Highly Available WebSphere Business Integration Solutions

- Develop a WebSphere Business Integration solution under HA control
- Explore middleware behavior under HA
- Example implementation with supporting HA scripts

Alicia Harvey
Rob Convery
Carlos Flores
Yvonne Hille
Sriram Madapura
Arnold Shoon

ibm.com/redbooks
Note: Before using this information and the product it supports, read the information in “Notices” on page ix.


This edition applies to Version 4, Release 2, Modification 2 of WebSphere Business Integration Server and Version 2, Release 4 of WebSphere Business Integration Adapters.

© Copyright International Business Machines Corporation 2004. All rights reserved.
Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
# Contents

Notices ............................................................................................................................ ix
Trademarks ........................................................................................................................ x

Preface ............................................................................................................................... xi
The team that wrote this redbook. .................................................................................... xii
Become a published author ............................................................................................ xiv
Comments welcome ....................................................................................................... xiv

Chapter 1. Introduction to this IBM Redbook ................................................................. 1
1.1 Objective of this redbook ........................................................................................... 2
1.2 Structure of this redbook .......................................................................................... 3
1.3 Overview of the sample implementation .................................................................. 4

Part 1. High availability and WebSphere Business Integration Architecture ................. 9

Chapter 2. Overview of high availability concepts ......................................................... 11
2.1 Introduction ............................................................................................................... 12
2.2 Availability levels ...................................................................................................... 12
2.3 Causes of downtime ................................................................................................... 14
2.4 The focus of this redbook ......................................................................................... 15
2.5 Clustering for high availability ................................................................................ 15
2.6 Advantages of HA clustering technologies ............................................................... 19

Chapter 3. Design considerations .................................................................................. 21
3.1 IBM WebSphere Business Integration solution architecture ..................................... 22
3.2 WebSphere Business Integration component dependencies ..................................... 22
3.3 WebSphere MQ availability ...................................................................................... 26
   3.3.1 Message availability .......................................................................................... 27
   3.3.2 WebSphere MQ queue manager clustering ....................................................... 28
3.4 InterChange Server availability ................................................................................ 30
3.5 Message Broker availability ..................................................................................... 33
3.6 HA cluster design ...................................................................................................... 34
   3.6.1 Highly available WebSphere MQ .................................................................... 38
   3.6.2 Highly available Message Broker ................................................................. 39
   3.6.3 Highly available InterChange Server ............................................................... 40
   3.6.4 A few possible HA configurations ................................................................. 42
   3.6.5 Setup1 .............................................................................................................. 43
   3.6.6 Setup2 .............................................................................................................. 44
3.7 Capacity planning ....................................................................................................... 46
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.8 Maintenance planning</td>
<td>46</td>
</tr>
<tr>
<td>3.8.1 Administration tasks</td>
<td>47</td>
</tr>
<tr>
<td>3.8.2 Adding and replacing hardware</td>
<td>47</td>
</tr>
<tr>
<td>3.8.3 Applying software upgrades and fixes</td>
<td>48</td>
</tr>
<tr>
<td>3.8.4 Deploying new interfaces</td>
<td>49</td>
</tr>
<tr>
<td>3.9 Performance</td>
<td>50</td>
</tr>
<tr>
<td>3.9.1 Minimize unmonitored downtime</td>
<td>50</td>
</tr>
<tr>
<td>3.9.2 Minimize failover time</td>
<td>50</td>
</tr>
<tr>
<td>3.9.3 General performance under HA</td>
<td>51</td>
</tr>
<tr>
<td>Part 2. Developing and implementing a solution</td>
<td>53</td>
</tr>
<tr>
<td>Chapter 4. Business scenario</td>
<td>55</td>
</tr>
<tr>
<td>4.1 Business challenge</td>
<td>56</td>
</tr>
<tr>
<td>4.2 Sample scenario</td>
<td>56</td>
</tr>
<tr>
<td>4.2.1 Scenario design</td>
<td>56</td>
</tr>
<tr>
<td>4.2.2 Description of the solution</td>
<td>57</td>
</tr>
<tr>
<td>4.2.3 Data store</td>
<td>62</td>
</tr>
<tr>
<td>4.2.4 WebSphere MQ connectivity</td>
<td>63</td>
</tr>
<tr>
<td>Chapter 5. Implementing the solution</td>
<td>65</td>
</tr>
<tr>
<td>5.1 Cluster layout</td>
<td>66</td>
</tr>
<tr>
<td>5.2 Solution architecture</td>
<td>68</td>
</tr>
<tr>
<td>5.3 Preparing components for HA</td>
<td>72</td>
</tr>
<tr>
<td>5.3.1 Initial environment</td>
<td>72</td>
</tr>
<tr>
<td>5.3.2 Configuring WebSphere MQ</td>
<td>72</td>
</tr>
<tr>
<td>5.3.3 Configuring remote adapters</td>
<td>75</td>
</tr>
<tr>
<td>5.3.4 Configuring Active/Passive Message Broker</td>
<td>75</td>
</tr>
<tr>
<td>5.3.5 Configuring Active/Active Message Broker</td>
<td>76</td>
</tr>
<tr>
<td>5.3.6 Configure system manager to connect to remote InterChange Server</td>
<td>78</td>
</tr>
<tr>
<td>5.3.7 Configuring InterChange Server and WebSphere BI Adapters</td>
<td>78</td>
</tr>
<tr>
<td>5.3.8 Configuring the ORB</td>
<td>79</td>
</tr>
<tr>
<td>5.3.9 Shared disk layouts</td>
<td>80</td>
</tr>
<tr>
<td>5.4 HA monitoring, start, stop scripts</td>
<td>80</td>
</tr>
<tr>
<td>5.4.1 WebSphere MQ</td>
<td>80</td>
</tr>
<tr>
<td>5.4.2 Message Broker</td>
<td>80</td>
</tr>
<tr>
<td>5.4.3 Remote WebSphere Business Integration Adapters</td>
<td>81</td>
</tr>
<tr>
<td>5.4.4 InterChange Server and WebSphere BI Adapters</td>
<td>81</td>
</tr>
<tr>
<td>5.4.5 InterChange Server remote adapter resource group</td>
<td>83</td>
</tr>
<tr>
<td>5.5 System setup</td>
<td>84</td>
</tr>
<tr>
<td>5.5.1 Hardware</td>
<td>84</td>
</tr>
<tr>
<td>5.5.2 Software</td>
<td>84</td>
</tr>
<tr>
<td>5.6 Adding applications to HACMP control</td>
<td>85</td>
</tr>
</tbody>
</table>
WBI Message Broker .............................................. 150
hamqsi_createbroker ........................................... 150
hamqsi_addbrokerstandby ................................. 154
hamqsi_deletebroker ......................................... 158
hamqsi_remotebrokerstandy ......................... 162
hamqsi_qm_datapath ......................................... 164
hamqsi_start_broker ......................................... 167
hamqsi_start_broker_as .................................... 170
hamqsi_stop_broker .......................................... 175
hamqsi_stop_broker_as ...................................... 178
hamqm_lsnr_start ............................................. 181
hamqm_lsnr_start_su ......................................... 182
hamqm_lsnr_stop .............................................. 184
hamqm_lsnr_stop_su ......................................... 185
hamqsi_start_broker_as.MBBRK ............................ 186
hamqsi_stop_broker_as.MBBRK ............................ 186
hamqsi_start_broker_as.MB2BRK ............................ 186
hamqsi_stop_broker_as.MB2BRK ............................ 186
hamqsi_monitor_broker_as ................................. 187
hamqsi_applmon.MBBRK ....................................... 192
hamqsi_applmon.MB2BRK ....................................... 192
Message Broker adapter scripts ......................... 193
remote_adapter_ha_monitor ............................... 193
remote_adapter_ha_start .................................... 195
remote_adapter_ha_stop ..................................... 196
remote_adapter_restart ..................................... 198
set_remote_adapter_variables ............................. 198
InterChange Server scripts ................................. 199
wics_ha_start ................................................... 199
wics_ha_stop ................................................... 203
wics_ha_monitor .............................................. 205
wics_orb_start .................................................. 207
wics_orb_stop .................................................. 208
wics_orb_test ................................................... 208
wics_orb_restart .............................................. 209
wics_mq_restart ............................................... 210
ics_restart ....................................................... 211
wics_adapter_restart ......................................... 212
InterChange Server remote adapter scripts ........... 214
rmt_adap_ha_start ............................................. 214
rmt_adap_ha_stop .............................................. 216
rmt_adap_ha_monitor .......................................... 217
rmt_adap_restart .............................................. 218
Appendix B. Additional material ........................................ 221
Locating the Web material ............................................. 221
Using the Web material ................................................. 221
  System requirements for downloading the Web material .......... 222
  How to use the Web material ........................................ 222
Abbreviations and acronyms ............................................ 223
Related publications ..................................................... 225
  IBM Redbooks .......................................................... 225
  Other publications ..................................................... 225
  Online resources ....................................................... 225
  How to get IBM Redbooks ........................................... 226
  Help from IBM .......................................................... 226
Index ............................................................................ 227
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:

<table>
<thead>
<tr>
<th>IBM @server®</th>
<th>DB2®</th>
<th>SupportPac™</th>
</tr>
</thead>
<tbody>
<tr>
<td>Redbooks (logo)™</td>
<td>HACMP™</td>
<td>WebSphere®</td>
</tr>
<tr>
<td>pSeries®</td>
<td>IBM®</td>
<td></td>
</tr>
<tr>
<td>AIX®</td>
<td>Redbooks™</td>
<td></td>
</tr>
<tr>
<td>DB2 Universal Database™</td>
<td>Redbooks (logo)™</td>
<td></td>
</tr>
</tbody>
</table>

The following terms are trademarks of other companies:

Microsoft, Windows, 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.

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

This IBM® Redbook discusses the design considerations involved in architecting a WebSphere® Business Integration (WebSphere BI) solution in a high availability environment. The WebSphere Business Integration components covered in this redbook include IBM WebSphere MQ, Version 5.3; IBM WebSphere Business Integration Message Broker, Version 5.0; IBM WebSphere InterChange Server, Version 4.2.2; and several adapters from the IBM WebSphere Business Integration Adapters, Version 2.4, portfolio.

We detail the implementation of a specific WebSphere Business Integration solution in a highly available environment, providing an end-to-end narrative of designing, implementing, and testing the solution. In conjunction with the testing, the behavior of both a complex WebSphere Business Integration solution and the portfolio components under various failure conditions is characterized.

Lastly, we provide a compendium of best practices, hints, and tips to smooth the practitioner's journey toward a highly available WebSphere Business Integration implementation.
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.

Sriram, Rob, Yvonne, Carlos, Arnold, and Alicia

**Alicia Harvey** is an IT Specialist with the IBM International Technical Support Organization, Raleigh Center. Alicia has been a Systems Integrator with IBM for 11 years. For the last 5 years she has specialized in EAI solutions using the WebSphere MQ family of products. Alicia writes extensively and teaches IBM classes worldwide on all areas of the WebSphere MQ family and Business Integration.

**Rob Convery** is a Software Engineer at the IBM Hursley Lab in the UK. He has four years of experience in Messaging. He holds a degree in Computer Systems Engineering from Nottingham University. His areas of expertise include WebSphere MQ and WebSphere Business Integration Message Broker.

**Carlos Flores** is an IT Architect in WebSphere Business Integration services based in the US. Carlos has extensive experience in the area of WebSphere Business Integration and high availability solutions. Carlos has worked at IBM for four years. His areas of expertise include middleware application architecture design, MQ & JMS, Publish & Subscribe, WebSphere MQ & WebSphere Business Integration Message Broker implementations, and highly available messaging solutions. He has written extensively and has presented at various
forums on the various topics relating to WebSphere Business Integration and high availability.

**Yvonne Hille** is an IT Specialist in WebSphere Business Integration based in the UK. She has 4 years of experience in Business Integration. She holds a degree in Mathematics from Marburg University, Germany. Her areas of expertise include WebSphere InterChange Server and WebSphere Business Integration Adapters.

**Sriram Madapura** is a Senior Consultant with the WebSphere Business Integration Focused Technology Practice group, which is part of the WebSphere Software Services organization. He holds a bachelor's degree in Materials Science from the Indian Institute of Technology Kanpur, and a master's degree in Electrical Engineering from the University of Cincinnati.

**Arnold Shoon** is a pre-sales IT Specialist in Toronto, Canada, with experience in the Enterprise Application Integration field. He holds a degree in Electrical Engineering from Carleton University with expertise in IBM's WebSphere Business Integration Server.

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

- **Octavian Lascu**  
  International Technical Support Organization, Poughkeepsie Center

- **Margaret Ticknor**  
  International Technical Support Organization, Raleigh Center

- **Craig Alexander**  
  IBM Partners in Development Technology Services, Systems Group

- **Jack Carnes**  
  IBM WW Business Integration Technical Sales Support

- **Greg Clarke**  
  IBM Development Lab Services, Systems Group

- **Chris Eisenmann**  
  IBM @server® and pSeries® Customer Benchmarking, Advanced Technical Support, Americas

- **Lyn Elkins**  
  IBM Advanced Technical Support, Americas

- **Bob Herbison**  
  IBM Advanced Technical Support, Americas
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
Introduction to this IBM Redbook
1.1 Objective of this redbook

This IBM Redbook discusses the design considerations involved in architecting a WebSphere Business Integration (WebSphere BI) solution in a high availability (HA) environment.

The WebSphere Business Integration components covered in this redbook include:

- IBM WebSphere MQ, Version 5.3
- IBM WebSphere Business Integration Message Broker, Version 5.0
- IBM WebSphere InterChange Server, Version 4.2.2
- Several adapters from the IBM WebSphere Business Integration Adapters, Version 2.4 portfolio

This redbook details the implementation of a specific WebSphere Business Integration solution in a highly available environment, providing an end-to-end narrative of designing, implementing, and testing the solution.

Through the testing we verify that:

- The applications (message flows, collaborations, etc.) can recover and restart.
- The application will continue to process or recover when a component fails.
- The scenario will perform under a heavy load.
  
  High availability is also about the performance of the scenario when individual components are under stress, not just failure.
- In a configuration with multiple brokers, the scenario continues to work such as when an application fails on one node, and it is restarted on another.

In conjunction with the testing, we explore the behavior of both a complex WebSphere Business Integration solution, and the portfolio components under various failure conditions.

Both failover and fallback (also known as failback) are addressed.

Lastly, this book provides a compendium of best practices, hints, and tips to smooth the practitioner's journey toward a highly available WebSphere Business Integration implementation.
1.2 Structure of this redbook

This redbook is organized into three parts, and it is designed to serve different audiences including IT specialists, IT architects, system administrators, and anyone with a knowledge of the WebSphere Business Integration suite who wants to learn something about WebSphere Business Integration middleware in the context of high availability.

Part 1: High availability and WebSphere Business Integration Architecture

Part 1 of this redbook is intended for anyone with an understanding of the WebSphere Business Integration portfolio with interest in high availability, as well as IT specialists, IT architects, and system administrators. The chapters in this portion of the book provide a foundation of the key concepts of an implementation of the WebSphere Business Integration portfolio in a high availability environment, including a general overview of high availability.

The chapters in Part 1 are as follows:

- Chapter 2, “Overview of high availability concepts” on page 11
- Chapter 3, “Design considerations” on page 21

Part 2: Developing and implementing a solution

Part 2 is intended for the more advanced practitioner such as IT specialists, IT architects, and system administrators. The chapters in this portion of the book detail the implementation of a specific WebSphere Business Integration solution in a highly available environment. The behavior of that implementation under various failure scenarios is also explored.

The chapters in Part 2 of this redbook are as follows:

- Chapter 4, “Business scenario” on page 55.
- Chapter 5, “Implementing the solution” on page 65.
- Chapter 6, “Representative test plan” on page 87.
- Chapter 7, “Testing and validation” on page 95.
- Chapter 8, “A few helpful tips” on page 121.

Part 3: Appendixes

Part 3 is intended for the advanced practitioner or HA implementor. It contains the HA scripts that the authors used in implementing the sample solution provided in Part 2.

The content of Part 3 is:

- Appendix A, “HA scripts” on page 127.
1.3 Overview of the sample implementation

The sample WebSphere Business Integration solution that is examined in this redbook is implemented under AIX® Version 5.2, Maintenance Level 1. The supporting high availability cluster is implemented using HACMP™ Version 5.1.

The WebSphere Business Integration scenario is shown in Figure 1-1.

![Diagram of the scenario overview]

The dashed box contains those components that are made highly available in the example implementation.

The solution is implemented on two p-Series machines. The storage that these machines share for high availability purposes is a FAStT Storage server. There is a single HA cluster.
The basic scenario is implemented under three different takeover configurations. This allows the authors the opportunity to achieve and explore a number of different HA capabilities. The first configuration shows Active/Passive takeover. The distribution of resources for this scenario implementation is shown in Figure 1-2.

![HACMP Cascading Cluster](image)

**Figure 1-2  Active/Passive resource group configuration**

Two resource groups are implemented on node 2 to allow the authors to emulate the existence of a third machine in the HA cluster.
The second configuration adds an Active/Active configuration for Message Broker. The distribution of resources for this configuration is shown in Figure 1-3.
In addition to the Active/Active Message Broker setup, the final configuration allows the authors to simulate remote WebSphere Business Integration Adapters that are under HA control. Figure 1-4 shows the distribution of resources in this configuration.

![HACMP Cascading Cluster](image_url)

**Figure 1-4  Active/Active MB, remote adapter resource group configuration**
Part 1

High availability and WebSphere Business Integration Architecture
Overview of high availability concepts

In this chapter we discuss the systems availability concepts that are employed across both software and hardware clustering technologies. The purpose is to acquaint the reader with common terminology, definitions, and practices that are used when discussing systems availability. Discussion surrounding availability focuses on systems resources. It is assumed that the IT professional is familiar with the operating system architecture, disk sub-systems, and networking topology for the specific platforms being considered.

Below is a brief description of the topics discussed in this chapter:

- Defining high availability
- Measuring high availability
- Commonly used high availability terminology
- Advantages of clustering technologies
2.1 Introduction

Modern businesses have become increasingly dependent upon the availability of their enterprise applications and data. The rapid growth of the Internet and the need to conduct business over the network have reinforced the requirement to provide access to business systems on a 24 hour basis.

Although it is necessary to keep critical applications running, it is important to consider that there will be times when the computer system has to be shut down in order to perform maintenance. These outages are referred to as planned outages. There will also be times when the system will be shut down because of hardware or software failures. These are referred to as unplanned outages. In both cases of planned and unplanned outages, the desire of enterprises is to minimize these periods of downtime of their systems.

There are different levels of availability that a particular system can require. The description of this requirement is called the service level, and it defines the degree of service that the system will provide to its users. This service level is often formalized in a service level agreement document offered by the service provider or agreed on between the service provider and the end user.

2.2 Availability levels

There are several ways to describe the degree of availability that a system requires. The degree of availability of a system is often expressed as a number of nines.

The number of nines references the percentage of time for which the system is available. Three nines means that it is available 99.9% of the time and five nines means 99.999%. These numbers become much more significant when you look at these figures in terms of downtime over a fixed period. For example, over a year a system with 99.9% availability can be expected to suffer only 8.75 hours of downtime, while a system with 99.999% availability will be down for just 315 seconds.

Table 2-1 shows availability as a number of nines and the associated downtime figures.
The following terms are also used to describe availability.

**Continuous availability**

The term *continuous availability* is used to describe a system that experiences no discernible downtime; neither scheduled nor unscheduled outages should occur. A continuously available system should detect errors and immediately be in a position to provide an alternative component that is already revved up. Also, it should support the scheduling of planned maintenance by allowing workload to be transparently transferred away from the components or subsystems that are the subject of the maintenance activity. Although this goal seems difficult to achieve, it is possible to obtain such availability by combining hardware, software, and operational procedures that will *mask* outages from the end user; that is, the user will not perceive that a system outage has occurred.

**Continuous operation**

The term *continuous operation* is used to describe a system that experiences no discernible downtime due to scheduled outages. However, this system's availability will not be as high as that of a continuously available system because a system that supports continuous operation may suffer unplanned outages.

**High availability (HA)**

The term *high availability* is used to describe a system that can detect the failure of a component of the system, and react to it automatically within a matter of a few minutes at most. To the user, a system with high availability will continue to operate, albeit after perhaps a brief delay, despite a non-catastrophic outage.

There are two significant aspects in this definition:

1. The system should survive a single failure but a second failure may result in a loss of service.
2. The detection of a fault and the triggering of an action to recover from it should be automatic; that is, require no manual intervention.

Figure 2-1 illustrates the relationship among the levels of system availability.
Fault tolerance
The term fault tolerant is often mistakenly used instead of high availability or continuous availability. It describes systems that in the event of a failure can switch in a replacement component in a matter of a few milliseconds. This is achieved through the use of components that have redundant sub-components, error checking and correction for data, retry capabilities for basic operations, alternate path for I/O requests, and so forth.

Single point of failure (SPOF)
A single point of failure is a single component of a system (either hardware or software) that is vital, and if it fails unexpectedly or is taken out of service for maintenance, the effect is the loss of the entire system or at least a substantial part of it.

2.3 Causes of downtime
As mentioned earlier, outages can be either planned or unplanned events. Clearly, the impact of a planned outage can be minimized by scheduling the outage outside of business hours for those business that require, for example, 20x6x52 (required availability is 20 hours a day, 6 days a week, 52 weeks a year) or 12x5x52 availability. For the business that requires a very high degree of
availability on the order of, for example, 24x7x365 availability, the duration of a planned outage can be minimized by applying techniques such as rolling upgrades for software and hot replacement of hardware. Additionally, these events can be scheduled during periods of lighter system load.

