VSAM data set that contains an inventory of all active and archived log data sets known to WebSphere MQ for z/OS. It is a wrap-around inventory of all recent WebSphere MQ for z/OS activity. The BSDS is required for WebSphere MQ restart.

For DB2: A VSAM data set that contains name and status information for DB2, as well as RBA range specifications, for all active and archive log data sets. It also contains lists of conditional restart and checkpoint records.

Broker. See **Message broker**.

Browse cursor. In message queuing, an indicator used when browsing a queue to identify the message that is next in sequence.

Browse. In message queuing, to use the MQGET call to copy a message without removing it from the queue.

Buffer pool. An area of main storage used for WebSphere MQ for z/OS queues, messages, and object definitions.

For DB2: Main storage that is reserved to satisfy the buffering requirements for one or more tablespaces or indexes.

Channel control function (CCF). In WebSphere MQ, a program to move messages from a transmission queue to a communication link, and from a communication link to a local queue, together with an operator screen interface to allow the setup and control of channels.

Channel definition file (CDF). In WebSphere MQ, a file containing communication channel definitions that associate transmission queues with communication links.

Channel event. An event indicating that a channel instance has become available or unavailable. Channel events are generated on the queue managers at both ends of the channel.

Check in. The Control Center action that stores a new or updated resource in the configuration or message repository.

Check out. The Control Center action that extracts and locks a resource from the configuration or message repository for local modification by a user.

Client connection channel type. A type of MQI channel definition associated with a WebSphere MQ client.

Collective. A totally connected set of brokers forming part of a multi-broker network for publish/subscribe applications.

Configuration Manager. A component of WebSphere MQ Integrator that acts as the interface between the configuration repository and an existing set of brokers.

Configuration repository. Persistent storage for broker configuration and topology definition.

Connection handle. The identifier or token by which a program accesses the queue manager to which it is connected.

Constraint. A rule that limits the values that can be inserted, deleted, or updated in a table. See **referential constraint.**

Context security. In WebSphere MQ, a method of allowing security to be handled such that messages are obliged to carry details of their origins in the message descriptor.

Context. Information about the origin of a message.

Control Center. The graphical interface that provides facilities for defining, configuring, deploying, and monitoring resources of the WebSphere MQ Integrator network.

Datagram. The simplest message that WebSphere MQ supports. This type of message does not require a reply.

Dead-letter-queue (DLQ). A queue to which a queue manager or application sends messages that it cannot deliver to their correct destination.

Debugger. A facility on the Message Flows view in the Control Center that enables message flows to be visually debugged.

Deferred connection. A pending event that is activated when a CICS subsystem tries to connect to WebSphere MQ for z/OS before WebSphere MQ for z/OS has started.

Deploy. Make operational the configuration and topology of the broker domain.

Distribution list. A list of MQSeries queues to which a message can be put using a single statement.

Document type definition. The rules that specify the structure for a particular class of XML documents. The DTD defines the structure with elements, attributes, and notations, and it establishes constraints for how each element, attribute, and notation can be used within the particular class of documents.

Domain. See **Message domain**.

DTD. See **Document type definition**.

Execution group. A named grouping of message flows that have been assigned to a broker.

Extended SQL. A specialized set of SQL functions and statements based on regular SQL, extended with functions and statements unique to WebSphere MQ Integrator.

Field reference. A sequence of period-separated values that identify a specific field within a message tree. An example of a field reference might be *Address.Country.City.Street.HouseNumber*.

Get. In message queuing, to use the MQGET call to remove a message from a queue.

Initiation queue. A local queue on which the queue manager puts trigger messages.

Input node. A message flow node that represents a source of messages for the message flow.

Java Database Connectivity. An application programming interface that has the same characteristics as ODBC but is specifically designed for use by Java database applications.

Java Development Kit. Software package used to write, compile, debug and run Java applets and applications.

Java Message Service. An application programming interface that provides Java language functions for handling messages.

Java Runtime Environment. A subset of the **Java Development Kit** that allows you to run Java applets and applications.

JDK. See **Java Development Kit**.

JMS. See **Java Message Service**.

JRE. See **Java Runtime Environment**.

Listener. In WebSphere MQ distributed queuing, the program that monitors for incoming network connections.

Local queue manager. The queue manager to which a program is connected and that provides message queuing services to the program.

Local queue. A queue that belongs to the local queue manager. A local queue can contain a list of messages waiting to be processed.

Log. In WebSphere MQ, a file recording the work done by queue managers while they receive, transmit, and deliver messages.

Message broker. A set of execution processes hosting one or more message flows.

Message channel agent (MCA). A program that transmits prepared messages from a transmission queue to a communication link, or from a communication link to a destination queue.

Message channel. In distributed message queuing, a mechanism for moving messages from one queue manager to another. A message channel comprises two message channel agents (a sender and a receiver) and a communication link.

Message descriptor. Control information describing the message format and presentation that is carried as part of a WebSphere MQ message. The format of the message descriptor is defined by the MQMD structure.

Message domain. The value that determines how the message is interpreted (parsed). The following domains are recognized:

- MRM, which identifies messages defined using the Control Center.
- NEONMSG and NEON, which identify messages created using the New Era of Networks user interfaces.
- XML, JMSMap, and JMSStream, which identify messages that are self-defining.
- BLOB, which identifies messages that are undefined.

Message flow. A directed graph that represents the set of activities performed on a message or event as it passes through a broker. A message flow consists of a set of message-processing nodes and message-processing connectors.

Message priority. In WebSphere MQ, an attribute of a message that can affect the order in which messages on a queue are retrieved, and whether a trigger event is generated.

Message-processing node connector. An entity that connects the output terminal of one message processing node to the input terminal of another.

Message-processing node. A node in the message flow, representing a well-defined processing stage. A message-processing node can be one of several primitive types or it can represent a subflow.

Message Queue Interface. The programming interface provided by MQSeries queue managers that enables application program access to message queuing services.

Message queuing. 1. A communication technique that uses asynchronous messages for communication between software components.

2. A programming technique in which each program within an application communicates with the other programs by putting messages on queues

Message Repository Manager (MRM). A component of the Configuration Manager that handles message definition and control. A message defined to the MRM has a message domain set to MRM.

Message repository. A database holding message template definitions.

Message set. A grouping of related messages.

MQRFH. An architected message header that is used to provide metadata for the processing of a message. This header is supported by MQSeries publish/subscribe.

MQRFH2. An extended version of MQRFH being used by WMQI applications.

Namelist. A WebSphere MQ for z/OS object that contains a list of queue names.

Nonpersistent message. A message that does not survive a restart of the queue manager.

Object. In WebSphere MQ, an object is a queue manager, a queue, a process definition, a channel, a namelist (z/OS only), or a storage class (z/OS only).

ODBC. See **Open Database Connectivity**.

Offloading. In WebSphere MQ for z/OS, an automatic process whereby a queue manager's log is transferred to its archive log.

Open Database Connectivity. A standard application programming interface for accessing data in both relational and non-relational database management systems. Using this API, database applications can access data stored in database management systems on a variety of computers even if each database management system uses a different data storage format and programming interface. ODBC is based on the call level interface (CLI) specification of the X/Open SQL Access Group.

Output node. A message processing node that represents a point at which messages flow out of the message flow.

Page set. A VSAM data set used when WebSphere MQ for z/OS moves data (for example, queues and messages) from buffers in main storage to permanent backing storage (DASD).

Persistent message. A message that survives a restart of the queue manager.

Plug-in. An extension to the broker, written by a third-party developer, to provide a new message processing node or message parser in addition to those supplied with the product.

Point-to-point. Style of messaging application in which the sending application knows the destination of the message.

Predefined message. A message with a structure that is defined before the message is created or referenced.

Publish/subscribe. Style of messaging application in which the providers of information (publishers) are decoupled from the consumers of that information (subscribers) through a broker.