Reasons for planned outages include:

- The installation of software upgrades or patches
- Periodically scheduled backups
- Hardware repairs or expansions
- The physical movement of applications of hardware

Unplanned outages are the more urgent concern as they may occur at a time when the system is required to be operational. The major contributors to unplanned downtime are:

- Hardware failure
- Software failure
- Human error
- Environmental conditions: climate control, power failure, power spike
- Natural disasters: fire, flood, earthquake

### 2.4 The focus of this redbook

In this book, the primary focus is on solutions to help manage hardware and software failure. Examples of the types of failures the given solution addresses are:

- Network failures
- Disk failures
- Processor failures
- Application software defects
- System software defects

There are several different ways to accomplish the various levels of system availability. These include the combination of software, hardware, and operational procedures that work together.

### 2.5 Clustering for high availability

Clustering solutions that provide failover support are the prevalent mechanism used to achieve high levels of availability.
IBM High Availability Cluster Multi-Processing for AIX (HACMP) is the platform specific clustering solution provided by IBM for pSeries servers running AIX. HA clustering solutions are provided by many vendors; among these solutions are: Sun Cluster, Microsoft® Cluster Server (MSCS), VERITAS Cluster Server (VCS), HP Serviceguard, and many others.

These clustering solutions are transparent to the applications that run under their control. Applications do not have to be modified before they can be deployed within an HA cluster.

Commercial HA clustering solutions allow for a wide range of cluster configurations. The simplest cluster configuration is comprised of a pair of servers, and an external SCSI or Fibre Channel storage subsystem.

The following terms are generally used when discussing cluster technologies.

**Cluster**
A *cluster* is a grouping of two or more interconnected computers that are viewed and used as a single computing resource.

**Node**
A *node* is an individual computer system (including its hardware resources such as local disks, processors, memory, etc.) which runs both the operating system and the corresponding clustering software.

Nodes can be uniprocessors, or as in the case of a large scalable system, symmetric-multi processors (SMP). An SMP node looks like a uniprocessor node to most of the clustering technologies. Each node in the cluster can vary in the number of processors. The number of nodes that can participate in a single cluster varies depending on the clustering technology being implemented.

**Resource groups**
*Resource groups* enable the combination of related resources into a logical entity. Most of the available systems clustering technologies employ the concept of moving the system resources that are contained within a resource group. Resource groups facilitate moving critical workload from one node to another, and the continuation of critical workload processing on a different cluster node. This concept will become more prevalent as we discuss system and application availability configurations, especially in the case of MQ and Message Broker.
Volume groups

A *volume group* is a set of physical volumes that AIX treats as a contiguous, addressable disk region. Volume groups are configured to AIX. They can be included in resource groups in IBM High Availability Cluster Multi-Processing (HACMP) for AIX, the clustering software on AIX. In the HACMP environment, a *shared volume group* is a volume group that resides entirely on the external disks shared by the cluster nodes.

Failover

*Failover* is the event that occurs when the HA cluster technology detects that there has been a failure of one of the system resources, and moves resource groups from the failed member of the cluster to a working member.

Fallback

*Fallback* refers to the movement of a resource group from a node it currently resides on (not the home node of this resource group) to a node that has joined or is being reintegrated into the cluster.

Failback

*Failback* is another term for fallback.

Cluster topologies

Clusters may have a number of different topologies. Examples of cluster topologies are clustered pairs, rings, N+1 configurations (also called *standby configurations*), and mutual takeover configurations. It is also possible to create configurations in which one or more nodes act as standby nodes, running other workload if desired.

The following are descriptions of several cluster topologies.

*N+1 standby takeover configuration*

An *N+1 standby takeover configuration* is a configuration of a two or more node clusters in which for each of the \( N \) nodes that are performing work there is another node that acts only as a standby. The standby node does not perform...
Highly Available WebSphere Business Integration Solutions

work and is referred to as *idle*. This configuration is sometimes referred as *Active/Passive* and requires a high degree of hardware redundancy. In order to economize on hardware, it is possible to extend this configuration to have multiple worker nodes with a single standby node, the idea being that the standby node can take over the work of any worker node. Such simple duplication is frequently known as Active/Passive, or N+1 redundancy, where N is the number of nodes running critical workload, and 1 is the number of available standby nodes, which can be used if necessary.

The terms *hot* and *cold* refer to the state of readiness of the standby nodes. A *hot standby* is active and ready to perform work. A *cold standby* is inactive, and will need to be made active before it can perform any work. With a hot standby, the failover process will be quicker than with a cold standby.

**One-sided takeover configuration**

The *One-sided takeover* configuration is a variant of the N+1 standby takeover configuration. The difference is that the standby node performs some additional, non-critical and non-movable work.

**Mutual takeover configuration**

The *mutual takeover* configuration, which is also referred to as a *hot standby* or *Active/Active* configuration, is a more advanced configuration in which all of the nodes are actively processing critical workload. Critical work can be taken over by any of the nodes in the event of a node failure. Takeover configurations make more efficient use of hardware resources than do standby configurations because there is no idle processor.

With a standby configuration consisting of more than two nodes (extended standby configuration) or either of the takeover configurations, it is important to consider the peak load that will be placed on a node when it is processing not only its own work but the work of the other nodes that may failover onto it. Such a node must possess sufficient capacity to maintain an acceptable level of performance when failover occurs.

**IP address takeover (IPAT)**

*IP address takeover* is a mechanism for recovering a service IP address by moving it to another network adapter on another node when the initial network adapter fails. IPAT is useful because it ensures that an IP label over which services are provided to the client nodes remains available.

**RAID**

Fundamental to *Redundant Arrays of Inexpensive Disks (RAID)* is *striping*, which is a method of concatenating multiple drives into one logical unit. RAID-1 through
RAID-5, provide different levels of disk fault-tolerance. Among the different types of RAID, there are different trade-offs in features and performance.

**Shared external disk devices**

Each node must have access to one or more shared external disk devices. A shared external disk device is a disk physically connected to multiple nodes. The shared disk stores mission-critical data, typically mirrored or RAID configured for data redundancy. A node must also have internal disks that store the operating system and application binaries, but these disks are not shared.

**Heartbeat messages**

In order for a cluster to recognize and respond to failures, it must continually check the health of the nodes, resources, and applications running within the cluster. Some of these checks are provided by the heartbeat function. Each cluster node sends heartbeat messages at specific intervals to other cluster nodes, and expects to receive heartbeat messages from the nodes at specific intervals. If messages stop being received, the cluster software recognizes that a failure has occurred.

Heartbeats can be sent over:

- TCP/IP networks
- point-to-point networks
- shared disks

**2.6 Advantages of HA clustering technologies**

Some of the advantages that are provided by all system clustering technologies include:

- Automatic failure detection and failover

- Grouping of hardware and software components in a way that allows moving the whole group from one node to another

- Minimization of system downtime
Design considerations

This chapter discusses general WebSphere Business Integration (WebSphere BI) high availabilty design considerations. This information will help prepare the reader to design a WebSphere Business Integration system that provides its mediation and synchronization services according to the environment's Service Level Agreements (SLA).
3.1 IBM WebSphere Business Integration solution architecture

The IBM WebSphere Business Integration portfolio is a suite of middleware products that are loosely coupled to each other. The product suite delivers the integration capabilities that are required for comprehensive, enterprise-wide application integration (EAI). The IBM WebSphere Business Integration Adapters provide the connectivity to and from enterprise applications. IBM WebSphere MQ provides the underlying transport layer. IBM WebSphere Business Integration Message Broker provides routing and mediation services, where as IBM WebSphere Interchange Server or IBM WebSphere MQ Workflow act as processing engines, and implement business synchronization logic. The availability of a system is a product of the availability of all its factors. In order to build a high availability integration solution, it is therefore important to understand the dependencies among the products.

3.2 WebSphere Business Integration component dependencies

The following bullet points describe the dependencies between various WebSphere Business Integration components:

▶ InterChange Server and Message Broker

The brokers are loosely coupled to each other. The brokers communicate with each other through MQ queues and the IBM WebSphere Business Integration Adapter for WebSphere MQ. As a result, the availability of InterChange Server does not directly influence the availability of Message Broker and vice versa.

▶ InterChange Server and WebSphere MQ

Generally, the availability of InterChange Server itself does not influence the availability of MQ. InterChange Server's dependence on MQ is determined by which transport mechanisms the adapters have been configured to use.

All of the adapters have been configured to use the same transport mechanisms, which are:

- **IDL**, InterChange Server, and adapters are not at all dependent on a queue manager for transport of business objects between the adapter agent and controller.

- **MQ**, polling adapters will stop delivering objects to InterChange Server. InterChange Server will continue to process events that are in process
already. Objects can still be sent to target adapter agents and be processed by them.

- **JMS**, adapter controllers will be stopped if the queue manager becomes unavailable. Before the queue manager can be restarted, InterChange Server has to be restarted. After a restart of InterChange Server and the associated queue manager, the controllers that use JMS as transport have to be manually restarted. (The team has raised this issue as a defect.) Objects that are sent to the target adapter while the controller is stopped will fail and have to be manually resubmitted.

### WebSphere Business Integration Adapters and WebSphere MQ

- Adapters that are configured to use **JMS** as their transport mechanism are directly dependent on WebSphere MQ for transport of business objects, and for administrative communication between the adapter agent and adapter controller. In the event that the queue manager becomes unavailable, the adapter agent and controller shut down. They have to be restarted following the restart of the MQ queue manager and InterChange Server.

- Adapters that are configured to use **MQ** as their transport mechanism are dependent on WebSphere MQ for event delivery. In the event that the queue manager becomes unavailable with a request processing adapter (an adapter that sends objects to a target destination), the controller is unable to delete events after delivery. InterChange Server must be restarted for processing to continue. But, if a polling adapter is unsuccessful in putting a polled business object on the delivery queue, the adapter agent shuts down and has to be restarted. In the case where the queue manager becomes available again before a polling adapter tries to access its delivery queue, the adapter agent does not have to be restarted and continues polling.

- Adapters that are configured to use **IDL** as their transport mechanism are not dependent on WebSphere MQ. In the case where the queue manager becomes unavailable, the adapter agent and controller continue event and request processing as normal.

### WebSphere Business Integration Adapters and InterChange Server

For a process to be synchronized successfully, all source and target adapters and InterChange Server have to be available. However, if one of the components becomes unavailable, only that one component needs to be restated. (In the authors' experience, agents reconnect even after hours.) Adapter agents ping InterChange Server every 30 seconds, and will stop processing if no response is received from the server. InterChange Server can continue processing unless all adapter agents become unavailable.
InterChange Server/WebSphere Business Integration adapters and the IBM Java™ Object Request Broker

In WebSphere Business Integration Version 4.2.2.2, the Interchange Server uses the IBM Java Object Request Broker (ORB) in order to communicate with the deployment and monitoring toolset. The adapter controllers and adapter agents use the ORB to exchange administrative and initialization communication. Depending on the adapter configuration, the ORB can also be used as the business object transport layer between the adapter agents and controllers:

- For the initial start of InterChange Server and WebSphere Business Integration adapters, the ORB needs to be running.
- For a restart of InterChange Server or WebSphere Business Integration adapters, the ORB does not need to be restarted.
- For a restart of the ORB, it is not required to restart InterChange Server or the WebSphere Business Integration adapters, if the old inter orb reference (ior) file is provided when restarting the ORB. That is only true when WebSphere Business Integration version 4.2.2.3 and not version 4.2.2.2 is used. Version 4.2.2.3, which is the version that was used during this residency, contains a fix for the ORB, which makes the ORB name server persistent.\(^1\)

- As an aside, the InterChange Server and the ORB share a single file called CxCosNameRepos.ior.

WebSphere Business Integration Adapters and the ORB

WebSphere Business Integration Adapters that use the Message Broker or WebSphere Application Server as integration brokers are not dependent on the ORB. There is no exchange of administration or initialization communications between the adapters and these integration brokers.

Message Broker and WebSphere MQ

Message Broker requires an MQ queue manager to be available. If the queue manager becomes unavailable, the in-flight message will be rolled back and will be retriggere after the queue manager and Message Broker are restarted. Currently (for Message Broker Version 5.0. CSD 3 and below), the Message Broker has to be restarted after the queue manager is restarted.

Message Broker and WebSphere Business Integration adapters

These components are loosely coupled and communicate only through WebSphere MQ. The availability of one component does not influence the availability of the other. Both components will put their messages asynchronously on the request/delivery queues, and the corresponding component will pick up the messages once it becomes available again.

---

\(^1\) The WebSphere InterChange Server 4.2.2.3 Fix Pack is available as of June 18, 2004.
Figure 3-1 shows the component interaction and communication for an event flow through the InterChange Server. Only the flow and transport layer germane to the example scenario detailed in Chapter 4., “Business scenario” on page 55, and implemented in Chapter 5., “Implementing the solution” on page 65 are depicted.

**Legend:**
1. The event is triggered by the source application.
2. The event is polled by the polling adapter.
3. The event is delivered to the delivery queue and the event is inserted in the work-in-progress (WIP) table.
4. The event is picked up by the adapter controller.

*Figure 3-1  WebSphere Interchange Server event processing*
5. The event is sent to the InterChange Server for mapping.
6. The event is sent to collaboration and updated in the WIP table.
7. The event is proccessed by collaboration, and sent to the target adapter.
8. The event is mapped to the outbound structure (application specific business object or ASBO).
9. The event is sent from the controller to the agent.
10. The event is received by the agent, and converted with bohandler/datahandler to the target format.
11. The event is sent to the controller.
12. The event is received by the controller.
13. The event is mapped to the generic structure (generic business object or GBO).
14. If successful, an event/event status is returned to collaboration, and the event is deleted from the WIP table. Otherwise, the event is updated in the WIP table, and a failed event entry is created in the failed events table.

3.3 WebSphere MQ availability

There are two main considerations that must be addressed when discussing a highly available MQ environment:

- Message resilience: How important is it that a message is not lost?
- Message availability: How important is it to have a running queue manager after a failure?

The following sections discuss techniques for achieving these objectives. The authors will address the queue manager clustering feature provided with the base MQ, the capabilities provided by HA clustering technology, and what can be achieved using a combination of their technologies. Queue manager clustering is discussed in 3.3.2, “WebSphere MQ queue manager clustering” on page 28.

Message resilience

The practice of protecting messages against loss is known as message resilience. Message resilience requires the ability to distinguish the messages that must be secured to prevent them from being lost due to certain failures (such as a disk failure) from those that it is not critical to recover (such as messages that can be recreated). MQ introduces message persistence as a way to protect data within messages. This feature allows messages to be defined as follows:
Persistent messages are written out to logs and queue data files. When a queue manager is restarted after a failure, it recovers these persistent messages from the logged data. The persistence of a message is taken either from the message descriptor (MQMD) or the default value for persistence associated with the queue to which the message was PUT. The performance of your application is affected when you use persistent messages.

When a queue manager is restarted, whether the stoppage is the result of an operator command or because of the failure of some part of the system, nonpersistent messages are discarded.

### 3.3.1 Message availability

For purposes of discussing message availability, messages will be considered in two groups relative to their state at the time of a queue manager failure. *Existing messages* are those messages that are committed on a queue at the time the queue manager fails. When a queue manager fails, access to existing messages is lost. These messages are said to be *marooned* with the failed queue manager. Messages that had yet to be committed on a queue when the queue manager failed or that are generated subsequent to the queue manager failure are considered *new messages*.

Remembering that message availability is concerned with whether an MQ application has access to the messages that it is processing, the following questions must be considered when determining the required level of message availability:

- How long can the applications wait to process marooned messages?
- How long can the applications wait to process new messages?
- Is there a requirement that messages be processed in sequence?

The objective of providing applications with a particular level of message availability can be addressed by employing the capabilities of queue manager clustering, HA clustering, or both.

In the case where MQ queue manager clustering alone has been implemented, marooned messages are inaccessible until the queue manager is back online following manual intervention. During restart, non-persistent messages will have been discarded, and persistent messages will once again be available. Access to new messages is uninterrupted. Note that queue manager clustering is not a high availability solution.

In the case where HA clustering alone has been implemented, upon automatic restart of the queue manager on the backup node, marooned non-persistent messages will have been discarded, and persistent messages will once again be
available. Access to existing and new messages is restored; albeit perhaps after a brief delay.

The combination of HA clustering and queue manager clustering provide uninterrupted access to both existing and new messages.

These facilities provide the means to achieve several levels of availability:

- **Continuous availability** - A requirement of continuous availability means that applications must (ideally) experience uninterrupted access to both existing and new messages.

  A combination of HA clustering and queue manager clustering is used to achieve this level of availability.

- **Automatic availability** - A requirement of automatic availability means that applications can experience interruptions in their access to either existing or new messages. Also, access must be restored without manual intervention or monitoring, thus minimizing downtime.

  HA clustering provides this level of availability.

- **Inherent availability** - For purposes of this discussion, inherent availability refers to the level of availability that the product can achieve “out of the box.” Inherent availability means that applications can experience interruptions when attempting to access either existing or new messages, and that access to these messages will not be restored without manual monitoring and intervention.

### 3.3.2 WebSphere MQ queue manager clustering

As stated previously, a high availability cluster is a set of two or more interconnected computers that are viewed and used as a single computing resource. This grouping of systems provides a mechanism by which it is possible to minimize the overall impact caused by the failure of any individual member of the cluster. The work that was being done by the member that failed passes to other members of the cluster.

A WebSphere MQ queue manager cluster is a group of two or more queue managers that are logically associated in some way. The queue managers that make up the cluster may be running on one or more computers.

The objectives of queue manager clustering are:

- Reduce system administration
- Provide workload balancing across the queue managers in the cluster
- Facilitate continuous operations
Queue manager clustering capabilities facilitate the automatic interconnection of the queue managers in the cluster, and allow queues to be shared among the members of the cluster. This support for multiple instances of a cluster queue offers improved scalability of MQ resources. The ability to have multiple instances of a cluster queue also increases overall availability. Also, because instances of the same queue can be defined on more than one queue manager, workload can be distributed throughout the queue managers in a cluster providing a degree of workload balancing.

The use of queue manager clustering reduces system administration by simplifying the configuration of an MQ network. Setting up a queue manager clustering environment requires far less effort than setting up a traditional MQ distributed network. There is no need to define sender channels to every other queue manager to which a queue manager wants to put messages; receiver channels from every queue manager that wants to put messages to that queue manager; transmission queues to every other queue manager with which a queue manager wants to communicate; and remote queue definitions for each remote queue to which the queue manager wants to put messages. The number of object definitions required is reduced considerably. Only two definitions per queue manager are necessary.

- A single cluster-sender channel used to connect to a full repository queue manager for the cluster.
- A single cluster-receiver channel to advertise to the cluster the queue manager IP address, port, and special channel attributes over which any queue manager can start a sender channel and send messages.

With the creation of these two objects, a given cluster queue manager and the applications that it supports have access to all of the other queue managers in the cluster. Once a cluster has been set up, it is a simple matter for the queue managers within the cluster to communicate with one another.

From a performance standpoint, queue manager clustering offers increased message throughput and decreased message response time. Any queue manager hosting an instance of a particular cluster queue can accept messages destined for that queue. Because more than one queue manager is able to accept the same type of message, the risk of delayed delivery when a queue manager or communications link is unavailable is greatly reduced. And, because a messaging application can send messages to any instance of a cluster queue, the impact caused by the failure of an individual cluster queue manager is decreased.

Figure 3-2 illustrates the characteristics of a queue manager cluster.
As shown in Figure 3-2, if an application doing MQPUTs has opened the output queue called Q1 specifying Bind Not Fixed, the loss of access to Q1 on QM3 will not stop operations. The messages will simply be routed to the remaining available instances of Q1 within the queue manager cluster CLUS1. For more in depth information on clustering, refer to the manual *WebSphere MQ Queue Manager Clusters*, SC34-6061.

### 3.4 InterChange Server availability

The availability of the InterChange Server system is built on a hardware clustering solution, which is discussed in 3.6, “HA cluster design” on page 34. For correct failover and recovery behavior, the broker is dependant on message resilience and message availability provided by the underlying transport layer discussed in 3.3, “WebSphere MQ availability” on page 26, and on event and transaction information provided by a highly available database. Independent from the cluster solution, the InterChange Server should be configured to provide a reliable service. That means, its database connectivity must not exceed the
database’s capacity; it must not consume more than the available memory, and its recover behavior must be configured correctly. All configuration parameters mentioned in this section are documented and explained in the IBM WebSphere InterChange Server System Administration Guide V4.2.2 and the IBM WebSphere InterChange Server Implementation Guide for WebSphere InterChange Server V4.2.2 respectively. Please refer to the product documentation for complete details.

**Database connectivity**

Configure InterChange Server for optimal database resource allocation:

1. **MAX_CONNECTIONS**: Specifies how many simultaneous connections InterChange Server can establish with the DBMS server. Ensure that this number is high to allow processing such as MAX_CONNECTIONS=100. The best way is not to specify any limit if the DBMS allows for that. InterChange Server times out idle connections after 2 minutes, or the configured value of the IDLE_TIMEOUT parameter.

2. **MAX_CONNECTION_POOLS**: Specify the number of databases you are using. The minimum value is 4 (EventDB, RepositoryDB, TransactionDB and RelationshipDB).