Publisher. An application that makes information about a specific topic available to a broker in a publish/subscribe system.

Queue manager. A subsystem that provides queuing services to applications. It provides an application programming interface so that applications can access messages on the queues that are owned and managed by the queue manager.

Queue. A WebSphere MQ object. Applications can put messages on, and get messages from, a queue. A queue is owned and managed by a queue manager. A local queue is a type of queue that can contain a list of messages waiting to be processed. Other types of queues cannot contain messages but are used to point to other queues.

Receiver channel. A channel that responds to a sender channel, takes messages from a communication link, and puts them on a local queue.

Recovery log. In WebSphere MQ for z/OS, data sets containing information needed to recover messages, queues, and the WebSphere MQ subsystem. WebSphere MQ for z/OS writes each record to a data set called the active log. When the active log is full, its contents are offloaded to a DASD or tape data set called the archive log.

Referential constraint. The requirement that non-null values of a designated foreign key are valid only if they equal values of the primary key of a designated table.

Remote queue manager. To a program, a queue manager that is not the one to which the program is connected.

Remote queue. A queue belonging to a remote queue manager. Programs can put messages on remote queues, but they cannot get messages from remote queues.

Reply message. A type of message used for replies to request messages.

Reply-to queue. The name of a queue to which the program that issued an MQPUT call wants a reply message or report message to be sent.

Report message. A type of message that gives information about another message. A report message can indicate that a message has been delivered, has arrived at its destination, has expired, or could not be processed for some reason.

Request message. A type of message used to request a reply from another program.

Requester channel. In message queuing, a channel that can be started remotely by a sender channel. The requester channel accepts messages from the sender channel over a communication link and puts the messages on the local queue designated in the message.

Self-defining message. A message that defines its structure within its content. For example, a message coded in XML is self-defining.

Sender channel. In message queuing, a channel that initiates transfers, removes messages from a transmission queue, and moves them over a communication link to a receiver or requester channel.

Sequential delivery. In WebSphere MQ, a method of transmitting messages with a sequence number so that the receiving channel can reestablish the message sequence when storing the messages.

Server channel. In message queuing, a channel that responds to a requester channel, removes messages from a transmission queue, and moves them over a communication link to the requester channel.

Signaling. In WebSphere MQ for z/OS, a feature that allows the operating system to notify a program when an expected message arrives on a queue.

Storage class. In WebSphere MQ for z/OS, a storage class defines the page set that is to hold the messages for a particular queue. The storage class is specified when the queue is defined.

Subflow. A sequence of message processing nodes that can be included within a message flow.

Subscriber. An application that requests information about a specific topic from a publish/subscribe broker.

Synchronous messaging. A method of communication between programs in which the sending program waits for a reply to its message before resuming its own processing.

Terminal. The point at which one node in a message flow is connected to another node.

Transmission queue. A local queue on which prepared messages destined for a remote queue manager are temporarily stored.

Trigger event. An event (such as a message arriving on a queue) that causes a queue manager to create a trigger message on an initiation queue.

Trigger message. A message containing information about the program that a trigger monitor is to start.

Trigger monitor. A continuously running application serving one or more initiation queues. When a trigger message arrives on an initiation queue, the trigger monitor retrieves the message. It uses the information in the trigger message to start a process that serves the queue on which a trigger event occurred.

Triggering. In WebSphere MQ, a facility allowing a queue manager to start an application automatically when predetermined conditions on a queue are satisfied.

Unit of recovery. A recoverable sequence of operations within a single resource manager.

Unit of work. A recoverable sequence of operations performed by an application between two points of consistency.

User Name Server. The WMQI component that interfaces with operating system facilities to determine valid users and groups.

WebSphere MQ. A family of IBM-licensed programs that provides message queuing services.

Wire format. A description of the physical representation of a message within the bit stream.

XML. eXtensible Markup Language, a standard for the representation of data.