3. **MAX_DEADLOCK_RETRY_COUNT**: Allow InterChange Server to wait for possible database deadlocks to be resolved by setting this parameter (and the parameter DEADLOCK_RETRY_INTERVAL) to a value higher than 0. If the value is set to 0 and a deadlock occurs, the transaction will not be retried. This can cause the InterChange Server to shut down. In short deadlocks can occur while concurrently executing collaboration groups that call each other. For details refer to the IBM WebSphere InterChange Server Implementation Guide for WebSphere InterChange Server V4.2.2.

**Memory management**

InterChange Server prevents itself from using up too much memory and shutting down with an OutOfMemory Exception. It does this by checking the available memory and pausing the polling of adapters should InterChange Server become short of memory. Refer to section “Controlling Server Memory Usage” in the IBM WebSphere InterChange Server Implementation Guide for WebSphere InterChange Server V4.2.2 for details:

1. **CW_MEMORY_MAX**: Set this value in the InterChange Server start script to the maximum amount of memory available for the InterChange Server JVM. It needs to accommodate your largest business objects. It also needs to accommodate other processes (such as WebSphere Business Integration Adapters) running in the same memory.

2. **FLOW CONTROL** (component based): Configure MaxCapacity on controller or collaborations where you expect the most memory consumption.
InterChange Server controls whether or not more events can be added to the internal event queues for adapter controllers or collaborations. The message size times the number configured for MaxCapacity will be held in memory, for each component.

3. MEMORY_CHECK_SLEEP (system wide): Set this value higher than 0 in the InterChange Server configuration file. In that case, InterChange Server puts polling adapters into paused state for the specified amount of time in minutes. For example, set this value to 5 to prevent the agents from polling for five minutes.

4. MEMORY_UPPER_THRESHOLD_PCT (system wide): Specify the percentage of heap usage for the JVM in the InterChange Server configuration file. When InterChange Server reaches this specified heap usage, it pauses the polling adapters. The authors suggest 95 as a reasonable value.

**Disk space management**
Configure your log and trace file management in the Interchange System Configuration, so that the log and trace files do not grow over the allowed amount of disk space on your shared log directory.

**Recover processes from unplanned failover/restart**
InterChange Server has the built-in ability to recover from an uncontrolled system shutdown. The configuration properties that control the recovery behavior are referenced in the *IBM WebSphere InterChange Server Implementation Guide for WebSphere InterChange Server V4.2.2*. Be aware that they generally interfere with InterChange Server’s built-in event sequencing ability, and can cause a delayed restart. They are briefly described below; please refer to the *IBM WebSphere InterChange Server Implementation Guide* to fully understand the details and consequences:

1. Persist Service Call In Transit State: If you configure that property of a collaboration object to true, a collaboration that is shut down unexpectedly while performing a service call request to an adapter, will fail that event with status In Transit. The event will not be recovered automatically on system restart, and can therefore corrupt event sequencing. The system administrator can decide whether or not to resubmit the event. It could have been the case that the system shut down just before it received the Create Success message from the agent. In that case the event is discarded and not resubmitted.

2. Recovery Mode: A collaboration that is not dependent on event sequencing should be configured to Deferred. In that case, all events In Progress during the system shut down will not be automatically recovered. They will be failed
with status Deferred. They can be manually resubmitted or discarded after manual examination. This speeds up the restart time considerably.

3.5 Message Broker availability

As with InterChange Server, the availability of Message Broker is built upon the hardware cluster design discussed in 3.6, “HA cluster design” on page 34. Note that the broker is dependent on the message resilience and the message availability provided by the underlying transport layer for its own failover and recovery capabilities. Refer to 3.3, “WebSphere MQ availability” on page 26 for a discussion of MQ availability.

If the broker is processing a high number of persistent messages, it is advisable to perform some tuning of the LogBufferPages setting of the brokers queue manager. This setting affects the amount of memory allocated to buffer records for writing, specifying the size of the buffers in units of 4 KB pages. Larger buffers lead to higher throughput, especially for larger messages.

The database connectivity can be improved by making changes as discussed in “Database connectivity” on page 31. There are also some additional settings that should be assessed:

1. MAXAGENTS: This parameter indicates the maximum number of database manager agents, whether coordinator agents or subagents are available at any given time to accept application requests. If you want to limit the number of coordinating agents, use the max_coordagents parameter.

2. MAXAPPLS: This parameter specifies the maximum number of concurrent applications that can be connected (both local and remote) to a database. Since each application that attaches to a database causes some private memory to be allocated, allowing a larger number of concurrent applications will potentially use more memory.

3. On AIX, you can only make 10 database connections per process by default (using shared memory connections), so you must update your instance to use TCP/IP connections in loopback mode.

Further details on database settings can be found in the DB2 Information Center:

http://publib.boulder.ibm.com/infocenter/db2help/index.jsp
3.6 HA cluster design

By using HA clustering software along with the WebSphere Business Integration portfolio it is possible to enhance the availability of the services that the portfolio is providing. With a suitably configured HA cluster, it is possible for failures of power supplies, nodes, disks, disk controllers, networks, network adapters, queue manager processes, or other component processes to be detected and automatically trigger recovery procedures to bring the effected WebSphere Business Integration component back online as quickly as possible.

Although each vendor adapts their clustering technology according to their particular hardware architecture, there are many common procedures that are shared among the different clustering solutions. This is also true in the case of implementing MQ, Message Broker, and InterChange Server within a given clustering solution. For instance, no matter what clustering approach is being used, MQ, Message Broker, and InterChange Server should have their own product files located on local disks and data and log files resident on a shared storage device. The same rule applies to database management systems that are part of the WebSphere Business Integration environment such as the Message Broker database or InterChange Server’s repository database. The setup described is required in order to facilitate takeover of resources by the acquiring node; this will allow for the WebSphere Business Integration services to be restored.

All clustering approaches provide different mechanisms to define resources that are going to be managed by the HA clustering software. Examples of these resources are: A particular queue manager, a database instance, all broker processes (including execution groups), or all user name server processes. Within the HA clustering environment, related resources are grouped together into resource groups².

Resource groups

Resource groups are the unit of failover in HACMP For example, a resource group is migrated as a unit when failover occurs. A resource group is a unit that contains dependent resources: IP labels, applications, file systems, and volume groups. A resource group has attributes that define how it behaves during node startup, resource group failover to another node in the case of application or node failure, and when the resource group will fallback to the reintegrating node. Ideally, a resource group will contain only the resources necessary to define an operational unit. This approach optimizes flexibility in failover, and minimizes disruption during a failure or planned maintenance.

² The term resource group is HACMP specific. For other equivalent terms, please see “Resource groups” on page 16.
The use of resource groups makes it possible to move related work from one system to another and subsequently increases those resources availability. In order to assure availability, the HA clustering software is constantly monitoring the resources that it has defined. Each cluster implementation checks for the health of the core resource processes such as the queue manager processes, broker processes, InterChange Server and adapter instances and DB2® core processes. Finally, it is important to notice that the use of HA clustering technology combined with WebSphere Business Integration will provide a highly available service to critical applications. If continuous availability is required, then the addition of the queue manager clustering feature to the HA clustering solution will aid in reaching that goal.

In a WebSphere MQ only environment, the minimum unit of failover is the queue manager. In a Message Broker environment, it is the broker, the underlying queue manager, and the broker's database if local. In an InterChange Server environment, the minimum unit of failover is the broker, the ORB, and the InterChange Server queue manager. Subsequent sections discuss whether or not WebSphere Business Integration Adapters should be part of the same resource group as the adapter's associated integration broker.

**Critical components**

When designing a WebSphere Business Integration HA cluster, the architect must decide which of these various middleware application components are *critical* and which are not.

A *critical* component triggers a failover when the component cannot be started or restarted successfully.

In considering the issue of critical versus non-critical components, the authors determined the following:

- Adapter agents should not be critical. If an application is down for a significant time period, a critical adapter agent would cause continual failover between nodes, and generally you do not want one application/adapter to bring down entire an WebSphere Business Integration system.

- InterChange Server queue manager and InterChange Server may or may not be critical. A system administrator would have to investigate failure instead of automatically failing over since human intervention may be required to resolve the problem. Situations where a failover would be beneficial are when the RAM or CPU of the primary node are not working properly, or when some files on the local disk have been corrupted.

If InterChange Server is configured as a critical application, the Application Server for the resource group containing the InterChange Server should try to restart the broker on the same machine prior to failing over the resource group.
• Message Broker queue manager and Message Broker may or may not be critical. A system administrator would have to investigate failure instead of automatically failing over, since human intervention may be required to resolve problem. Situations where a failover would be beneficial are when the RAM or CPU of the primary node are not working properly, or some files on the local disk have become corrupted.

If Message Broker is configured as a critical application, the Application Server for the resource group containing the Message Broker should try to restart the broker on the same machine prior to failing over the resource group.

**Active/Active versus Active/Passive HA clustering environments**

*Active/Active* is a description of an HA cluster in which both nodes are actively used during normal operations. Critical work can be taken over by any of the nodes in the event of a node failure. Takeover configurations make use of hardware resources more efficiently than standby configurations since there is no idle processor.

*Active/Passive* is a description of a cluster in which only one node is used during normal operations.

• **InterChange Server**

The Interchange Server supports an Active/Active configuration only in the sense that other applications (such as database or Message Broker) are running on the second node. InterChange Server also supports static workload segmentation, where different types of events are processed on the different InterChange Servers.
WebSphere Business Integration Adapters

WebSphere Business Integration Adapters are not able to run actively on both nodes if they would be connecting to the same application and subscribing to the same business object. Starting with IBM WebSphere Business Integration Adapters Version 2.4, the adapters must poll for different business objects. Otherwise, it is not guaranteed that an event will be picked up only once. Also, if using WebSphere Business Integration Adapters in conjunction with InterChange Server, a unique registration of the adapter is required by the ORB.

Message Broker

The Message Broker supports both Active/Passive and Active/Active operation. Active/Active, also referred to as mutual takeover, is a more advanced configuration in which all Message Broker nodes are actively processing. Such a configuration allows for an immediate takeover giving an always-on processing environment.

There are, however, several implications of an Active/Active environment that one must keep in mind. For one thing, there is no guaranteed message processing order. In an Active/Active scenario, it is not a given that a single-threaded flow will process messages in FIFO order. It is possible that two messages will enter two instances of the flow in a particular order but that the second message to arrive will be processed before the first.

Also, in Active/Active processing there is the possibility of marooned messages. If a message is placed onto the input queue of a flow and the broker becomes unavailable, that message will not be processed until the broker returns online. A way around this scenario will be to have either a third

Restriction: InterChange Server does not support an Active/Active configuration where an instance of InterChange Server is running on each node and the surviving InterChange Server instance dynamically takes over the load of the InterChange Server instance that was running on the failed node. The main reason is that two different InterChange Server instances need to have different names and one adapter can only be connected to one InterChange Server which is uniquely identified by name.

Also, while it is theoretically possible to run independent InterChange Servers supporting the same business objects, and subscribed to the same collaborations with adapters connected to the same applications, this is not a viable configuration. Because it is not supported that two InterChange Servers connect to the same database, event sequencing between the two servers is not possible.
failover node in Active/Active/Passive or failover the failed broker's resource group onto the running node, and run it for an appropriate amount of time to allow all input messages to be cleared before shutting it down.

With either of the takeover configurations, it is important to consider the peak load that may be placed on any node. Such a node must possess sufficient capacity to maintain an acceptable level of performance.

Both configurations are in Chapter 5, “Implementing the solution”.

### 3.6.1 Highly available WebSphere MQ

The smallest unit of failover for MQ is the queue manager. For flexibility, each queue manager has its own resource group. It follows that the optimal configuration of resources in HA MQ is to place each queue manager in a separate resource group along with the other resources upon which it depends.

It is possible to put multiple queue managers into a resource group. If you do so, they will all failover as a unit. Therefore, in this case a problem confined to a single queue manager will cause all of the queue managers in the resource group to failover. This causes the unnecessary disruption of the other queue managers.

HACMP/ES (IBM High Availability Cluster Multi-Processing for AIX Enhanced Scalability) users who wish to use application monitoring should note the restriction that only one Application Server in a resource group can be monitored. If multiple queue managers are put into a single resource group and monitoring is needed for them all, the user is required to provide a monitor capable of monitoring several queue managers.

It is assumed that if mirroring or RAID is used to provide protection from disk failures, then references in the following text to physical disks should be taken to mean the disk or group of disks, which is being used to store the data that is being described.

A queue manager that is to be used in an HACMP cluster needs to have its logs and data on shared disks, so that they can be accessed by a surviving node in the event of a node failure. A node running a queue manager must also maintain a number of files on internal disks. These files include files that relate to all queue managers on the node, such as /var/mqm/mqs.ini, and queue manager specific files that are used to generate internal control information. Files related to a queue manager are therefore divided between internal and shared disks.

Regarding the queue manager files that are stored on shared disk it would, in principle, be possible to use a single shared disk for all the recovery data (logs and data) related to a queue manager. However, for optimal performance, it is
recommended to practice placing logs and data in separate file systems such that they can be separately tuned for disk I/O. The example scripts included in the WebSphere MQ SupportPac™, *MC63: WebSphere MQ for AIX - Implementing with HACMP* use separate file systems.

If the HACMP cluster will contain multiple queue managers, then depending on your chosen cluster configuration, two or more queue managers may at times need to run on the same node. To provide correct routing of MQ channel traffic to the queue managers, you must use a different TCP/IP port number for each queue manager. The default MQ port is 1414. It is common practice to use a range of port numbers immediately above 1414 for additional queue managers. Note that whatever port number you assign to a queue manager, that port needs to be consistently defined on all cluster nodes that may host the queue manager, and all channels to that queue manager need to refer to the port.

When configuring a listener for incoming MQ connections, you can choose between `inetd` and `runmq1sr`. If you use `inetd` then you do not need to perform any start or stop of the listener from within the HA cluster scripts. If you use `runmq1sr` then you should configure a user exit. Refer to the documentation in MQ SupportPac, *MC63: WebSphere MQ for AIX - Implementing with HACMP* for information on configuring user exits.

To conclude, the advantages of an HA MQ solution are features such as automatic queue manager failure detection and failover, as well as recovery from other points of failure (operating systems errors, a failed network card, application errors, database errors, natural disasters, and so on). It is important to note that these features are not available from a pure queue manager clustering solution.

**Important:** WebSphere MQ SupportPac MC63 and the related SupportPacs for other platforms handle the setup of ipc keyfile directories. It is essential that this be done correctly if more than one queue manager may run on a single node. The reader is strongly encouraged to use the facilities provided in the SupportPac that is appropriate to their platform and choice of HA clustering software rather than trying to do the HA MQ setup by hand.

### 3.6.2 Highly available Message Broker

For Message Broker the smallest unit of failover is the broker together with the associated queue manager, and the broker database if it is running locally. It is recommended that each Message Broker has its own resource group, sometimes this is not possible if the brokers all use the same database instance. If a User Name Server is being used, it should be treated similar to a broker, and should be in its own resource group.
When considering the Configuration Manager's availability, there are different opinions. Some opinions sustain that as the component does not perform critical work in a runtime environment, and its database is not vital to the correct execution of the broker service, it is not necessary to make this component highly available. Another opinion is that because the Configuration Manager provides the only administration interface into the broker and the resources it owns (execution groups and message flows), it should be included in the failover planning.

Note that because the Configuration Manager is currently only supported on the Microsoft Windows® platform, it cannot be part of the HACMP based configuration. To make the Configuration Manager highly available, it must be placed under a suitable Microsoft Windows based HA solution such as Microsoft Cluster Server (MSCS). An alternative is to have a backup plan. It is possible to restore the Configuration Manager onto another machine if:

- The Configuration Manager queue manager on the new machine keeps the same name.
- You have an up-to-date backup of the Configuration Manager database.

In the scenario above, you can restore the Configuration Manager's database into the new Configuration Manager, and it would restore the full configuration from the original machine.

### 3.6.3 Highly available InterChange Server

For InterChange Server, the smallest unit of failover is InterChange Server, ORB, and the InterChange Server queue manager. For InterChange Server 4.2.2 to be HA, the ORB must be included as part of the InterChange Server HA cluster resource group, because as stated in 3.2, “WebSphere Business Integration component dependencies” on page 22, it is not possible to start InterChange Server without restarting the ORB. It is not possible to run the ORB name servers on both nodes at the same time, because they do not communicate with one another. So the HA cluster must take responsibility for ensuring that the ORB Name Server is up and running on the InterChange Server machine.

InterChange Server and the ORB also share one file called $CxCosNameRepos.ior$ and therefore need access to the same volume group. With InterChange Server 4.2.2.3, the ORB name server is persistent. If the ORB is restarted it reads the previous inter orb registrations from a file, which should be located on the shared storage device. In that case it is still necessary that the ORB is running when InterChange Server is started or restarted, but neither InterChange Server nor InterChange Server WebSphere Business Integration Adapters need to be restarted if the ORB is restarted. Restarting the InterChange
Server queue manager requires a restart of InterChange Server, and therefore they have to be in the same resource group. The adapter agents can connect to a separate queue manager, and that way they are not dependent on the InterChange Server resource group. However, there are reasons why you might want to add the InterChange Server adapters to the InterChange Server resource group.

**InterChange Server and WebSphere Business Integration Adapters in one resource group**

- Most failover scenarios will be caused by a hardware failure, and in that case all effected resource groups will have to failover.
- InterChange Server and InterChange Server adapters follow standard installation in the same directory.
- Separate resource groups cause administration overhead:
  - If adapters and the InterChange Server are in separate resource groups, the agents need to be installed as remote agents on the local machine. With JMS as transport, the adapter agent connects to a different queue manager than the controller. The additional administration might not be worth the effort, if there are only a few adapters.
  - The CosNameRepos.ior file needs to be persisted on the shared drive.

**InterChange Server and WebSphere Business Integration Adapters in different resource groups**

- If InterChange Server and ORB failover, the adapters can reside on the primary node even if the InterChange Server queue manager and InterChange Server failover.
- Failing over InterChange Server and adapters although only InterChange Server needs to failover, increases the failover time unnecessarily.
- If only InterChange Server needs to failover, the adapter agents are connected to an independent queue manager and can continue processing without interruption and need for recovery. That means different resource groups provide a more reliable solution.
- The adapters have to be installed according to remote agent technology. That means that the adapter installation has to be independent from the InterChange Server installation and, that the adapter log and configuration files need to be save in a different volume group than is used for the InterChange Server log and configuration files. The adapter agent queue manager is not the same as the InterChange Server queue manager.
The ior reference is written by InterChange Server during start up and is read by the ORB in the case of a restart. Therefore, in order to allow an ORB restart or failover without also requiring that the adapter agents be restarted, the ior reference must be located on the shared drive.

**Note:** Remote agent technology seems to be the better choice, mainly because it maintains the provided independence of the components. During the residency we show the high availability of this setup in cases of unavailability of InterChange Server, ORB or resource group failover; see 7.3.1, “InterChange Server resource group with local adapters” on page 106.

### 3.6.4 A few possible HA configurations

- **Setup1**: InterChange Server and InterChange Server adapters are installed on one node in one resource group with the ORB and queue manager. Message Broker and Message Broker adapters are installed on the second node, also in one resource group. The nodes are configured for mutual takeover.

- **Setup2**: InterChange Server is installed on one node in one resource group along with the ORB and the InterChange Server queue manager. Message Broker is installed on the other node in one resource group along with the Message Broker queue manager. All WebSphere Business Integration Adapters are installed on a third node along with the adapter’s queue manager. The nodes are configured for mutual takeover. The WebSphere Business Integration Adapters are part of the application’s HA solution.

- **Setup3**: InterChange Server is installed on one node in one resource group along with the ORB and the InterChange Server queue manager. Message Broker is installed on the other node in a resource group along with the Message Broker queue manager. The nodes are configured for mutual takeover. All WebSphere Business Integration Adapters are installed on third party application server boxes. The WebSphere Business Integration Adapters are not part of the application’s HA solution. Therefore this sample will not be further discussed in this book.

Chapter 5, “Implementing the solution” provides a sample implementation of local and remote WebSphere Business Integration Adapters, which combines Setup1 and Setup2.
3.6.5 Setup

Two nodes are included in this HA scenario. The WebSphere Business Integration Adapters run locally on the same node as the integration broker with which they communicate. One node hosts the InterChange Server resource group including the InterChange Server WebSphere Business Integration Adapters. The WebSphere Business Integration Adapters and InterChange Server share the same queue manager. The second node hosts the Message Broker resource group including the Message Broker WebSphere Business Integration Adapters. Again, the broker and adapter share the same queue manager. The nodes are configured for mutual take over. See Figure 3-3.

**Rationale**

Reasons for installing InterChange Server and InterChange Server adapters on one machine are:

- The application administrator does not allow another application (WebSphere Business Integration adapter) to be run on the application server box, or
- Adapters shall be installed and maintained centrally. They are not installed on a third node because the administrative overhead of installing remote adapter agents is higher than the performance gain. A small number of adapter agents or small message sizes suggest that brokers and adapter agents can aptly share one node.
3.6.6 Setup2

Three nodes are included in this scenario. The WebSphere Business Integration Adapters run on a separate node from both of the integration brokers. It is a general guideline that the WebSphere Business Integration Adapters should be installed as closely as possible to the application to which they connect. This minimizes the potential effects of network latency that could be encountered due to the conversational nature of retrieving and updating a business object versus the single call to send the business object between the controller and the adapter agent. One node hosts the InterChange Server resource group. The second
node hosts the Message Broker resource group. All adapters are installed on a third node and share the same queue manager among them. The nodes are configured for mutual take over. See Figure 3-4.

**Rationale**
This setup is favored if:

- The application administrator does not allow another application (WebSphere Business Integration Adapter) to run on the application server box, or
- Adapters will be installed and maintained from a central point, and will be included in the WebSphere Business Integration HA scenario, but for load balancing, they should not share the hardware with the integration brokers.

**Figure 3-4  WBI cluster: One RG for WICS, one for WBIMB and one for adapters**

**Resource Group Layout**
ICS_RG (Priority: Node1, Node2) - ICS, QManager1, ORB, IPAddress1

MB_RG (Priority: Node2, Node3) - WBIMB, QManager2, IPAddress2

AD_RG (Priority: Node3, Node1) - JText/JDBC/WBIMB Adapter, QManager3, IPAddress3
3.7 Capacity planning

In an HA cluster environment where under normal operations each node only runs one application, the nodes will be configured for mutual takeover. Therefore, the IT professional must consider that in a failover scenario, each node should be capable of supporting both applications. To gather the hardware requirements, one must consult the standard system installation guides of each application. However, there are certain requirements that are specific to an HA environment:

- The requirement is that you have at least two nodes. For example, you could have two IBM pSeries 670s, or you could have a single IBM pSeries 690. The reason that a single p690 will work is that it can be configured into multiple partitions and each partition can be a node. The pSeries 690 delivers true logical partitioning (LPAR). Each system can be divided into as many as 16 virtual servers, each with its own set of system resources such as processors, memory, and I/O.

- A shared disk system such as IBM's FASTT storage servers or IBM's 7131-405 SSA disk subsystems. Application data and log files are written to the shared drive frequently during normal operations. So, it is important to ensure that there is a fast connection to the shared storage from both nodes such as a fibre switch. And to increase availability, one may want to provide a redundant fibre connection from each node to the shared storage.

- A device over which the cluster can exchange heartbeats. This can be a public TCP/IP network, a private point-to-point network, or a shared disk.

- A public IP network to communicate with the outside world. Again, the IT professional must be aware that in the case of a failover, one node must be able to accommodate the work of both nodes. If you use IPAT through IP aliases, you have to provide as many different subnets as you have resource groups. If you use IPAT through IP replacement, every node needs to have as many NICs as the total of all resource groups in the cluster.

Refer to the *High Availability Cluster Multi-Processing for AIX Planning and Installation Guide Version 5.1*, SC23-4861 for details of how to select appropriate disk technologies and networks.

3.8 Maintenance planning

Maintaining an HA system means maintaining hardware and software changes without causing any unnecessary system outages. Any changes to a highly available production system should be done in the period when there is the lightest load on the system. Although WebSphere Business Integration can handle unintended shut downs while events are in progress, any maintenance
Chapter 3. Design considerations

should aim to shut down components when there are no events in progress. The reasons for this are that the recovery of events in progress is not always fully automatic and may require manual intervention and investigation. The second reason is that if the automatic recovery of an event occurs during component start up, the restart may take considerably longer than a restart without recovery requirements.

3.8.1 Administration tasks

For the production system, establish and document the following tasks. Clearly assign roles and responsibilities to each person as well as the back-up personnel:

- Back up and restore procedures for all WebSphere Business Integration components including databases whether they are located locally or remotely.
- Define recovery procedures that clearly outline any manual steps to be taken after a failure.
- Establish system and software migration and upgrade procedures (as outlined in 3.8.3, “Applying software upgrades and fixes” on page 48), and allocate the necessary roles and responsibilities to the designated people.
- Establish a regular maintenance schedule (such as establishing a particular maintenance window upfront).
- Define emergency maintenance procedures.

3.8.2 Adding and replacing hardware

In order to necessitate the shortest system outages on hardware changes, opt for hot-swapable hardware where ever possible. Some IBM pSeries servers allow for hot changes of logical partition configuration (if using LPAR) or adding new hardware components like memory or CPU to the cluster. For hardware changes that cannot be done while the system is running, follow these steps:

1. Pause event delivering adapters on node1.
2. Monitor the system and wait until all events are processed.
3. Switch off HA monitoring of all resource groups on node1.
4. On node1, stop all active applications through HA stop scripts.
5. Initiate a failover of resource group1 from node1 to node2.
6. Restart the effected application on node2 through HA start scripts.
7. Make the necessary hardware changes on node1.
8. Pause all event delivering adapters on node2.
9. Monitor the system and wait until all events are processed.
10. Fall back resource group1.
11. Initiate failover for resourcegroup2 and resourcegroup3 from node2 to node1.
12. Start all resource groups on node1.
13. Make the necessary hardware changes on node2.
14. Fallback resource groups to node2.

3.8.3 Applying software upgrades and fixes

Any fixes or upgrades that need to be applied to any of the WebSphere Business Integration components and require the WebSphere Business Integration component to be shut down should be done using the following sequence of steps.

**WebSphere Business Integration adapters**

These are the steps:
1. Apply the change to the component on the inactive node first, hereafter called node2.
2. Pause event delivery for all adapters belonging to that resource group on the node1. Here it is important to have the resource groups defined as small as possible.
3. Monitor the system and wait until all events are processed.
4. Switch off HA monitoring of the adapter relevant resource group on node1.
5. Stop the adapter relevant resource group on node1.
6. Initiate a failover for the adapter resource, group1, from node1 to node2.
7. Restart the application on node2 through HA start scripts.
8. Make the necessary changes on node1.
9. Fall back the adapter resource group, repeating steps 2 to 7.

**InterChange Server**

These are the steps:
1. Apply the change to the component on the inactive node first, hereafter called node2.
2. Pause all event delivering adapters in the system.
3. Monitor the system and wait until all events are processed.
4. Switch off HA monitoring of the InterChange Server resource group on node1.
5. Shut down the InterChange Server resource group using the HA stop script.
5. Initiate a failover for the InterChange Server resource group from node1 to node2.
6. Restart the application on node2 through HA start scripts.
7. Make the necessary changes on node1.
8. Fall back the InterChange Server resource group, repeating steps 2 to 7.

**Message Broker**
These are the steps:
1. Apply the change to the component on the inactive node first.
2. Adjust the restart attempts to 0 on the inactive node to minimize downtime due to a bad patch.
3. Initiate a failover for the Message Broker resource group from current node to failover node.
4. After a successful start of the Message Broker on the failover node, apply the necessary changes on original active node.
5. Fall back the Message Broker resource group to its default node if required.
6. Adjust restart attempts back to default value of the inactive node from step 2.

**WebSphere MQ**
These are the steps:
1. Apply the change to the component on the inactive node first.
2. Adjust the restart attempts to 0 on the inactive node to minimize downtime due to a bad patch.
3. Initiate a failover for the resource group containing that MQ queue manager.
4. After a successful start of MQ on the failover node, apply the necessary changes on original active node.
5. Fall back the resource group to its default node if required.
6. Adjust restart attempts back to original value from step 2 on the inactive node.

### 3.8.4 Deploying new interfaces

**InterChange Server**
In InterChange Server version 4.2.2.2, maps, collaborations, and relationships can be deployed at runtime without a restart of the InterChange Server. An adapter must be restarted if a map is deployed that requires new explicit associations.
If a new interface processes only business objects that are already supported by the relevant adapters, then it is not necessary to restart InterChange Server in order to deploy the new interface. However, if new business objects have been added, then the connectors that support these business objects will need to be redeploysed which requires a restart of the InterChange Server.

### 3.9 Performance

This section provides HA specific WebSphere Business Integration performance tuning guidelines. If you are concerned about the general performance of your WebSphere Business Integration system, please review the product manuals and SupportPacs for information.

HA specific performance subjects are listed in the following sections.

#### 3.9.1 Minimize unmonitored downtime

- Detect application outages early by frequent monitoring.
- Provide efficient monitoring scripts that detect application outages quickly, such as checking on process IDs if possible.

#### 3.9.2 Minimize failover time

- IPAT through IP aliases is faster than IPAT through IP replacement. It is suggested that you test the actual time difference for both possibilities on your system before choosing one method or the other for your production system.
- Resource groups should be defined as small as possible in order to avoid increasing shut down and restart time unnecessarily.
- Minimize the restart time for the components:
  - InterChange Server: Configure collaborations for deferred recovery if your business process or administrative environment allows for it.
  - InterChange Server: Delete all unnecessary components from your repository. The difference between starting InterChange Server with a big repository, and starting InterChange Server with a small repository can reach up to ten minutes.
  - InterChange Server: In order to keep the size of the repository down, consider implementing a second InterChange Server instance (separate server name, separate resource group, different interfaces and adapters).
Message Broker: When working with a large number of message flows, spread them across a number of brokers to decrease the broker startup time.

Follow the standard guidelines laid out in the product manuals and performance related SupportPacs to tune the Java JVM for InterChange Server and WebSphere Business Integration Adapters.

Follow the standard guidelines laid out in the product manuals and performance related SupportPacs to tune the InterChange Server, Message Broker, and repository database.

Start as many components in parallel as possible.

### 3.9.3 General performance under HA

Data and logs are held on a shared drive. In most cases they are frequently accessed. Therefore:

- Provide a fast shared disk subsystem like the IBM FAStT Storage Server or Serial Storage Architecture (SSA) technology with Fast-Write Cache.
- Provide a high-speed connection from the nodes to the shared disks such as Ultra320 and Fibre Channel.
- Use the minimum number of disk groups required.
- Dynamic Processor Sparing capability can help maintain performance and improve system availability when running in failover mode. (Starting with AIX 5L V5.2, the Dynamic Processor Sparing capability is offered on pSeries servers with pSeries Capacity Upgrade on Demand.)
- The IBM @server pSeries Dynamic Logical Partitioning (DLPAR) feature in AIX Version 5.2 will allow you to provide your failed-over production partition with an additional processor, which can be taken from a less business critical partition.
Part 2

Developing and implementing a solution
Business scenario

High availability solutions present unique challenges for the IT professional. In the following chapter we discuss a typical business scenario and one possible solution architecture for a WebSphere Business Integration HA solution.
4.1 Business challenge

The primary reasons behind businesses wanting to implement high availability solutions result from the need for core applications that are critical to the business to work seamlessly, and continue to operate in spite of failures from either the system where those applications run or the application itself. The key for designing a highly available solution that ensures business continuity is to approach the issue of WebSphere Business Integration availability as another critical service component to the application.

4.2 Sample scenario

The objective of the sample scenario is to demonstrate how the WebSphere Business Integration middleware can be configured to withstand different types of failures including hardware, software, and end application failures. The following sections describe one possible solution. The scenario is built using a combination of WebSphere Business Integration products, which are used by customers worldwide.

4.2.1 Scenario design

The example scenario is illustrated in Figure 4-1.

These are the steps:

1. The purchasing application receives a new order through the Web. The WebSphere Business Integration Adapter for JDBC (the JDBC adapter) picks up that entry and starts a collaboration in the Interchange Server.
2. An Interchange Server collaboration bound to that trigger is started and processes the entry.

3. Some of the information is inserted into the orders database using the JDBC adapter.

4. Another portion of the data from the order is passed to the Interchange Server queue manager. The data is then passed to the Message Broker using the WebSphere Business Integration Adapter for WebSphere MQ Integrator Broker (the Message Broker adapter).

5. The data is received by the Message Broker’s queue manager, and this triggers a flow within the Message Broker itself. As part of its processing, the message flow saves part of the information to a database.

6. The message is received by the WebSphere Business Integration Adapter for JText (the JText adapter) and is placed onto the local file system thus ending our flow.

### 4.2.2 Description of the solution

The high availability solution is based on an AIX implementation running in an Active / Active cluster. The Active / Active cluster was chosen because it is the most cost-effective configuration. This mutual-takeover configuration allows for both machines to be fully functional with each machine being ready to takeover the workload of its partner in the event of a failure. Figure 4-2 gives the cluster setup for our solution.
Figure 4-2 Cluster setup

The basic solution is implemented under three different takeover configurations. This allows the authors the opportunity to achieve and explore a number of different HA capabilities.

Figure 4-3 shows an Active/Passive Message Broker with an InterChange Server setup.
The resource groups are defined as follows:

On Node 1:

- **WICS_RG**
  - Interchange Server
  - MQ queue manager for the Interchange Server
  - ORB
  - Adapters (JDBC x 2, WMQMB)

For this scenario, we have defined the adapter agents as non-critical, for example, they will not cause a node failover event if they stop running. The rationale behind this is that the agents will fail if your end application goes down, but that does not construe a middleware failure. The HA software is configured such that it will attempt to restart the adapter agent.

Also, we have decided that a failure of the ORB does not require a full node failover as the ORB is persistent. The scripts will attempt to restart the ORB three times before failing it over. Also, note that the adapters are independent of the InterChange Server resource group specifically the ORB.

On node 2:

- **MB_RG**
  - WebSphere Business Integration Message Broker
  - MQ queue manager for the Message Broker
The Message Broker failover behavior is similar to that of the Interchange Server resource group. If either the queue manager or the Message Broker fails, then the scripts will attempt to restart them. If unsuccessful, the resource group will failover.

Note that the JText adapter is in its own resource group. This design decision was made because the JText adapter will continue to function even if the Message Broker fails over. This setup simulates an adapter that is remote to the broker. This configuration is appropriate to all supported brokers: Interchange Server, Message Broker, or WebSphere Application Server.

**Attention:** In the current setup, we have the capability to restart the adapters within the resource group infinitely with the limitation that if the adapter cannot start, there is no notification within HACMP.

An alternate setup is to have the adapter agent in its own resource group, which is dependant on the broker resource group. This allows for notification through HACMP of an adapter failure, which requires a manual restart.

Another configuration that we explored was constructed by taking the Active/Passive configuration, and adding an additional resource group containing a second Message Broker and the associated queue manager. This gives an Active / Active Message Broker takeover scenario. See Figure 4-4. For this scenario, the Message Broker resource groups were configured for notification, therefore there is no failover.
Another configuration was created by taking the Active / Active Message Broker with InterChange Server setup, and adding an additional resource group containing the second JDBC adapter and associated queue manager. This resource group has similar HACMP behavior to the Message Broker adapter resource group where it infinitely attempts to restart the adapter. This gives the Remote InterChange Server adapter setup. See Figure 4-5.
4.2.3 Data store

For both the Message Broker and Interchange Server, the database repository chosen was IBM DB2 Universal Database™. This database resides on a third AIX box that is not part of the Active/Active middleware cluster.

Since this redbook focuses solely on implementing a highly available WebSphere Business Integration scenario, it is out of scope for this redbook to discuss the clustering and HA solutions available for all supported databases. Also, most IT shops have already defined database storage, backup, and recovery procedures. Just remember that as a WebSphere Business Integration implementer, you must be cognizant of the fact that for a fully functional high availability solution, the database repository must also be highly available.

For purposes of this redbook, the authors assume that the required databases are always available.
4.2.4 WebSphere MQ connectivity

MQ is the transport mechanism used for communication between the pieces of a WebSphere Business Integration implementation. MQ messaging provides a reliable, robust multi-platform messaging middleware layer. The MQ high availability setup will be described in more detail throughout Chapter 5, “Implementing the solution”.
Implementing the solution

This chapter details the specifics of implementing the scenario put forward in Chapter 4, “Business scenario” on page 55.
5.1 Cluster layout

The cluster is configured with two nodes and altogether three resource groups. One for InterChange Server, one for Message Broker, and one for those WebSphere Business Integration Adapters using Message Broker as an integration broker. See Figure 5-1.
A volume group has been created for each resource group. It contains runtime data and log files for the specific resource group, and are located on the FASTT200 shared disk system. In Figure 5-2 they are depicted with their logical volume names and mount points.
5.2 Solution architecture

In the first example configuration, the solution architecture is comprised of two HACMP nodes with three resource groups between them. The WebSphere InterChange Server RG is active on node 1 of the HA cluster. This RG includes:

- InterChange Server
- The ORB
- The queue manager associated with InterChange Server
- One WebSphere Business Integration Adapter for JDBC
- One WebSphere Business Integration Adapter for WebSphere MQ Integrator Broker

One WebSphere Business Integration Adapter for JDBC is installed remotely on a database application server.
The Message Broker and WebSphere Business Integration Adapter for JText RGs are active on node 2 of the HA cluster. The Message Broker RG includes:

- Message Broker
- The queue manager associated with Message Broker

And, the WebSphere Business Integration Adapter for JText RG includes:

- WebSphere Business Integration Adapter for JText
- The queue manager associated with WebSphere Business Integration Adapter for JText

The WebSphere Business Integration Adapter for JText receives its input from Message Broker.

In the second example configuration, the solution architecture is comprised of two HACMP nodes with six resource groups between them. The following resource groups are active on node 1 of the HA cluster.

- WebSphere InterChange Server RG which includes:
  - InterChange Server
  - The ORB
  - The queue manager associated with InterChange Server

- WebSphere InterChange Server adapters RG (such as those WebSphere Business Integration Adapters that use InterChange Server as an integration broker):
  - One WebSphere Business Integration Adapter for JDBC
  - One WebSphere Business Integration Adapter for WebSphere MQ Integrator Broker
  - The queue manager associated with WebSphere InterChange Server adapters

- Message Broker RG which includes:
  - Message Broker
  - The queue manager associated with Message Broker

- WebSphere Business Integration Adapter for JText RG which includes:
  - WebSphere Business Integration Adapter for JText
  - The queue manager associated with WebSphere Business Integration Adapter for JText

One WebSphere Business Integration Adapter for JDBC is installed remotely on a database application server.

The Message Broker and WebSphere Business Integration Adapter for JText RGs are active on node 2 of the HA cluster. The Message Broker RG includes:
- Message Broker
- The queue manager associated with Message Broker

And, the WebSphere Business Integration Adapter for JText RG includes:
- WebSphere Business Integration Adapter for JText
- The queue manager associated with WebSphere Business Integration Adapter for JText

The WebSphere Business Integration Adapter for JText receives its input from Message Broker.
**Note:** The Active/Active configuration for the Message Broker (the Message Broker is running actively on both HACMP nodes) is supported using MQ queue manager clustering. The Message Broker input queue (which is the Message Broker adapter’s output queue) is a cluster queue that forms part of a queue manager cluster consisting of the InterChange Server queue manager, the Message Broker queue managers, and the WebSphere Business Integration Adapter for WebSphere MQ Integrator Broker queue manager. The Message Broker output queue is a cluster queue in the same cluster, and is the input queue for the WebSphere Business Integration Adapter for JText.

See Figure 5-3.

---

**Figure 5-3  WBI Solution architecture**
5.3 Preparing components for HA

Every WebSphere Business Integration component needs to be prepared separately to operate in a high availability environment. The following sections provide detailed steps for preparing each component.

5.3.1 Initial environment

As a business begins to grow an enterprise-wide middleware infrastructure, it is not unusual for a non-HA implementation of middleware services to be put into place. Then, as more projects and groups take advantage of these services, the business recognizes that these applications and the middleware services that support them are critical to their operations. And, the IT department is tasked with making the middleware services highly available. In this vein, the authors began with a fully functional non-HA implementation of the scenario described in Chapter 4, “Business scenario”, and made that existing implementation highly available.

5.3.2 Configuring WebSphere MQ

As a result of the existing non-HA implementation, all of the queue managers and their associated queues and channels, etc. already existed and needed to be converted to an HA implementation. The WebSphere MQ SupportPac, MC63: WebSphere MQ for AIX - Implementing with HACMP provides a set of scripts for configuring MQ under HA assuming that the desired queue manager does not currently exist. No scripts are provided for converting existing queue managers.

This subsection details the steps required to convert existing queue managers. The scripts provided in the WebSphere MQ SupportPac, MC63: WebSphere MQ for AIX - Implementing with HACMP, were used as a starting point. Specifically, additional steps were needed beyond the functionality provided in the hacrtmqm script so that existing queue manager directories were moved to the correct locations.

Manually moving queue managers to a shared disk for HA

For each existing queue manager ($qmgr$), the following steps were completed. All commands, unless stated otherwise, were run as root. $messagedqmgr$ is equal to the representation of the queue manager under the /var/mqm/qmgrs directory. If any periods “.” are used in the name they are represented by an exclamation pont “!” such as ics422.queue.manager would be ics422!queue!manager.

1. As user mqm stop the $qmgr$.
2. Create a volume group per $qmgr$ to store all the queue manager data, and mount this onto the shared disk. /MQHA/ this must be owned by user mqm.
3. Untar WebSphere MQ SupportPac, MC63 into a bin directory under the volume group mount point:

   tar -xvf mc63.tar /MQHA/bin

4. Create /MQHA/$qmgr/data on the shared disk. This is where the queue manager files will reside. /MQHA is the top level directory used in the example scripts.

   mkdir /MQHA/$qmgr/data/qmgrs/$messedqmgr

5. Create /MQHA/$qmgr/log on shared disk. This is where the queue manager log directory will reside.

   mkdir /MQHA/$qmgr/log/$messedqmgr

6. Give ownership of all of the created directories to user mqm.

   chown -R mqm:mqm /MQHA/$qmgr/

7. Move all the $qmgr data onto the shared disk. Put the files into a folder with the same name as that listed under the /var/mqm/qmgrs directory such as ics422.queue.manager would be ics422!queue!manager.

   mv /var/mqm/qmgrs/$messedqmgr/* /MQHA/$qmgr/data/qmgrs/$messedqmgr/

8. Remove the (empty) local $qmgr data directory and create a link from the local /var/mqm/qmgrs directory to the new location of the $qmgr data.

   rmdir /var/mqm/qmgrs/$messedqmgr/
   ln -s /MQHA/$qmgr/data/qmgrs/$messedqmgr /var/mqm/qmgrs/$messedqmgr

9. Move all the existing logs to the shared disk. Put the files into a folder with the same name as that listed under the /var/mqm/log directory.

   mv /var/mqm/log/$messedqmgr/* /MQHA/$qmgr/log/$messedqmgr/

10. Edit the qm.ini for the $messedqmgr to change the log location for the queue manager. This file will now be located on the shared disk under the /MQHA/$qmgr/data/qmgrs/$messedqmgr directory.