Abbreviations and acronyms

AMI	Application Messaging Interface
API	Application Programming Interface
ARM	Automatic Restart Manager
BLOB	Binary Large Object
CAF	Client Access Facility
CEC	Central Electronic Complex
CPC	Central Processor Complex
CICS	Customer Information Control System
CIF	Customization Input File
CWF	Custom Wire Format
DB2	Database 2
DB2 UDB	DB2 Universal Database
DLQ	Dead-Letter Queue
DNS	Domain Name System
DTD	Document Type Definition
DVIPA	Dynamic Virtual IP Address
EAI	Enterprise Application Integration
EDI	Electronic Data Interchange
EJB	Enterprise Java Bean
ESM	External Security Manager
ESQL	Extended Structured Query Language
EXCI	External CICS Interface
GDPS	Geographically Dispersed Parallel Sysplex
GID	Group Identifier
GUI	Graphical User Interface
HFS	Hierarchical File System
HTML	HyperText Markup Language
HTTP	HyperText Transport Protocol
IBM	International Business Machines Corporation
IDE	Integrated Development Environment
IIOP	Internet InterORB Protocol
IMS	Information Management System
ISPF	Interactive System Productivity Facility
ISV	Independent Software Vendor
ITSO	International Technical Support Organization
JAR	Java Archive
JCL	Job Control Language
JDBC	Java Database Connectivity
JDK	Java Development Kit
JMS	Java Message Service
JRE	Java Runtime Environment
LAN	Local Area Network
LE	Language Environment
MDAC	Microsoft Data Access Component
MQ	Message Queuing
MQI	Message Queuing Interface
MQSI	MQSeries Integrator
MQWF	MQSeries Workflow
MRM	Message Repository Manager
NEON	New Era of Networks
NFS	Network File System
ODBC	Open Database Connectivity
OS/390	Operating System/390
OSPF	Open Shortest Path First

OTMA	Open Transaction Manager Access
PDS	Partitioned Data Set
PDSE	Partitioned Data Set Extended
QSG	Queue-Sharing Group
RACF	Resource Access Control Facility
RIP	Routing Information Protocol
RRS	Resource Recovery Services
SIT	System Initialization Table
SMP/E	System Management Program/Extended
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SWIFT	Society For Worldwide Interbank Financial Telecommunications
TDWF	Tagged/Delimited Wire Format
TSO	Time Sharing Option
TSO/E	Time Sharing Option/Extended
TWF	Tagged Wire Format
UID	User Identifier
UNS	User Name Server
UOW	Unit of Work
UR	Unit of Recovery
URL	Uniform Resource Locator
VIPA	Virtual IP Address
VTAM	Virtual Telecommunications Access Method
WLM	WorkLoad Management
WMQI	WebSphere MQ Integrator
XCF	Extended Connection Facility
XML	eXtensible Markup Language

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see "How to get IBM Redbooks" on page 351.

- *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 4: Connectivity and Routing,* SG24-6516
- *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 5: Availability, Scalability, and Performance,* SG24-6517
- *TCP/IP in a Sysplex,* SG24-5235
- *WebSphere MQ Integrator for z/OS V2.1 Implementation Guide,* SG24-6528
- *Enterprise JavaBeans for z/OS and OS/390 CICS Transaction Server V2.2,* SG24-6284
- *DB2 UDB for OS/390 and Continuous Availability*, SG24-5486

Other resources

These publications are also relevant as further information sources:

- *WebSphere MQ for z/OS Concepts and Planning Guide V5.3,* GC34-6051
- *WebSphere MQ for z/OS System Setup Guide V5.3,* SC34-6052
- *WebSphere MQ for z/OS System Administration Guide V5.3,* SC34-6053
- *WebSphere MQ for z/OS Problem Determination Guide V5.3,* GC34-6054
- *WebSphere MQ for z/OS Messages and Codes V5.3,* GC34-6056
- *WebSphere MQ Intercommunication,* SC34-6059
- *WebSphere MQ Queue Manager Clusters,* SC34-6061
- *WebSphere MQ Script (MQSC) Command Reference,* SC34-6055
- *WebSphere MQ Application Programming Guide,* SC34-6064
- *WebSphere MQ Application Programming Reference,* SC34-6062
- *WebSphere MQ Security,* SC34-6079