    Log:
    LogPrimaryFiles=3
    LogSecondaryFiles=2
    LogFilePages=1024
    LogType=CIRCULAR
    LogBufferPages=0
    LogPath=/MQHA/MB_QM/log/$messedqmgr/
    LogWriteIntegrity=TripleWrite

11. Edit the mqs.ini file and remove the reference to the $qmgr.

12. Edit the /MQHA/bin/halinkmqm script to match your volume group setup:

    apmon=/MQHA/$qmgr/bin/hamqm_applmon.$1
    cat > $apmon << EOF
    #!/bin/ksh
    EOF
su mqm -c /MQHA/$qmgr/bin/hamqm_applmon_su $1
EOF
chmod +x $apmon

13. Run the MQ HA script:

```bash
halinkmqm MB_QM MB_QM /MQHA/MB_QM/data
```

After completing these steps, the queue manager data is configured in the same way as it would be after running the hacrtmqm script provided as part of MQ SupportPac MC63: WebSphere MQ for AIX - Implementing with HACMP. Now, the final step is to prepare the failover node.

14. For each $qmgr on the machine, which is the failover node for that $qmgr, run `halinkmqm` passing the script the queue manager name, the name of the queue manager under the `/var/mqm/qmgrs` directory, and the location of the shared data:

```bash
halinkmqm MB_QM MB_QM /MQHA/MB_QM/data
```

Ignore the file or directory in the path name does not exist error if the volume group is not mounted for the failover node.

After these steps the queue manager files are located in the correct location for HA.

**WebSphere MQ object configuration**

Both the queue manager and its queue objects may need configuration changes.

Set the default persistence of all queues to persistent. This will ensure that applications using the queue’s default setting for persistence will generate persistent messages that can be recovered after queue manager restart.

Considering the given environment such as the message arrival rate, event processing rate, component failure rates, general system activity, etc., set the queue manager trigger interval to a value that will effectively manage the generation of channel initiation trigger messages. Based on the rapid message arrival rate and the frequency with which failures were induced, 60,000 milliseconds was chosen as the trigger interval for purposes of this residency.

The queues used by the Message Broker MQInput nodes had their backout threshold set to 2. This setting is changed so that if the Message Broker is processing a message when it goes down, and the message will be re-processed. If this setting is not changed from the default of 0, when the Message Broker is brought back up any messages that were mid-way through processing will be passed to the MQInput nodes failure terminal or the SYSTEM.DEAD.LETTER.QUEUE.
5.3.3 Configuring remote adapters

The following steps are what is required to make the WebSphere Business Integration Adapters highly available. The configuration applies to all WebSphere Business Integration Adapters working with all supported brokers. These steps assume that the adapter has already been installed and configured in a non-HA environment, all configuration files exist in a local directory, and a volume group has been created on the shared disks called $MOUNT_POINT:

1. If the adapter’s queue manager is not already configured for failover, go to 5.3.2, “Configuring WebSphere MQ” on page 72 and follow all steps.

2. Create the repository and bin directories on the mount point and change ownership of the directory to the user that starts the adapter, in this case cwadmin:
   
   ```
   mkdir /$MOUNT_POINT/$ADAPTER_NAME/repository
   chown -R cwadmin:mqm /$MOUNT_POINT/$ADAPTER_NAME
   ```

3. Copy the existing files over to the new drive:
   
   ```
   mv /$ADAPTER_ROOT/repository/* /$MOUNT_POINT/$ADAPTER_NAME/repository
   ```

4. Remove the local directory:
   
   ```
   rmdir /$ADAPTER_ROOT/repository
   ```

5. Link the mount directories to the root disk:
   
   ```
   ln -s $MOUNT_POINT/$ADAPTER_NAME/repository /$ADAPTER_ROOT/repository/
   ```

6. Give ownership of linked directories to user cwadmin:
   
   ```
   chown -R cwadmin:mqm /$ADAPTER_ROOT/repository/
   ```

7. Install the adapter executables onto the failover node.

8. On the failover node delete the current repository and bin directories:
   
   ```
   rm -R /$ADAPTER_ROOT/repository
   ```

9. Repeat step 5 for the failover node.

5.3.4 Configuring Active/Passive Message Broker

As with the queue managers, the non-HA configuration had existing brokers that required filesystem modifications to make it suitable for control under HA. WebSphere Business Integration Broker SupportPac C61: Configuring MQSeries Integrator for AIX with HACMP provides a collection of scripts such as hamsicreatebroker, which is usually used to create the broker and that place the files in the correct locations on the shared disk. Instead of using this script, manual steps were taken to move the existing files onto the shared disk. These steps are detailed below. More information on the recommended layouts can be
found in WebSphere Business Integration Broker SupportPac C61: Configuring MQSeries Integrator for AIX with HACMP.

1. Make the registry file system on the shared disk, move over the registry data, and then link the data from its original locations:

   ```
   mkdir /MQHA/$qmgr/data/registry/$BROKER
   mv /var/mqsi/registry/$BROKER/* /MQHA/$qmgr/data/registry/$BROKER
   rmdir /var/mqsi/registry/$BROKER
   ln -fs /MQHA/$qmgr/data/registry/$BROKER /var/mqsi/registry/$BROKER
   ```

2. Make the broker's file system on the shared disk, move over the registry data, and then link the data from its original locations:

   ```
   mkdir /MQHA/$qmgr/data/brokers/$BROKER
   mv /var/mqsi/brokers/$BROKER/* /MQHA/$qmgr/data/brokers/$BROKER
   rmdir /var/mqsi/brokers/$BROKER
   ln -fs /MQHA/$qmgr/data/brokers/$BROKER /var/mqsi/brokers/$BROKER
   ```

3. Create the Application Monitor located in the HA scripts directory:

   ```
   apmon=/MQHA/bin/hamqsi_applmon.$BROKER
   cat > $apmon << EOF
   #!/bin/ksh
   /MQHA/bin/hamqsi_monitor_broker_as $BROKER $QM $DBINST $MQUSER
   EOF
   chmod +x $apmon
   ```

4. On each standby node, the files system needs to be prepared for the broker using the `hamqsiaddbrokerstandby` script:

   ```
   hamqsiaddbrokerstandby $BROKER $qmgr $mquser
   ```

5.3.5 Configuring Active/Active Message Broker

To configure an Active/Active failover scenario you must first create the queue managers for both brokers. It is recommended that you do not use the same name for the broker queue managers. If you require failover of the broker then follow the instructions for Section 5.3.2, “Configuring WebSphere MQ” on page 72 or run the `hacrtmqm` script provided as part of MQ SupportPac MC63: WebSphere MQ for AIX - Implementing with HACMP.

Once the queue managers has been created either follow the instructions in 5.3.4, “Configuring Active/Passive Message Broker” on page 75, or create the broker using the `hamqsi_createbroker` script provided as part of WebSphere Business Integration Broker SupportPac C61: Configuring MQSeries Integrator for AIX with HACMP.

Both of the brokers should have exactly the same message flows deployed. Do not specify the queue manager in the MQOutput nodes, as your brokers deal
with different queue managers. Specifying the queue manager in the MQOutput nodes also prevents applications from placing messages on a clustered queue.

Once the brokers have been set up, the queue managers and local queues need to be configured for the clusters. (Please refer to *WebSphere MQ Queue Manager Clusters*, SC34-6061, for information.) For this scenario we created two clusters where the repositories for both were held by the Message Broker’s queue managers.

The IN.Q is a local queue on both broker queue managers, which are shared in the cluster allowing the InterChange Server to dynamically bind to either broker. The JText request queue is local to the queue manager supporting the JText adapter, and is shared within the second MQ cluster allowing both brokers to send requests and handle replies. The details of the cluster setup are in Figure 5-4.

![Figure 5-4](image-url)
5.3.6 Configure system manager to connect to remote InterChange Server

The system manager is a Microsoft Windows client that is used to monitor, deploy, and configure the InterChange Server.

To configure the system manager:

1. Stop the ORB on the Microsoft Windows client machine.
2. Modify C:\IBM\Websphere\ICS\bin\CWSharedEnv.bat. The system manager executes that file to locate the remote ORB. Modify the following entries accordingly:
   
   set ORBPORT=14500
   set ORBHOST=<Remote Server Address>
3. Before starting the modified system manager, make sure that the ORB and InterChange Server are started on the remote machine.

5.3.7 Configuring InterChange Server and WebSphere BI Adapters

InterChange Server and the InterChange Server WebSphere Business Integration Adapters should be installed and tested locally before preparing the system for HA. The WebSphere Business Integration Adapters and Collaborations SupportPac, *XC63: WebSphere Interchange Server with HACMP*, provide a set of script templates for starting, stopping, and monitoring InterChange Server, the supporting queue manager, and InterChange Server adapters as one resource group under HA.

In this sample implementation, InterChange Server and InterChange Server adapters form part of the same resource group. For a remote agent setup under HA control, they need to be installed in different volume groups, and form part of separate resource groups. For that scenario, follow the steps outlined in 5.3.3, “Configuring remote adapters” on page 75.

Moving InterChange Server and InterChange Server
WebSphere Business Integration Adapters to a shared filesytem

The following steps are required to make InterChange Server highly available in a two node cluster environment. These steps assume that InterChange Server and the adapter have already been installed, configured, and tested in a non-HA environment and a volume group has been created on the shared disks called /WICSHA/ICS. This filesystem contains the class files and configuration files that are created during normal operations when interfaces are deployed or
reconfigured. The InterChange Server product library and binary files will reside in the local disks installation folder referred to as $CROSSWORLDS:

1. If InterChange Server queue manager is not already configured for failover, go to 5.3.2, “Configuring WebSphere MQ” on page 72 and follow all steps.

2. Create a volume group for InterChange Server and InterChange Server adapters and mount this onto the shared disk such as /WICSHA/ICS; this must be owned by the cwadmin user.

3. Create /WICSHA/ICS/collaborations/ on the shared disk; this is where the collaboration class files will reside. /WICSHA/ICS is the top level directory used in the example scripts:
   ```
   mkdir /WICSHA/ICS/collaborations/
   ```

4. Create /WICSHA/ICS/DLMs/ on shared disk; this is where the map class files will reside:
   ```
   mkdir /WICSHA/ICS/DLMs/
   ```

5. Create /WICSHA/ICS/logs/ on shared disk; this is where the log files will reside:
   ```
   mkdir /WICSHA/ICS/logs/
   ```

6. Create /WICSHA/ICS/config/ on shared disk; this is where configuration files will reside:
   ```
   mkdir /WICSHA/ICS/config/
   ```

7. Move existing map and collaboration folders onto the shared disk:
   ```
   mv $CROSSWORLDS/collaborations /WICSHA/ICS
   mv $CROSSWORLDS/DLMs /WICSHA/ICS
   ```

8. Move the existing configuration file onto the shared disk:
   ```
   mv $CROSSWORLDS/InterchangeSystem.cfg /WICSHA/ICS/config/InterchangeSystem.cfg
   ```

9. Create links between the local filesystem and the shared filesystem:
   ```
   ln -s /WICSHA/ICS/collaborations $CROSSWORLDS/collaborations
   ln -s /WICSHA/ICS/DLMs $CROSSWORLDS/DLMs
   ln -s /WICSHA/ICS/logs $CROSSWORLDS/logs
   ln -s /WICSHA/ICS/config $CROSSWORLDS/config
   ```

10. Configure circular logging for InterChange Server.

### 5.3.8 Configuring the ORB

In the sample implementation, the ORB is included as part of the InterChange Server cluster resource group so that InterChange Server can be automatically restarted on the same machine if the ORB process fails or failed over to the
backup machine if the primary machine fails. No special configuration is necessary.

5.3.9 Shared disk layouts

InterChange Server shared disk:

/WICSHA/ICS/  InterChange Server shared files
/WICSHA/MQ/ics422.queue.manager/data files  InterChange Server QM shared data files
/WICSHA/MQ/ics422.queue.manager/logs  InterChange Server QM shared logs

Message Broker shared disk:

/MQHA/MB_QM/log  MB QM shared logs
/MQHA/MB_QM/data/qmgrs  MB QM data files
/MQHA/MB_QM/data/brokers  MB shared configuration
/MQHA/MB_QM/data/registry  MB shared registry

Adapter shared disk:

/ADPHA/JTXT/jtxtfiles  Output directory for JText Adatper
/ADPHA/JTXT/repository  JText shared repository
/ADPHA/MQ/ADAPTER_QM/log  Adapter QM shared logs
/ADPHA/MQ/ADAPTER_QM/data  Adatper QM shared data files

5.4 HA monitoring, start, stop scripts

Every resource group needs a set of scripts that are executed by the HACMP cluster software in order to start, stop, and monitor the applications.

5.4.1 WebSphere MQ

The start, stop, and monitoring scripts are provided with MQ SupportPac, MC63: WebSphere MQ for AIX - Implementing with HACMP. These scripts are referenced by other applications that are part of the HA cluster. They are used for controlling the queue managers under HA. A set of scripts was installed for each resource group. The scripts had fully referenced file locations, which were changed to reflect the installation locations.

5.4.2 Message Broker

The start, stop, and monitoring scripts provided with WebSphere Business Integration Broker SupportPac, IC61: Configuring MQSeries Integrator for AIX with HACMP were used for running the Message Broker under HA. hamqsi_start_broker_as, hamqsi_stop_broker_as, and
**Chapter 5. Implementing the solution**

Hamqsi_monitor_broker_as were extensively modified for the scenario. WebSphere MQ HA scripts were used for starting and stopping the MQ listeners.

The sample scripts were modified to remove the sections for starting and stopping the databases, as in our scenario the databases are not under the control of HACMP. The methodology of the scripts was also changed because by default they only stopped or restarted all components. In the WebSphere Business Integration scenario, the following behavior was required:

- The monitor script should return 1 if any of the components are not running (queue manager, queue manager listener, broker)
- The stop script should stop all components.
- The start script should do the following:
  - If the listener is stopped, restart it.
  - If the queue manager is running and broker is stopped, start the broker.
  - If the queue manager is stopped and the broker is running, stop the broker, start the queue manager, and then restart the broker.
  - If the queue manager is stopped, and the broker is stopped, start the queue manager and then broker.

### 5.4.3 Remote WebSphere Business Integration Adapters

The HA control scripts need to ensure that the remote adapter and its dependent processes are capable of starting, stopping, and monitoring all required processes. The scripts used in the sample implementation are provided in Appendix A, “HA scripts” on page 127. They provide a sample template that needs to be edited to match your environment.

The monitor script returns 1 if any of the MQ components (queue manager and its listener) are not running. However, if the adapter is not running, the script attempts to restart the adapter but returns 0. This prevents repeated failover due to the end application not being available.

The stop script should stop all components of the remote adapter.

The start script should start only components that are not already started.

### 5.4.4 InterChange Server and WebSphere BI Adapters

The HA scripts start, stop, and monitor the applications that are part of the InterChange Server resource group. The HA control scripts need to ensure that InterChange Server with all its dependent processes are restarted on the failover node should the HA software initiate a failover. The WebSphere Business
Integration Adapters and Collaborations SupportPac for the relevant OS provide templates of those start, restart, stop, and monitor scripts. Generally, it is good practice to try to restart the components on the primary node first before initiating a failover.

For the sample implementation, the scripts provided as part of WebSphere Business Integration Adapters and Collaborations SupportPac, XC63: *WebSphere Interchange Server with HACMP*, were used as templates and modified in the following ways:

- Included the ORB start, stop, and monitor scripts:
  - $CROSSWORLDS/bin/hascripts/wics_orb_start
  - $CROSSWORLDS/bin/hascripts/wics_orb_restart
  - $CROSSWORLDS/bin/hascripts/wics_orb_stop.

  The following lines were added:
  
  su - cwadmin '-c $CROSSWORLDS/bin/hascripts/wics_orb_start'
su - cwadmin '-c $CROSSWORLDS/bin/hascripts/wics_orb_restart'
su - cwadmin '-c $CROSSWORLDS/bin/hascripts/wics_orb_stop'

- Modified PersistenNameserver.sh to send Nameserver process in background. It was renamed to PersistenNameserver_ha. The following lines were modified:

  # start name server
  exec ${JAVA} -classpath ${CWCLASSES}
  com.ibm.CosNaming.TransientNameServer -ORBInitialPort ${ORB_PORT} &
  echo "The Transient Name Server Process Id is $!"

- Modified the script $CROSSWORLDS/mqseres/crossworlds_mq.tst to create persistent event delivery queues:

  DEFINE QLOCAL(AP/REQJDBC/WICS422HA) DEFPSIST(YES)
  DEFINE QLOCAL(AP/ORDJDBC/WICS422HA) DEFPSIST(YES)
  DEFINE QLOCAL(AP/WBIMB/WICS422HA) DEFPSIST(YES)

  **Note:** In the next several bullets, the variable $QUEUE refers to a queue manager.

- Called the MQHA monitoring scripts inside WebSphere InterChange Server monitoring script. $CROSSWORLDS/bin/hascripts/wics_mq_restart. The following line was added:

  `/WICSHA/MQ/bin/hamqm_running $QUEUE`

- End MQ listener was included in the stop script $CROSSWORLDS/mqseries/end_mq. The following line was added:

  `endmqlsr -m $QUEUE`
Start MQ listener and create MQ Administration Server was included in the start script $CROSSWORLDS/mqseries/start_mq. The following lines were added:

```
runmqlsr -m QUEUE -t TCP -p 1415 &
strmqcsv QUEUE
```

All sample adapters were included in InterChange Server start, stop, and monitor scripts according to the template suggestion.

In the InterChange Server and adapter start and restart scripts:

```
$CROSSWORLDS/bin/hascripts/wics_ha_start,
$CROSSWORLDS/bin/hascripts/ics_restart,
$CROSSWORLDS/bin/hascripts/wics_adapter_restart
```

InterChange Server and adapters are called with the configuration file in a customized location. The following lines were modified:

```
su - cwadmin '-c $CROSSWORLDS/bin/ics_manager -start \
-c$CROSSWORLDS/config/InterchangeSystem.cfg'
su - cwadmin '-c $CROSSWORLDS/bin/connector_manager_ORDJDBC \
-start-c$CROSSWORLDS/config/InterchangeSystem.cfg'
su - cwadmin '-c $CROSSWORLDS/bin/connector_manager_REQJDBC -start \
-c$CROSSWORLDS/config/InterchangeSystem.cfg'
su - cwadmin '-c $CROSSWORLDS/bin/connector_manager_WBIMB -start \
-c$CROSSWORLDS/config/InterchangeSystem.cfg'
```

### 5.4.5 InterChange Server remote adapter resource group

Remote InterChange Server adapters are part of a separate resource group if InterChange Server are configured using remote agent technology. The resource group also contains a remote queue manager. The HA scripts for the remote InterChange Server adapter resource group are listed below. Appendix A, “HA scripts” on page 127 documents the actual scripts used.

- **rmt_adap_ha_start**: Starts the InterChange Server remote adapter and the queue manager.
- **rmt_adap_ha_stop**: Stops the InterChange Server remote adapter and the queue manager.
- **rmt_adap_ha_monitor**: Monitors the InterChange Server remote adapter and the queue manager. This references the two restart scripts listed below.
- **rmt_adap_restart**: If the InterChange Server adapter is unavailable it restarts the adapter.
- **rmt_mq_restart**: If the queue manager is unavailable it exits with an error code.
The scripts also reference scripts from WebSphere MQ SupportPac, *MC63: WebSphere MQ for AIX - Implementing with HACMP* for starting, stopping, and monitoring the queue manager.

### 5.5 System setup

This section details the hardware and software setup for the sample implementation. Note that there is no redundant power supply, however, as a high availability designer you should consider this option for meeting more stringent service level agreements.

#### 5.5.1 Hardware

- Primary and failover node: IBM pSeries 630, Model 7028 E64, 1xPowerPC (1.2GHz), 16 GB RAM each
- Shared disk system providing the application volume groups: FastT200 disk system with 10 disks, with five logical disk volumes at 32 GB each.
- Fast fibre connection between the nodes and the shared disk system
- Three NIC cards connecting to the public ethernet. On node 1 with one resource group, they are configured as one service and two standby; on node 2 they are configured for two services and one stand by
- Serial network adapter RS232 providing the cluster heartbeat

#### 5.5.2 Software

Below is a listing of the software required for implementing this high availability scenario:

- AIX 5.2, Maintenance Level 2
- HACMP 5.1, current APAR applied
- RSCT 2.3.0
- InterChange Server 4.2.2.2, with a HotFix for the ORB that makes it persistent, this fix will be available with InterChange Server version 4.2.2.3¹
  - JDBC Type 2 drivers for DB2 Version 8.1
- IBM JDK Version 1.3.1 SR5(AIX)
- WebSphere MQ Version 5.3, CSD05 patch for AIX
- WebSphere Business Integration Message Broker 5.0 CSD 3 for AIX

¹ The WebSphere InterChange Server 4.2.2.3 Fix Pack is available as of June 18, 2004.
Chapter 5. Implementing the solution

- WebSphere Business Integration Toolset 4.2.2.2 for Windows 2000 (Fix Pack 2)
- WebSphere Business Integration Message Broker Tooling for Windows 5.0 CSD 3
- WebSphere Business Integration Adapters for AIX:
  - JText 5.4.4
  - JDBC 2.4.3
  - WebSphere MQ Integrator Broker 2.5.1
- XML Datahandler 2.5

It is recommended that the latest supported software and code fixes are applied to your specific Business Integration solution.

5.6 Adding applications to HACMP control

1. Shut down the cluster if it is already running on all nodes of the cluster:
   ```
   smitty clstop
   ```
2. Configure application server:
   a. Log on to the AIX console.
   b. `smitty hacmp`
   c. Extended Configuration
   d. Extended Resource Configuration
   e. HACMP Extended Resources Configuration
   f. Configure HACMP application server.
   g. Add an application server such as WICS422HA.
   h. Select the server name.
   i. Select the location of the start script such as
      ```
      $CROSSWORLDS/bin/hascripts/wics_ha_start
      ```
   j. Select the location of the start script such as
      ```
      $CROSSWORLDS/bin/hascripts/wics_ha_stop
      ```
   k. Exit by pressing Enter.
3. Configure the application monitor:
   a. `smitty hacmp`
   b. Extended Configuration
   c. Extended Resource Configuration
d. HACMP Extended Resource Configuration

e. Configure the HACMP application monitoring.

f. Configure the custom application monitor.

g. Add the custom application monitor.

h. Select the server name.

i. Stabilization Interval: choose 200 (seconds) to allow the cluster to start the applications.

j. The restart count is 3.

k. The restart interval is 0.

l. The clean up method is usually the stop script.

m. The restart method is usually the start script.

n. Exit by pressing Enter.

4. Adding application server to the resource group:

   a. smitty hacmp
   b. Extended Configuration
   c. Extended Resource Configuration
   d. HACMP Extended Resource Group Configuration
   e. Change/show resource attributes for a resource group.
   f. Select the resource group.
   g. Select the application server.
   h. Exit by pressing Enter.

5. Synchronize the cluster:

   a. smitty hacmp
   b. Extended configuration
   c. Extended verification and synchronization
   d. Use defaults and press Enter.

Verify the success of the HACMP configuration by checking the log file on /tmp/hacamp.out. Now start the cluster with:

   smitty clstart

   choose Reacquire resources after forced down to false.

Execute usr/es/sbin/cluster/utilities/clRGinfo to check whether the resource group is running.
Representative test plan

I wish I could tell you that if you follow a good solid test plan, it will make you rich and famous; I can’t. But, it might improve your looks. You won’t be getting all those 3:00 am phone calls because the production environment just fell over.
-- Lyn Elkins
6.1 Introduction

This chapter provides samples to help the reader get started contemplating their test plans. It is outside the scope of this redbook to provide a treatise on test methodology and the like. We all know that proper, thorough testing of our applications, systems, environments, etc. is critical to ensuring as painless a transition as possible when new elements are promoted into the production environment.

So, briefly, as in any testing start with the basics: Convince yourself that the HA cluster is working and configured as you expect. Can you ping one machine in the HA cluster from another machine in the cluster? What about a machine outside the cluster? Work your way slowly through the expected functionality. The tests should build on one another and in complexity as you go along. In considering HA in particular, the authors found it helpful to keep the following goals in mind:

- Most importantly, the applications (flows, collabs, etc.) can recover and restart.
- The applications will continue to process or recover when a component fails.
- The applications will perform under heavy load.
  
  HA is also about performance of the application when individual components are under stress. Not just failure.
- The scenario works when there are multiple Message Brokers and InterChange Servers; for example, a flow fails on one node and is restarted on another.

**Note:** Throughout this chapter, the term *kill* refers to gracefully stopping the component using available facilities, or stopping the component ungracefully by terminating its processes. Either is equally acceptable.

6.2 Verify basic machine failover

Both nodes are up.

1. Pull the plug
   a. Pull plug on node 1.
   b. Does HACMP failover occur?
   c. Repeat for node 2.

2. Ping from outside the HA cluster:
   a. Ping node 1 from an AIX box that is not part of the HA cluster.
b. Unplug machine from network.
c. Does the ping eventually start going through to node 2?
d. Repeat for node 2.

3. Verify access to shared storage:
   a. For node 1, verify that the expected shared file systems are accessible.
   b. Verify that the database server and databases are accessible.
   c. Pull plug on node 1.
   d. Does node failover?
   e. Are the expected shared file systems now accessible from node 2?
   f. Can we access the database?
   g. Repeat for node 2.

### 6.3 Test WebSphere MQ and Message Broker failover

MQ is under HA on nodes 1 and 2 by virtue of InterChange Server and Message Broker having been brought under HA.

Both nodes are up.

Set the number of retries for each component to 0 to force failover rather than restart:

1. Kill the queue manager:
   a. Start the broker and the associated queue manager on node 1. For example, start the Message Broker RG on node 1.
   b. Kill the associated queue manager.
   c. Do the broker and queue manager failover to node 2?
   d. Send a test message to verify that the RG components are operational.
   e. Repeat test on node 2.

2. Kill the Message Broker:
   a. Start the broker and the associate queue manager on node 1. For example, start the Message Broker RG on node 1.
   b. Kill broker.
   c. Do the broker and queue manager failover to node 2?
   d. Send test message to verify that RG components are operational.
   e. Repeat test on node 2.

3. Kill the listener
a. Start the broker and the associate queue manager on node 1. For example, start the Message Broker RG on node 1.
b. Kill the queue manager's listener.
c. Do the broker and queue manager failover to node 2?
d. Send a test message to verify that RG components are operational.
e. Repeat test on node 2.

### 6.4 Test Message Broker adapter failover

1. Kill the adapter:
   a. Start the Adapter RG on node 1.
   b. Kill the adapter.
   c. Does the Adapter RG failover to node 2?
   d. Send test message to verify that RG components are operational.
   e. Repeat test on node 2.

2. Kill the queue manager:
   a. Start the Adapter RG on node 1.
   b. Kill the associated queue manager.
   c. Does the Adapter RG failover to node 2?
   d. Send test message to verify that RG components are operational.
   e. Repeat test on node 2.

3. Test queue manager connections:
   a. Start Message Broker RG and adapter RG on node 1.
   b. Set up channels between the two queue managers.
   c. Run batch PUTter against broker.
   d. Kill the adapter queue manager.
   e. Does the adapter RG failover to node 2?:
      - Do the channels restart and reconnect?
      - What happens to the adapter?
      - How does the PUTter behave?
      - What happened to the messages?
6.5 Test InterChange Server and InterChange Server adapter failover

MQ is under HA on nodes 1 and 2 by virtue of InterChange Server and Message Broker having been brought under HA.

Both nodes are up.

Set the number of retries for each component to 0 to force failover rather than restart:

1. Kill the InterChange Server:
   a. Start the InterChange Server RG on node 1.
   b. Kill InterChange Server.
   c. Does the InterChange Server RG failover to node 2.
   d. Does everything restart and reconnect as expected?
   e. Repeat test on node 2.

2. Kill the queue manager:
   a. Start the InterChange Server RG on node 1.
   b. Kill the associated queue manager.
   c. Does the InterChange Server RG failover to node 2.
   d. Does everything restart and reconnect as expected?
   e. Repeat test on node 2.

3. Kill the ORB
   a. Start the InterChange Server RG on node 1.
   b. Kill the ORB.
   c. Does the InterChange Server RG failover to node 2.
      Does everything restart and reconnect as expected?
   d. Repeat test on node 2.

4. Kill the adapter:
   a. Start the InterChange Server RG on node 1.
   b. Kill the JDBC adapter associated with the REQ DB.
   c. Verify that no RG failover occurs.
   d. Send test message.
   e. What happens to InterChange Server?
      No events will be processed. After the adapter restarts, all events will be processed without manual intervention.
      Queue manager?
      ORB?
f. Repeat test on node 2.

**Test with ControllerStoreAndForwardMode = TRUE**

1. Kill the JDBC adapter:
   a. Start the InterChange Server RG on node 1.
   b. Kill the JDBC adapter associated with the ORD DB.
   c. Verify that no RG failover occurs.
   d. Send test message.
   e. What happens to InterChange Server?
      Expect: Collaboration waits until adapter is restarted. No events will be processed. After adapter restart, all events will be processed without manual intervention.
      Queue manager?
      ORB?

f. Repeat test on node 2.

2. Kill the MQ Integrator Broker (Message Broker) adapter:
   a. Start the InterChange Server RG on node 1.
   b. Kill the Message Broker adapter.
   c. Send test message.
   d. What happens to the InterChange Server?
      Expect: Collaboration waits until adapter is restarted. No events will be processed. After adapter restart, all events will be processed without manual intervention.
      Queue manager?
      ORB?

   e. Repeat test on node 2.

**Test with ControllerStoreAndForwardMode = False**

1. Kill the JDBC adapter
   Start the InterChange Server RG on node 1:
   a. Kill the JDBC adapter associated with the ORB DB.
   b. Send test message.
   c. What happens to the InterChange Server?
      Expect: Collaboration fails all events that are received while the adapter is down. After adapter restart, none of the failed events will be processed automatically. New events received after adapter restart will be processed normally.
      Queue manager?
ORB?
d. Repeat test on node 2.

2. Kill the MQ Integrator Broker (Message Broker) adapter:
   a. Start the InterChange Server RG on node 1.
   b. Kill the Message Broker adapter.
   c. Send test message.
   d. What happens to the InterChange Server?
      Expect: Collaboration fails all events that are received while the adapter is
down. After adapter restart, none of the failed events will be processed
automatically. New events received after adapter restart will be processed
normally.

Queue manager?
ORB?
e. Repeat test on node 2.
Chapter 7. Testing and validation

This chapter discusses in detail the test cases required to comprehensively test a highly available WebSphere Business Integration environment. This chapter also documents and compares the expected and the actual test results.

Testing of WebSphere Business Integration HA clusters is done at various levels. This is to ensure that all single points of failure are tested and accounted for. The various levels are listed below:

- Individual applications in a WebSphere Business Integration HA cluster; for example, the ORB, JText adapter, etc.
- Resource groups
- Cluster nodes

This chapter also discusses the impact of various parameters such as the InterChange Server repository size, number of MQ queues, and the number of adapters on the availability of the solution.
7.1 WebSphere Business Integration HA sanity tests

For all the test cases discussed in the current section, the default configuration is described here:

- HACMP cluster manager active on node 1 and node 2.
- IBM ORB, InterChange Server queue manager, InterChange Server, InterChange Server adapters (REQJDBC, ORDJDBC and WBIMB) running on node 1.
- Message Broker, Message Broker adapter, Message Broker queue manager running on node 2.
- InterChange Server repository and Message Broker broker database are located on a third AIX box, and is not part of the WebSphere Business Integration HA cluster.
- Application databases that InterChange Server adapters connect to are located on a Windows 2000 servers, and are not part of the WebSphere Business Integration HA cluster.
- InterChange Server System Manager (CSM) and Message Broker Configuration Manager are installed on Windows 2000 workstations, and are not part of the WebSphere Business Integration HA cluster.

7.1.1 Manual system start up tests

Before enabling the HA cluster, sanity tests must be done for all the WebSphere Business Integration applications on each cluster node. These tests are executed with the cluster disabled:

1. Node 1 and node 2 are both active:
   a. Disable the cluster on both the nodes by the executing the command `smitty clsstop`
   b. Stopping the cluster unmounts the shared drives. Mount the shared drives.
   c. Start the IBM ORB, InterChange Server queue manager, InterChange Server, and the WebSphere Business Integration Adapters on node 1.
   d. Start the Message Broker queue manager, Message Broker, and the JText adapter on node 2.
   e. Test a sample flow by creating a customer through the requisitions application.
   f. Observe the results to verify that no errors are reported during flow execution.
2. Node 2 is the only active node:
   a. Stop all the applications on node 1.
   b. Assign the service IP address of node 1, start the node1 application
      services on node 2 to the standby adapter on node 2, and shut down node
      1.
   c. Repeat the above test on node 2.

3. Node 1 is the only active node:
   a. Assign the service IP address of node 2, start the node2 application
      services on Node1 to the stand by adapter on node 1, and shut down node
      2.
   b. Repeat the above test on node 1.

7.1.2 Verify WebSphere Business Integration HA configuration

The tests cases for verifying the WebSphere Business Integration HA
configuration are:

1. Persistence of MQ/JMS queues: Verify that the queues are configured as
   persistent queues.

2. Shared files: The following files should be located in shared directories:
   a. InterChange Server log files, InterchangeSystem.cfg, InterChange Server
      class files, WebSphere MQ logs, and adapter log files
   b. IBM ORB IOR file: By default this is called CxCosNameRepos.ior and is
      located in the InterChange Server home directory. If InterChange Server
      and InterChange Server adapters are part of different resource groups,
      this file location should be configured through the system manager to be
      located on a shared drive. To change the location of the
      CxCosNameRepos.ior, the steps are listed below:
      i. Connect to the InterChange Server through the system manager.
      ii. Open the Edit Configuration screen of the server.
      iii. Click the **Environmental Properties** tab.
      iv. Verify that value of the property **CosNamingPeristencyFile** is the
          shared location of the IOR file.
      v. CSM automatically adds a CORBA section to the cfg file when the
         above environmental property is added.
   c. Queue manager data for every queue manger should reside in a shared
      location.
7.2 WebSphere Business Integration Components under HACMP control

These tests are similar to the ones in 7.1.1, “Manual system start up tests” on page 96. However, in this test suite, the WebSphere Business Integration applications are under the control of the HACMP cluster manager. The objective of this test suite is to verify if HACMP is able to start, restart, and monitor all the individual applications that are part of the cluster. The HACMP cluster has one monitoring script per resource group. These monitoring scripts continuously monitor the health of the cluster, and ensure that the individual WebSphere Business Integration applications are restarted in the correct order if a failure happens.

HACMP log files
An easy way to monitor cluster events and messages is by tailing the following HACMP log files:

- /tmp/hacmp.out - Outputs detailed logs on cluster events and output from application start and stop scripts. Note that output from the monitoring is not part of this log file.
- /usr/es/adm/cluster.log - Contains timestamped, formatted messages generated by HACMP start and stop scripts and daemons. This file contains only high level events, and not detailed log information. The above log files are local to each node.

Application monitoring log files
Application monitoring in HACMP has its own set of log files. Here are the log files for our current scenario:

- /tmp/clappmond.WICS422HA.log
- /tmp/clappmond.WICS422HA.monitor.log
- /tmp/clappmond.MBHA.log
- /tmp/clappmond.MBHA.monitor.log
- /tmp/clappmond.WBIA.log
- /tmp/clappmond.WBIA.monitor.log

All the above files are local to each node. Notice that there are two log files per application. The clappmond.$APPNAME.monitor.log is overwritten on each monitor cycle. The clappmond.$APPNAME.log file is overwritten everytime monitoring is restarted. The application log files contain messages that are generated while executing the monitoring scripts. The timestamp on the application monitoring files is an indicator of the current status of the application monitoring.
7.2.1 Automatic system startup/restart under HACMP control

The various test scenarios are discussed below. The cluster services on all the cluster nodes before starting this test suite. The components to test are:

- **IBM ORB**
  - a. Manually stop or kill the IBM ORB process:
    
    ```bash
    ps -ef | grep TransientNameServer
    kill OrbProcessId
    ```
  
  - b. Cluster manager restarts the IBM ORB.
  
  - c. Check if the ORB is alive:
    
    ```bash
    ps -ef | grep TransientNameServer
    ```
  
  - d. After the ORB is restarted, clients such as the system manager (CSM) will also need to be restarted in order to connect to the InterChange Server repository.
  
  - e. **Result:** The test was successful. The name server is persistent. InterChange Server and the InterChange Server adapters need not be restarted after the IBM ORB is restarted.

  - **InterChange Server adapters:** If a InterChange Server adapter process fails, the cluster manager will restart the adapters. The InterChange Server and the queue manager are not impacted. Manually stop each of the InterChange Server adapters, and observe that it is restarted successfully.

    - a. Kill the adapter process:
      
      ```bash
      ps -ef | grep REQJDBC
      kill REQJDBCAdapterProcess
      ```
    
    - b. Application Monitor restarts adapter.
    
    - c. Test if adapter is alive:
      
      ```bash
      $CROSSWORLDS/bin/ connector_manager_REQJDBC IsConnectorAgentAlive
      ```
    
    - d. Repeat tests for all InterChange Server adapters.
    
    - e. **Result:** The ORBJDBC adapter had timing issues when the hacmp monitoring frequency was 60 seconds. This is because monitoring starts 60 seconds after start up. But the **wics_ha_start** script may not complete within the time interval. This leads to timing problems. On changing the monitoring frequency to 180 seconds, all the adapter restart work is fine.

  - **InterChange Server queue manager:**

    - a. Stop the InterChange Server queue manager:
      
      ```bash
      endmqm -i ics422.queue.manager
      ```
    
    - b. InterChange Server adapters are shut down by the cluster manager.
c. The InterChange Server is shut down by the cluster manager.

d. The cluster manager restarts InterChange Server queue manager, InterChange Server, and then the InterChange Server adapters in the stated order.

e. Repeat the test by killing the queue manager process (an ungraceful exit of the queue manager):

```
ps -ef | grep ics422.queue.manager | grep amq
kill -9 $AllThePAboveProcessIds
```

f. **Result:** When the queue manager processes are killed ungracefully, the system does not clean up the .ini file. Hence, to test the health of the queue manager, it is not sufficient to parse the ini file.

► InterChange Server:

a. Stop the InterChange Server manually:

```
ics_manager -stop
```

b. The InterChange Server is restarted by the cluster manager.

c. InterChange Server restart does not require restart of other applications. Adapters will reconnect to the InterChange Server automatically.

d. Verify that InterChange Server is running.

```
ics_manager -IsServerAlive
```

e. Repeat the test by killing the interchange server process (and ungraceful exit of the InterChange Server).

f. **Result:** The monitoring interval was initially set to 60 seconds, and the sleep time on InterChange Server restart to 60 seconds. The app monitor was timing out, and monitoring was exiting with error code. On examining the `clappmond.WICS422HA.log` file, exit code for the monitor was 12. The monitoring interval for the InterChange Server application was increased to 180. This fixed the time out issue and the InterChange Server restart was successful.

► Message Broker:

a. Stop the Message Broker manually:

```
mqsistop -i MBBRK
```

b. Message Broker is restarted by the cluster manager.

c. Message Broker restart does not require restart of the other applications.

d. **Result:** HACMP did restart the broker, but due to the configuration of the monitor, it also stopped and restarted the queue manager. As this is not required, the configuration was modified. Initially the monitor was set up to
run the stop script after every monitor failure. This is not required as there is logic in the start scripts, which behave dependant on what is running.

After removing the stop script from the monitor setup, the test was rerun. As before, the broker was restarted, but unlike before, the queue manager was not restarted.

► Message Broker queue manager:
   a. Stop the Message Broker queue manager:
      ```
      endmqm -i MB_QM
      ```
   b. Message Broker is shut down by the cluster manager.
   c. Cluster manager restarts Message Broker queue manager and then Message Broker in the stated order.
   d. **Result:** This test completed without any problems.

► JTEXT adapter queue manager:
   a. Stop the JTEXT adapter queue manager.
      ```
      endmqm -i ADAPTER_QM
      ```
   b. JTEXT adapter is shut down by the cluster manager.
   c. Cluster manager restarts the JTEXT adapter queue manager and then the JTEXT adapter.
   d. **Result:** This test completed without any problems.

► JTEXT adapter:
   a. Stop the JTEXT adapter.
      ```
      connector_manager_JText -kill
      ```
   b. The JTEXT adapter is restarted by the cluster manager.
   c. The JTEXT adapter restart does not require restart of other applications.
   d. **Result:** This test completed without any problems.

### 7.2.2 Restart attempts setting

Verify the number of retry attempts. Each WebSphere Business Integration application will be restarted three times before a resource group failover is initiated. The number of retry attempts can be configured in HACMP. If the WebSphere Business Integration application monitoring is configured not to trigger failover (as may be in the case of WebSphere Business Integration Adapter resource group), the resource group will continue to run while attempting to start the adapter:

► InterChange Server:
• Shut down the InterChange Server repository database. This will make InterChange Server unavailable.

b. InterChange Server will shut down.

c. Cluster manager will restart InterChange Server.

d. After the maximum number of retry attempts, the InterChange Server node will trigger a failover.

e. Failover can be verified by checking the HACMP logs.

▶ InterChange Server adapters:

a. Shut down the REQCUST application database.

b. The REQJDBC adapter will shut down.

c. Cluster manager restarts the REQJDBC adapter.

d. After the maximum number of retry attempts, the adapter will shut down. The node will not failover as adapter resource groups are not configured to trigger failover.

e. Repeat the test for ORDJDBC and MQ Integrator Broker adapter.

▶ Message Broker:

a. Shut down Message Broker renaming the bipservice executable (to prevent restart) and then killing the broker:

```
ps -ef | grep bipservice | grep -v grep | \ 
awk '{print $2}' | xargs kill -9
```

b. The Message Broker will be shut down.

c. After the maximum number of retry attempts, the Message Broker node will trigger a failover.

d. Failover can be verified by checking the HACMP logs.