- *Program Directory for WebSphere MQ for z/OS*, GI10-2548
- *z/OS MVS System Commands,* SA22-7627
- *z/OS Parallel Sysplex Overview: Introducing Data Sharing and Parallelism in a Sysplex*, SA22-7661
- *z/OS MVS Setting Up a Sysplex,* SA22-7625
- z/OS *MVS Sysplex Services* Guide*,* SA22-7617
- *z/OS MVS Installation Exits,* SA22-7593
- *z/OS MVS Planning: Global Resource Serialization,* SA22-7600
- *z/OS MVS Planning APPC/MVS Management,* SA22-7599
- *z/OS MVS Planning: Workload Management,* SA22-7602
- *z/OS UNIX System Services Planning,* GA22-7800
- *A Positioning Paper March 2002 Coupling Facility Configuration Options,* GF22-5042
- *DB2 UDB for OS/390 and z/OS Data Sharing: Planning and Administration***,** SC26-9935

Referenced Web sites

These Web sites are also relevant as further information sources:

- IBM WebSphere MQ home page

  `http://www.software.ibm.com/ts/mqseries`
- IBM download page for WebSphere MQ manuals in PDF format

  `http://www.software.ibm.com/ts/mqseries/library/manualsa/index.html`
- IBM WebSphere MQ SupportPacs

  `http://www.software.ibm.com/ts/mqseries/txppacs`
- GDPS

  `http://www.ibm.com/servers/eserver/zseries/announce/april2002/gdps.html`
- Coupling facility information

  `http://www.ibm.com/servers/eserver/zseries/pso/cftable.html`
- Coupling facility structure sizer tool

  `http://www.ibm.com/servers/eserver/zseries/cfsizer`
- z/OS home page

  `http://www.ibm.com/servers/eserver/zseries/zos/`

- DB2 for z/OS and OS/390

  http://www.ibm.com/software/data/db2/os390/

How to get IBM Redbooks

You can order hardcopy Redbooks, as well as view, download, or search for Redbooks at the following Web site:

ibm.com/redbooks

You can also download additional materials (code samples or diskette/CD-ROM images) from that site.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the CD-ROMs button on the Redbooks Web site for information about all the CD-ROMs offered, as well as updates and formats.

Index

D

U

V

W

X

Z

WebSphere MQ in a z/OS Parallel Sysplex Environment

(0.5” spine)
0.475”<->0.875”
250 <-> 459 pages

WebSphere MQ in a z/OS Parallel Sysplex Environment

Redbooks

Learn how WebSphere MQ makes use of z/OS Quality of Service features

Provide a high-availability environment for your message-driven applications

Develop modern message-driven CICS Java applications

This IBM redbook looks at the latest enhancements to WebSphere MQ for z/OS and shows how you can make use of the z/OS Parallel Sysplex to improve throughput and availability of your message-driven applications. It helps you configure and customize your system to use shared queues in a high-availability environment and to migrate from earlier releases.

In the first part of the redbook, we provide an overview of z/OS Parallel Sysplex Technologies that you can integrate into your WebSphere MQ solution. We describe:

- z/OS Resource Recovery Services (RRS)
- Automatic Restart Manager (ARM)
- Sysplex Distributor

In the second part of the redbook, we introduce advanced features of WebSphere MQ for z/OS V5.3, namely the queue-sharing and clustering technologies.

In the third part of the redbook, we describe our test implementation scenarios and cover the following topics:

- Implementation of a queue-sharing environment
- Migration from MQSeries V5.2
- High-availability configurations
- CICS, WebSphere MQ Integrator Broker, and IMS applications in a queue-sharing environment

Finally, we look at the operational impact of a queue-sharing environment and discuss failure and recovery scenarios.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-6864-00 ISBN 0738427349