▶ Message Broker queue manager:

a. Shut down Message Broker queue manage renaming he AMQXSSVN.EXE executable (to prevent restart) and then killing the broker:

```
ps -ef | grep AMQXSSVN.EXE | grep -v grep | \ 
awk '{print $2}' | xargs kill -9
```

b. The Message Broker queue manager will be shut down.

c. After the maximum number of retry attempts, the Message Broker node will trigger a failover.

d. Failover can be verified by checking the HACMP logs.

▶ Remote adapters:

a. Shut down the Message Broker adapter.
b. Cluster manager attempts to restart the Message Broker adapter.
c. An adapter failure for this specific setup does not trigger a failover. The resource group will continue to run, and will continue to attempt to start the adapter infinitely.

**Results:** The tests documented in the current section were successful, and verified the number of retry attempts.

3. Hardware failures not resulting in a node failover. Each node in the WebSphere Business Integration HA cluster has a minimum of two network adapters. In the event of a failure of a single network adapter, the standby network adapter can take over the service addresses of the primary adapter:

a. Disconnect the primary adapter on node 1 from the network.
b. The cluster manager will detect a network down event.
c. HACMP cluster manager fails over the all the service IP addresses from the primary network adapter to the standby network adapter.
d. HACMP cluster manager stops and restarts services on node 1 using the standby network adapter.
e. Verify using HACMP log files.
f. Repeat the test on node 2 for both the MB resource group and the adapter resource group using the respective network adapters.

**Result:** Since the network adapters were configured to use IP aliasing, a single standby network adapter will take over all the service addresses. This test worked as expected.

### 7.2.3 Failover

This group of tests validates the failover process of the WebSphere Business Integration HA solution. The objective is to validate resource group and node failover so that the WebSphere Business Integration solution is indeed highly available. *For this suite of tests, change the number of retry attempts for each component to zero to force the failover rather than restart.*

1. Failover of InterChange Server resource group from node 1 to node 2. There are several situations that will trigger InterChange Server resource group failover. The scenario to test for InterChange Server resource group failover are:

   - IBM ORB is unavailable:
      i. Rename the ORB startup file `PersistentNameServer.sh` to `PersistentNameServer.bkp`
      ii. Manually stop the IBM ORB process. Execute the HA script `wics_orb_stop`
iii. Verify that application monitor log file exits with error code:

```
tail -f /tmp/clappmond.WICS422HA.log
```

iv. Verify ics_rg resource group takeover by tailing the HACMP log files on both the nodes:

```
tail -f /tmp/hacmp.out
```

v. As always, test failover by creating a sample flow through the Web application and reconnecting to InterChange Server through the system manager.

vi. Rename the ORB start up file to the original name.

vii. **Result**: The test results were as expected. The current ORB monitoring script also restarts the IBM ORB if it is not running. Hence, setting the number of retry attempts to zero is not enough to trigger a failover. Renaming the batch file forces the ORB restart to fail, hence triggering a failover.

- InterChange Server queue manager is unavailable.

i. Manually stop the InterChange Server queue manager:

```
endmqm -i ics422.queue.manager
```

ii. InterChange Server resource group fails over.

iii. Failover can be verified by checking the HACMP logs.

iv. Repeat the failover test by killing the queue manager.

v. **Result**: The test results were as expected. When the InterChange Server queue manager is not available, the monitoring script returns an error code (*Exit Code 1*). It does not attempt to restart just the InterChange Server queue manager as this will not suffice.

- InterChange Server is unavailable:

i. Rename the interchange server start up script `ics_manager` to `ics_manager_bkp`.

ii. Manually stop the InterChange Server process:

```
ps -ef | grep $WICS_SERVER_NAME
kill $AboveProcessId
```

iii. InterChange Server resource group fails over.

iv. **Result**: The test results were as expected. The current monitoring script also restarts InterChange Server if it is not running. Hence, setting the number of retry attempts to zero is not enough to trigger a failover. Renaming the batch file forces the InterChange Server restart to fail hence triggering a failover.
2. Failover of Message Broker resource group from node 2 to node 1. The scenarios to test are:
   a. The Message Broker queue manager is unavailable.
   b. Message Broker is unavailable.

3. Failover of the Remote Adapter resource group:
   a. Failover remote adapter resource group using any of the test cases stated in #4.
   b. HACMP cluster manager shuts down Message Broker adapter.
   c. Remote adapter resource group fails over.

4. Verify node 1 failover - configuration:
   a. After node 1 fails over to node 2, test the failover:
      i. InterChange Server, InterChange Server queue manager, InterChange Server adapters are up and running and node 2.
      ii. Test failover creating a customer record through the requisitions application.
   b. Remote adapters if any should automatically reconnect.
   c. Restart InterChange Server System Manager (CSM) after InterChange Server failover is complete. CSM should connect successfully to the InterChange Server on the secondary node.

5. Verify node 2 failover:
   a. Message Broker, Message Broker queue manager, Message Broker adapters are up and running and node 1.
   b. Message Broker configuration manager should be able to connect to the Message Broker after Message Broker fails over. The Message Broker failover is really transparent to the configuration manager.

For all the previous test cases, the failover can be verified by doing the following:

1. Examine HACMP log files to observe resource group take over.
   
   `tail -f /tmp/hacmp.out`

2. The application monitor files for the resource group on the primary node will indicate that the monitoring has exited with an error code 1.
   
   `tail -f /tmp/clappmond.$APPNAME.log`

3. Verify system availability by creating a sample customer through the Web application.
7.2.4 Fallback

Fallback refers to the movement of a resource group from a secondary or a failover node to the primary node, which is being reintegrated into the cluster. In the current WebSphere Business Integration cluster, automatic fallback is disabled. However, manual reintegration should be validated:

1. Validate manual fallback:
   a. Bring node 1 down:
      ```
      shutdown -r now
      ```
   b. Node 1 resource group fails over to node 2.
   c. Verify failover by looking at the HACMP log file:
      ```
      tail -f /tmp/hacmp.out
      ```
   d. Start up cluster on node 1 after node 1 is back up:
      ```
      smitty clstart
      ```
   e. Observe that the resource groups are still running on node 2 even though the cluster on node 1 is backed up.
   f. Repeat the above test for node 2 fallback.

   **Test results:** The test results were as expected. Manual reintegration is preferable as compared to automatic fallback. This is because it is preferable to fallback to the original configuration when the system usage is low.

7.3 Impact of component availability on flow integrity

To observe the impact of component failures on flow integrity, and to understand how in progress events and messages are handled during failover, it is recommended to create a batch of event flows. In the current scenario, a batch of events can be created by inserting a number of records simultaneously into the requisitions application database. The REQJDBC adapter detects these events and triggers the synchronization process.

Each test will be performed in isolation with 2500 business events.

7.3.1 InterChange Server resource group with local adapters

These configuration properties were used for the tests in the current section.

**Collaboration configuration**

- PersistServiceCallInTransitState=True
- PauseWhenCriticalErrorOccurs=False
Adapter configuration

- ControllerStoreAndForwardMode=True
- PollQuantity=20
- ConcurrentTriggeredEvenFlows=20
- PollFrequency=1000
- Agent and Controller Trace are set to 0

For this suite of tests, change the number of retry attempts for each component to zero to force the failover rather than restart.

Test #1

Send 25000 events to Requisition Customer database.

Verify expected behavior

2500 events are successfully processed by InterChange Server, Message Broker, and the adapters.

Time = Approximately 3.5 minutes.

Test #2

While the business events are being processed, shut down the active InterChange Server node.

Result

- 13880 messages arrived at the destination, when shutdown was initiated.
- Total messages 2491 after 8:00 minutes
- Nine messages had to be manually resubmitted as they were persisted as Deferred or In Transit events.
- Time = 8 minutes

Test #3

Change RecoveryMode to Always and rerun Test #2.

Result

- The restart took a few seconds longer, where the deferred events were recovered.
Test #4
While the events are running, *kill InterChange Server queue manager* with *kill -9* on its process ID. Keep RecoveryMode to “always”. Verify the expected behavior.

**Result**
- InterChange Server continues processing all events that currently are in progress.
- InterChange Server is shut down gracefully, and finishes its processing before the resource group is failed over to node 2.
- After a successful restart, InterChange Server and the adapters continue processing.
- No events are lost; no events had to be manually resubmitted.

Test #5
While the events are running, kill *InterChange Server queue manager listener* with *kill -9* on its process ID, set the restart count to 0, and a resource group failover will be initiated.

**Result:**
- Events continued to be processed as normal.
- All 2500 messages were successfully processed, and were delivered to the file system.

Test #6
While the events are running, kill InterChange Server queue manager’s listener with *kill -9* on its process ID. No failover or restart of the listener is initiated. Wait for 5 to 10 minutes before resending the batch.

**Result:**
- All 2500 messages were processed.
- All events failed at the target Message Broker adapter, which connects to the same queue manager.
- All events were persisted InTransist.
- Error message:
  
  WBI_CustomerSync.thread9REQJDBC_83121xworlds_events2004-05-06
  00:00:00.0CxCCommon.FlowExecContext@5f57b0e.Main:, Error 11065 Request CustomerPartner.Create was not processed by WBIMB, return status -1: Reason
After restart of listener, events In Transit need to be resubmitted manually, without restart of any other component.

Test #7
After the batch has been started, kill the REQJDBC adapter.

Expected behavior
- Adapters do not trigger a failover.
- InterChange Server continues processing as normal, no events Deferred or In Transit.
- Polling adapter will not continue sending events to InterChange Server.
- One event modified in events table to a status 3. It is reprocessed before the others.
- After the adapter restarts successfully, processing is continued. No events need to be resent manually.

Test #8
After the event processing has been started, kill the Message Broker adapter.

Result
- Adapters do not trigger a failover.
- InterChange Server continues processing as normal.
- REQJDBC continues sending events to InterChange Server.
- The collaboration waits for the adapter to become available again before sending new requests to the Message Broker adapter.
- A total of 441 events were processed before the shutdown, the rest after restart without intervention.

Test #9
After the event processing has been started, kill the Message Broker adapter. Sometime later kill the InterChange Server.

Result
- Adapters do not trigger a failover.
REQJDBC continues sending events to InterChange Server.
- InterChange Server continues processing as normal.
- The collaboration waits for the adapter to become available again.
- InterChange Server and Message Broker adapter are stopped and restarted; all events are recovered and reprocessed.
- In our collaboration logic, the event is sent to the ORD database before it is sent to Message Broker. Events that were recovered were recovered from the start of the collaboration, and therefore they were sent twice to the ORD database. This behavior can be controlled by use of the collaboration property “Convert Create” if the target supports updates.

**Test #10**
While the events are running, *kill ORB’s name service*. Keep RecoveryMode to *Always*. Verify the expected behavior.

**Result**
- InterChange Server and adapters continue processing.
- All messages arrive at destination without manual intervention.
- Adapters that go down will not be able to reconnect.

**Test #11**
Continue Test #10, send 20 events.

**Result**
- Events are queued in delivery queue. Status *In Transit*.

**Test #12**
Continue Test #11, send 20 events. Restart ORB. Restart Message Broker adapter.

**Result**
- Message Broker adapter reconnects as per logfile. According to IsConnectorAgentAlive, all adapters are running but Message Broker is not according to SM.
- Events are queued in the delivery queue.

**Test #13**
Continue Test #12, and restart ORB. Restart the Message Broker adapter.
Results

- The second batch of 20 events are processed without manual intervention.
- The first batch of 20 events with status In Transit have to be resubmitted.
- All events are recovered and processed.

Test #14

After 2500 events have been triggered and are processed by InterChange Server. Kill InterChange Server. The `ics_restart` script was modified to exit 1 when it tried to restart. This was done to force a failover. When InterChange Server was killed, there were 1396 events in the xworlds_events table. A total of 233 events had completed processing. The remaining events had been polled and were ready to be processed by the InterChange Server, or were being processed by InterChange Server when it was killed.

Result

- InterChange Server failover was successful. After failover InterChange Server continued processing events. There were no events left in the xworlds_events table after event processing was complete.
- Polling adapter will be paused and will not continue sending events to InterChange Server. Events are polled after restart.
- After InterChange Server restarts, all messages are recovered.
- One message was stuck on `/AP/REQJDBC/WICS422HA` queue. On querying for messages using the flow manager, the message was found be an In Transit message. This message was sucessfullly processed on manual submit.
- All inProgress messages were recovered.
- Revert the `ics_restart` script changes after the test is done.

7.3.2 InterChange Server resource group with remote adapter

Collaboration configuration

The collaboration configuration is the same as Chapter 7.3.2, “InterChange Server resource group with remote adapter” on page 111.

Adapter configuration

The adapter configuration is the same as Chapter 7.3.1, “InterChange Server resource group with local adapters” on page 106.

The ORDJDBC adapter has been deployed remotely on the target database. It is not under HA control.
For this suite of tests, change the number of retry attempts for each component to zero to force the failover rather than restart. Only InterChange Server was tested, and the results were held on the transmission to Message Broker.

Test #1
Send 20 events to Requisition Customer database. Shut down InterChange Server. Wait until InterChange Server is restarted and send another 20 events.

Result
Forty events are successfully processed by InterChange Server, Message Broker, and the adapters. No MQ triggering had to be configured, the adapter was activated when InterChange Server restarted sending and receiving admin messages through JMS.

Test #2
Send 2500 events to the requisition database. While the batch is running, shut down the InterChange Server.

Result
- 1000 messages had arrived at the destination when shutdown was initiated.
- About 600 messages piled up on the REQCUST delivery queue until InterChange Server was active again.
- All 2500 messages arrived at the destination, and no messages had to be manually resubmitted.
- Time = 3 minutes
- The transfer time is not increased.

Test #3
While the events are running, shut down the active InterChange Server node.

Result
- 1000 messages had arrived at the destination when shutdown was initiated.
- After the InterChange Server resource group failed over to node 2, the remote adapter reconnected automatically to InterChange Server, and started processing events.
- Fifteen messages were in the In Transit state and had to be manually resubmitted.
- Failover time was 3 minutes and 35 seconds.
- Total time to process 2500 events including failover time was 8 minutes and 20 seconds.
7.3.3 Remote agent configuration under HA

ORDJDBC adapter has been deployed in a remote agent configuration mode. This means that the ORDJDBC adapter is part of a separate resource group. It is installed using remote agent technology. For details regarding usage of remote agent technology, please refer to Appendix B of the *WebSphere InterChange Server installation guide for UNIX*. The remote adapter resource `ics_adap_rg` also contains a remote queue manager apart from the ORDJDBC remote adapter.

**Collaboration configuration**

Collaboration configuration is the same as in 7.3.1, “InterChange Server resource group with local adapters” on page 106.

**Adapter configuration**

The adapter configuration is the same as in 7.3.1, “InterChange Server resource group with local adapters” on page 106.

Below are a limited set of tests using the WebSphere Interchange Server remote adapter configurations.

**Test #1**

- The InterChange Server resource group is on node 1, and the InterChange Server remote adapter resource group is on node 2.
- Send 2500 events to the Requisition Customer database.
- *Shut down InterChange Server queue manager after 1000 events are processed.*

**Results**

- The monitoring scripts shut down and restart all the applications on the InterChange Server resource group.
- ORJDBC Adapter Controller is stopped by the InterChange Server because the InterChange Server queue manager is unavailable.
- The ORDJDBC remote adapter using JMS transport can only reconnect to InterChange Server if the adapter controller is manually restarted.
- A total of 183 failed messages in *In Transit* status. This was because the ORDJDBC connector controller was not available. They had to be manually resubmitted.
- One message in the ORDJDBC/ResponseQueue.
Test #2

- The InterChange Server resource group is on node 1, and the InterChange Server remote adapter resource group is on node 2.
- Send 2500 events to Requisition Customer database.
- Shut down InterChange Server remote adapter queue manager after 1000 events are processed.

Result

- The monitoring scripts shuts down and restarts the remote adapter.
- The monitoring scripts restarts the adapter queue manager and the adapter.
- The ORDJDBC remote adapter is reconnected to InterChange Server on restart.
- All 2500 events were processed successfully.
- No In transit or failed messages

Test #3

- The InterChange Server resource group is on node 1, and the InterChange Server remote adapter resource group is on node 2.
- Send 2500 events to Requisition Customer database.
- Failover the Remote Adapter resource group to node 1

Result

- The remote adapter resource group moves over to node 2.
- The monitoring scripts restarts the adapter queue manager and the adapter.
- The ORDJDBC remote adapter reconnected to InterChange Server on restart.
- All 2500 events were processed successfully.
- No In Transit or failed messages

Test #1 indicates a limitation of using remote agent technology in an HA configuration. When the InterChange Server queue manager is unavailable, the adapter controller shuts down (only true for InterChange Server adapters using JMS as transport). Since this configuration requires manual intervention, it is not a recommended HA configuration.

7.3.4 Message Broker resource group

Test #1

While the events are running, kill the Message Broker queue manager.
Result
The WebSphere InterChange Server and Message Broker adapter are unaffected by the unavailability of the Message Broker queue manager, and will continue to process business processes. The InterChange Server will continue to put messages onto the transmission queue for the Message Broker queue manager; these messages will remain there until the queue manager is restarted. Any messages on Message Broker queue managers queues will be preserved and made available upon the queue manager restart. Bringing down the Message Broker queue manager will cause the Message Broker to fail. The broker uses the queue manager to output the result of the flow, with this being unavailable the message will be rolled back to be reprocessed (as discussed in “Application monitoring log files” on page 98).

Once the queue manager is made available, the messages waiting on the InterChange Servers queue managers transmit queue will be transferred to the Message Brokers queue manager, and then will be processed by the Message Broker once the broker has been restarted. No messages will be lost.

The Compute Node that contained the ESQL to perform the database insert had the transactionality set to Automatic. This means that the database insert is only committed once the message has been placed onto the output queue; due to this we did not get any duplicate database entries.

If the compute node is set up to commit its database interactions, then it is possible to get duplicate inserts/entries in the database. This would happen if the broker went down mid-way through processing message and the messages had passed through the point where the database transaction had been committed but hadn’t made it to the output node. Due to the message failing to complete the flow it would be rolled back and retried when the broken came back up. On the second attempt of the database transaction, you might receive duplicate inserts and entries.

Test #2
While the events are running, kill the Message Broker queue manager listener.

Result
When the listener was down, it did not prevent any of the components processing messages, and the intercoms between components was unbroken.

The main reason there was no disruption to service was that the monitor was run (which in turn restarted the listener) before the channels timeout. The time interval of the monitors need to be less than that of the channels heart beat to prevent channels going down. (This will not lose messages, but will cause a pause in service.)
Test #3
While the events are running, kill the Message Broker.

Result
The WebSphere InterChange Server and Message Broker adapters are unaffected by the unavailability of the Message Broker queue manager, and will continue to process business processes.

When the broker picks up a message for processing it does not commit the get from the MQ queue until the output has been placed onto a queue. As a result, if there are any messages being processed when the broker goes down, the get on them is released and the message is reprocessed when the broker is running again. As in the above, there are the same database issues regarding committing messages.

Test #4
Message Broker resource group failover

Result
The behavior after the failover was no different to a normal restart of a components. As before, all the messages were successfully processed.

7.3.5 Message Broker adapter resource group

The adapter component of the WebSphere Business Integration solution has been proven to not lose messages during failover while it is working under high load. All requests to the adapter will be held within the messaging layer or application until the adapter is available for processing.

It is noted that the behavior of an adapter with regards to the in-progress requests such as requests that are currently being handled by the adapter, varies from adapter to adapter. Also, note that this behavior is configurable. In the above scenario, the adapter will place in-progress requests into its fault queue during failover. It is recommended that your adapter's failure recovery be configured for your specific business requirements. Such information can be found in the adapter's user guide.

Test #1
Remote adapters resource group failover.
Result
The full resource group was brought up on the failover node where processing continued as expected. All requests to that adapter will be held within the requesting broker's or application until the resource group returns.

In progress requests currently being processed by the adapter will behave based on the adapter itself. Please refer to the specific adapter guide for its behavior for failure recovery.

Test #2
While the batch is running, bring the remote adapter down.

Result
With the remote adapter down, processing of events between the middleware layer and the end application interface will not occur, but this will not cause a total WebSphere Business Integration system failure. The requests that were currently being processed by the adapter when it failed are transferred to the fault queue where they can be re-submitted based on your business requirements. Note that this behavior is adapter specific, and can be changed based on the adapter's failure recovery design. Please refer to the specific adapter guide for that adapter's behavior for failure recovery. All other messages that have not been picked up for processing by the adapter will be unaffected, and shall continue to process when the adapter returns.

Test #3
Remote adapter queue manager down.

Result
If the adapter queue manager fails, all remote adapters will shut down as well. Requests to that adapter will be stored either on the requesting broker's queue manager, or within the end application itself until both the queue manager and adapters are restarted.

In progress, requests currently being processed by the adapter will behave based on the adapter itself. Please refer to the specific adapter guide for its behavior for failure recovery.

Test #4
Remote adapter queue manager listener down.

Result
The remote adapter queue manager listener being down will prevent communication between queue managers, and will also prevent adapter access
to the queues failing all remote adapters. All requests to that adapter will be held within the requesting broker’s or application until the listener returns.

Once the listener is restarted, the channels should automatically re-establish connection between the queue managers.

In progress requests currently being processed by the adapter will behave based on the adapter itself. Please refer to the specific adapter guide for its behavior for failure recovery.

### 7.3.6 Message Broker Active/Active setup

**Test #1**
While the events are running, bring down one of the Message Brokers.

**Result**
MQ default clustering provided a round robin setup. While the broker was down, half the messages were processed as normal, and half were stored in the second brokers queue manager while the broker restarted. This happens as on the messaging level - MQ is running properly. Once the broker had restarted, all of the back-logged messages were processed.

**Test #2**
While the events are running, kill the Message Broker queue manager.

**Result**
The result was very similar to Test #1. The main difference was there were less marooned messages as the only messages back logged where those on the brokers input queue prior to when the queue manager went down. During the time the queue manager was unavailable, all new requests were processed by the second active broker. Once the failed queue manager and broker had been restarted, all the back-logged messages were processed.

**Test #3**
While the events are running, bring down one of the Message Brokers such that it cannot restart and can cause a node down notification.

**Result**
The result of this test was very similar to that of Test #1, except, since the broker did not restart, HACMP ran a notify event after the configured number of unsuccessful restarts. This runs the `stop` script shutting down all components of the resource group. While HACMP initially tried restarting the broker, the queue manager was back-logging messages. When the broker could not be restarted,
and as a result the queue manager was stopped, all of the messages that had been back-logged were marooned with the stopped queue manager.
A few helpful tips

This chapter provides a number of best practices, hints, and tips collected from HA WebSphere Business Integration practitioners out in the field.
8.1 Introduction

This chapter is an attempt to capture a few of the useful tidbits that the specialists who work day-to-day architecting, implementing, and administering highly available WebSphere Business Integration environments have stashed away in their heads. Hopefully, the lessons learned by these specialists will help smooth the reader’s journey through implementing a highly available WebSphere Business Integration solution.

8.2 General recommendations

Some of the advice pertinent to an HA environment in general is:

- Each node in cluster must be sized large enough to support total load under failure conditions. Dynamic process allocation under AIX (shutting down less important processes on the production failover machine) can help lower hardware costs.

- Optimal resource utilization in an HA environment occurs when each node in an Active/Active cluster drives similar loads.

- Use HA clustering for high availability only, not for performance or simplified administration.

- The HA solution in the QA environment should exactly mimic the production environment. This helps to avoid critical production problems and minimizes down time.

- Test individual HA components for online, offline, and failover.

- Attempt to restart the individual HA components (for example, InterChange Server or Message Broker) a minimum of three times on the primary node before failing over to the alternate node.

- Automatic fallback is not recommended. Manual fallback is recommended to minimize system disruption.

- It is recommended that a stock of spare components such as network adapters, cables, etc. be maintained. Related to this is also a need to define an escalation plan to deal with unexpected failures.

- Be sure to create a list identifying who to call when something HA specific goes wrong, or the normal contacts are not available.
8.3 InterChange Server

Some of the best practices specific to the InterChange Server environment are:

- One of the reasons that the InterChange Server process may fail is due to out-of-memory exceptions. This can be minimized by configuring properties such as mem_checker_upperlimits/frequency/sleeptime and MAX_EVENT_Capacity.
- Failover time in a InterChange Server environment can be minimized by:
  - Tuning the repository database and deleting unnecessary repository components
  - Tuning the JVM
  - Configuring deferred recovery for every InterChange Server collaboration, where event sequencing is not required
  - Ensuring that adapter agents are started in parallel. This is critical in deployments with large numbers of adapters as it can impact startup time significantly.
  - Using the minimum number of required disk groups

8.4 WebSphere Business Integration Adapters

Best practices specific to WebSphere Business Integration Adapters are:

- Adapter agents should not be critical HA components in most scenarios. This implies that on adapter failures, InterChange Server, MQ, and Message Broker should continue to run in their respective nodes and not failover.

For example, let us consider a scenario where a SAP adapter in an HA environment is designed as a critical HA component. If the SAP application is shut down for regular maintenance, this will then cause the SAP adapter to shut down, triggering a failover of the InterChange Server. The failover of the InterChange Server will not solve the SAP adapter problem and causes unintentional InterChange Server downtime.
Appendixes
HA scripts

This appendix contains all of the HA scripts that were used in the development of this IBM Redbook.
WebSphere MQ scripts

hacrtmqm

This script is used for creating a queue manager for use in an HA environment.

#!/bin/ksh
# @(#) public/aix/hacrtmqm, supportpacks, MQHA 1.3 03/12/04 14:15:40
#
# (C) Copyright IBM Corporation 2000, 2003
#
# MC63: WebSphere MQ for AIX - Implementing with HACMP
#
# PLEASE NOTE - This script is supplied "AS IS" with no
# warranty or liability. It is not part of
# any product. Please ensure that you read
# and understand it before you run it. Make
# sure that by running it you will not
# overwrite or delete any important data.
#
# DESCRIPTION:
# hacrtmqm - create a QM that will be amenable to High Availability
#
# You only run this on one node - the one with the disks.
# This is the node where the QMGR will be created. At the end
# of this script, the halinkmqm script will be automatically run
# to prepare various directories and control files.
#
# Then, to prepare the remaining cluster nodes, run the
# halinkmqm script on any other nodes which may also host the QMGR. The
# required command line will be displayed at the end of running
# this hacrtmqm command.
#
# Args:
# Two environment variables must be set before calling the script
# $MQHAFSDATA: mounted filesystem which will hold MQSeries data files
# $MQHAFSLOG: mounted filesystem which will hold MQSeries log files
# Other cmdline parms as for crtmqm (eg -ll, -lf ...)
#
# For an example layout, we could name these 2 filesystems as follows:
# - top component (MQHA) for a single point in the root directory
#   where all queue managers on the box will be anchored
# - next component ($qmgr) for the queue manager to be HA'd
# - final component to show the type of data being managed
#
# qmgr1=csq1
# MQHAFSDATA="/MQHA/$qmgr1/data"
# MQHAFSLOG="/MQHA/$qmgr1/log"
# You do not need to follow these conventions, but this set will work, and # is fairly easy to comprehend.

# Sanity checks
if [ -z "$MQHAFSDATA" -o -z "$MQHAFSLOG" ]
then
    echo "Filesystem environment variables MQHAFSDATA and/or MQHAFSLOG not set"
    exit 1
fi

if [ `id -u` -ne 0 ]
then
    echo "Must be running as root"
    exit 1
fi

# Check we can access the filesystems, and if so, set the permissions on # the directories. Each queue manager should have its own set of filesystems # and so the .../qmgrs subdirectory should not already exist (or be empty) in the # data directory. We'll use that fact later on to discover the name of # the directory used for the actual qmgr information.
curdir=`pwd`
for fs in $MQHAFSDATA $MQHAFSLOG
doi
    cd $fs
    if [ $? -ne 0 ]
    then
        echo "Cannot access filesystem $fs"
        exit 1
    else
        if [ $fs = $MQHAFSDATA ]
        then
            mkdir $MQHAFSDATA/qmgrs >/dev/null 2>&1
            i=`ls -a $MQHAFSDATA/qmgrs | wc -l`
            if [ $i -ne 2 ]
            then
                echo "Directory "$MQHAFSDATA/qmgrs" is not empty. It should be!"
                exit 1
            fi
        fi
chown -R mqm:mqm $fs
chmod -R 2775 $fs
fi
done

# Do the actual creation of the queue manager. Needs to be run # under mqm authority, as root may not be in the mqm group # - The MQSPREFIX variable forces the data onto the 1st filesystem
# - The "-ld" option forces logs onto the 2nd filesystem
MQSPREFIX=$MQHAFSDATA su mqm -c "crtmqm -ld $MQHAFSLOG $*"
rc=$?

# We weren't passed the qmgr name directly, so we need to find it from
# the input parms - it's on the end of the line.
while [ ! -z "$1" ]
do
  qmgr=$1
  shift
done

if [ $rc -eq 0 ]
then
  cd $curdir
  # Find the mangled directory name
  qmgrdir="ls $MQHAFSDATA/qmgrs" 
  echo "The halinkmqm script will now be run on this machine. On other"
  echo "machines which will potentially takeover this queue manager you"
  echo "must run the following command while logged on as \"mqm\"."
  echo "\n\t halinkmqm $qmgr $qmgrdir $MQHAFSDATA"
  # And actually do it.
  su mqm -c "/usr/mqm/hascripts/halinkmqm $qmgr $qmgrdir $MQHAFSDATA"
fi
Appendix A. HA scripts

hadltmqm

This script is used for deleting MQ queue managers that have been created using hacrtm.

#!/bin/ksh
# @(#) public/aix/hadltmqm, supportpacks, MQHA
#
# (C) Copyright IBM Corporation 2000, 2003
#
# MC63: WebSphere MQ for AIX - Implementing with HACMP
#
# PLEASE NOTE - This script is supplied "AS IS" with no
# warranty or liability. It is not part of
# any product. Please ensure that you read
# and understand it before you run it. Make
# sure that by running it you will not
# overwrite or delete any important data.
#
# DESCRIPTION:
# To delete a queue manager on the owning node. This script
# will also delete subdirectories on standby nodes, but leaves
# the mqs.ini file alone. Administrators will have to do the
# edit manually (for now).
#
# Arguments:
# $1: Queue Manager name

qmgr=$1
d=``/usr/mqm/hascript/hamqm_qm_directory $qmgr``
if [ -z "d" ]
then
  echo "Could not find information about queue manager "$1"
  exit 1
fi

b=`basename $d`
if [ -r $d ]
then
  # Ensure the queue manager has stopped
  echo "ping qmgr" | runmqsc $qmgr
  # Return code 0 is a normal end, so the qmgr is still running
  if [ $? -eq 0 ]
  then
    echo "Queue Manager appears to still be running. Please stop it before"
    echo "trying to delete it."
    exit 1
  fi
fi
# Delete the symlinks, and replace with real directories. This
# is so the tree deletion code doesn't
# throw up half-way through.
first_time="yes"
for topdir in @ipcc @qmpersist
do
cd $d
for subdir in esem isem msem ssem shmem
do
if [ $first_time = "yes" ]
then
rm -f $subdir
mkdir $subdir
chown mqm:mqm $subdir
chmod 775 $subdir
fi
rm -f $topdir/$subdir
mkdir $topdir/$subdir
chown mqm:mqm $topdir/$subdir
chmod 775 $topdir/$subdir
done
chmod a+w ssim
chmod a+w $topdir/ssim
first_time="no"
done

mkdir @ipcc/shmem/PerQUEUE
mkdir shmem/PerQUEUE
#
# Then really delete the queue manager

dltmqm $qmgr
else
echo "Directories will be deleted, but you will need to edit"
echo "the /var/mqm/mqs.ini file yourself, to delete the queue manager stanza"
fi
#
# If everything is OK, delete the IPC subtree, and the symlink out of the
# 'normal' path to the queue manager that we created in hacrtmqm.
if [ $? -eq 0 ]
then
rm -rf /var/mqm/ipc/$b
rm -f /var/mqm/qmgrs/$b
fi
halinkmqm

This script is for creating the files and directories needed on a failover node for a queue manager to failover to it.

#!/bin/ksh
# @(#) public/aix/halinkmqm, supportpacks, MQHA
#
# (C) Copyright IBM Corporation 2000, 2003
#
# MC63: WebSphere MQ for AIX - Implementing with HACMP
#
# PLEASE NOTE - This script is supplied "AS IS" with no
# warranty or liability. It is not part of
# any product. Please ensure that you read
# and understand it before you run it. Make
# sure that by running it you will not
# overwrite or delete any important data.
#
# DESCRIPTION:
#   halinkmqm - relink a QM tree so that it works for an HA environment
#
# This script is automatically run by the hacrtmqm script that
# has been run to create the QM on the node with the disks.
#
# You must run halinkmqm on all OTHER nodes that may host the
# QMGR. Don't run it again on the node which already has
# the QMGR.
#
# Args:
#       $1: Qmgr name
#       $2: Mangled qmgr directory name -- may or may not be the same as qmgr
#       $3: Shared Prefix               -- e.g. /MQHA/<qmgr>/data
#
if [ `whoami` != "mqm" ]
then
  echo "Need to be running as mqm"
  exit 1
fi

if [ $# -ne 3 ]
then
  echo "Usage: halinkmqm QMgrName QMgrDataDirSubdirectory QMgrDataDirPrefix"
  echo "    eg halinkmqm ha.csq1 ha!csq1 /MQHA/ha.csq1/data"
  exit 1
fi

# Make sure others can read the directory we're about to create
umask 022
# Variables pointing to the original directory trees and the local IPC versions
ipcbase="/var/mqm/ipc/$2"
ipcorig="$3/qmgrs/$2"

# Create the IPC directory (which will quite possibly already exist)
# And then create the QMgr-specific subdirectory (which shouldn't)
mkdir /var/mqm/ipc >/dev/null 2>&1
chmod 775 /var/mqm/ipc

mkdir $ipcbase
mkdir $ipcbase/@ipcc
chmod 775 $ipcbase $ipcbase/@ipcc
mkdir $ipcbase/@qmpersist
chmod 775 $ipcbase $ipcbase/@qmpersist

# The link from /var/mqm/qmgrs is not necessary, but some programs
# may assume a default path to the queue manager so we'll create it anyway
ln -fs $3/qmgrs/$2 /var/mqm/qmgrs/$2

# The /var/mqm/ipc directory is associated with a machine, not
# a queue manager, so it is never failed over to a standby node.
#
# We need to create the ipc subdirs and set permissions on them.
first_time="yes"
cd $ipcbase
for topdir in @ipcc @qmpersist
do
    for subdir in esem isem msem shmem ssem shmem/PerQUEUE
do
        if [ $first_time = "yes" ]
        then
            mkdir $subdir
            chown mqm:mqm $subdir
            chmod 775 $subdir
        fi
        mkdir $topdir/$subdir
        chown mqm:mqm $topdir/$subdir
        chmod 775 $topdir/$subdir
done
    done
    chmod a+w ssem
    chmod a+w $topdir/ssem
    first_time="NO"
done

if [ -r $3/qmgrs/$2/qm.ini ]
then
    # We're running on the master node that owns the queue manager
# so we will create symlinks back to /var/mqm/ipc subdirs
for topdir in @ipcc @qmpersist
do
  for subdir in esem isem msem shmem ssem
do
    rm -fr $ipcorig/$subdir
    rm -fr $ipcorig/$topdir/$subdir
    ln -fs $ipcbase/$subdir $ipcorig/$subdir
    ln -fs $ipcbase/$topdir/$subdir $ipcorig/$topdir/$subdir
done
done
else
  # We're running on a standby node, so all we have to do is to
  # update the config file that tells us where the queue manager lives
  cat >> /var/mqm/mqs.ini <<EOF
  QueueManager:
    Name=$1
    Prefix=$3
    Directory=$2
  EOF
fi

# HACMP/ES does not seem to support passing parameters to application
# monitoring scripts. So we need to create a unique monitor script for
# each queue manager. And as we also need to switch userids from root
# to mqm, the wrapper technique works quite well. The auto-created script
# is called /MQHA/bin/hamqm_applmon.<qmgr name> and it is this script
# which is entered in the HACMP/ES configuration panels.
apmon=/usr/mqm/hascripts/hamqm_applmon.$1
cat > $apmon << EOF
#!/bin/ksh
su mqm -c /usr/mqm/hascripts/hamqm_applmon_su $1
EOF
chmod +x $apmon
This script is used to find the location of the files for a MQ queue manager.

#!/bin/ksh
# @(#) public/aix/hamqm_qm_directory, supportpacks, MQHA, MQHA-001108
1.2 00/11/08 09:57:13
#
# (C) Copyright IBM Corporation 2000
#
# MC63: MQSeries for AIX - Implementing with HACMP
#
# PLEASE NOTE - This script is supplied "AS IS" with no warranty or liability. It is not part of any product. Please ensure that you read and understand it before you run it. Make sure that by running it you will not overwrite or delete any important data.
#
# DESCRIPTION:
# /MQHA/bin/hamqm_qm_directory <qmname>
# Accepts the name of a QM and returns (prints) the fully qualified path of the (mangled) QM directory for that QM.
#
NAME_TO_FIND=$1

# The regular expression in the next line contains a tab character
# Edit only with tab-friendly editors.
cat /var/mqm/mqs.ini | grep -v "^[ ]*#" | \
    awk -F= -v name_to_find=$NAME_TO_FIND \
    ' BEGIN {
        in_qmstanza="no";
        correct_stanza="no";
        name="not set";
        prefix="not set";
        directory="not set";
    }

    # body
    {
        if (in_qmstanza="no" && $1=="QueueManager:"
        {
            in_qmstanza="yes";
        }
        else if (in_qmstanza="yes")
        {
            if ($1==" Name")
            {
                # body
                
            }
#print "found name "$2
name=$2;
}
if ($1=="   Prefix")
{
    #print "found prefix "$2
prefix=$2;
}
if ($1=="   Directory")
{
    #print "found directory "$2
directory=$2;
}
if ( match($1,\[.\]*:$) )   # start of a stanza
{
    if (name==name_to_find)
    {
        result=sprintf("%s/qmgrs/%s",
                        prefix,directory);
        #print "result set to "result
    }
    if ($1=="QueueManager:")
    {
        in_qmstanza="yes";
        name="not set";
        prefix="not set";
        directory="not set";
    }
    else
    {
        in_qmstanza="no";
        name="not set";
        prefix="not set";
        directory="not set";
    }
}
}
END {
    if (in_qmstanza=="yes")
    {
        if (name==name_to_find)
        {
            result=sprintf("%s/qmgrs/%s",prefix,directory);
            #print "result set to "result
        }
    }
    if (result!="not set")
    {

print result;
}
else
{
  print "not found";
}
}
hamqm_running

This script is used to check if the MQ queue manager is running.

#!/bin/ksh
# @(#) public/aix/hamqm_running, supportpacks, MQHA, MQHA-001108 1.3
00/11/08 09:57:15
#
# (C) Copyright IBM Corporation 2000
#
# MC63: MQSeries for AIX - Implementing with HACMP
#
# PLEASE NOTE - This script is supplied "AS IS" without warranty or liability. It is not part of any product. Please ensure that you read and understand it before you run it. Make sure that by running it you will not overwrite or delete any important data.
#
# DESCRIPTION:
# /MQHA/bin/hamqm_running <qmname>
# This script is used by other hamqm scripts to return a boolean (1 or 0) indicating whether the named QM is running.
QM=$1

# This function will test whether the QM is running and returns 1 if it is, or 0 if it is not.

# Find the status file....
QMDIR=`/usr/mqm/hascripts/hamqm_qm_directory ${QM}`

# Parse the status file to find the CurrentStatus for the Queue Manager. The following awk script selects only QueueManagerStatus stanzas and ignore others, such as CommandServerStatus.
# The regular expression in the next line contains a tab character Edit only with tab-friendly editors.
RESULT="/usr/bin/cat ${QMDIR}/qmstatus.ini | \
grep -v "^[[]][^*#"] | awk -F= \'
BEGIN {
    expecting_status="no";
    status="not_found";
}

# body
{
    if ($1="QueueManagerStatus:")
    {
}
expecting_status="yes";
}
else if (expecting_status=="yes")
{
    if ($1=="  CurrentStatus")
    {
        status=$2;
        expecting_status="no";
    }
    if ( match($1,"[.]*:$") )  # start of a stanza
    {
        expecting_status="no";
    }
}
}
END {
    print status;
}
' 2>&1`

if [ $RESULT = Running ]
then
    print 1
else
    print 0
fi
hamqm_start

This script is used to start \texttt{hamqm\_start\_su} with the \texttt{mqm} user.

\begin{verbatim}
#!/bin/ksh
# @(#) public/aix/hamqm_start, supportpacks, MQHA, MQHA-001108          1.1
00/11/03 08:06:37
#
# (C) Copyright IBM Corporation 2000
#
# MC63: MQSeries for AIX - Implementing with HACMP
#
# PLEASE NOTE - This script is supplied "AS IS" with no
# warranty or liability. It is not part of
# any product. Please ensure that you read
# and understand it before you run it. Make
# sure that by running it you will not
# overwrite or delete any important data.
#
# DESCRIPTION:
# Because HACMP runs scripts with root authority, and MQSeries prefers
# to be managed by the mqm userid, we've got here a simple wrapper
# which will switch to the correct account before running the real
# command.
#
su mqm -c /usr/mqm/hascripts/hamqm_start_su $*
\end{verbatim}
hamqm_start_su

This script is used to start a MQ queue manager under HA.

#!/bin/ksh
# @(#) public/aix/hamqm_start_su, supportpacks, MQHA 1.2 00/11/07
18:16:50
# (C) Copyright IBM Corporation 2000
#
# MC63: MQSeries for AIX - Implementing with HACMP
#
# PLEASE NOTE - This script is supplied "AS IS" with no
# warranty or liability. It is not part of
# any product. Please ensure that you read
# and understand it before you run it. Make
# sure that by running it you will not
# overwrite or delete any important data.
#
# DESCRIPTION:
# /MQHA/bin/hamqm_start_su <qm-name>
#
# This script should do whatever it can to start the QM (name
# passed as arg).
#
# The script has to be robust regarding making no assumptions about
# the current state of the QM, and cleaning up any damage that might
# have been done when the QM last terminated - which could have included
# termination of the EC, which can leave logger and checkpoint processes
# lying around, preventing a restart. Such orphans need to be cleaned up.
# There's no need to look for orphaned agents as these are child processes
# of the EC and stay in the same process group, so they will have been
# cleaned up already.
#
# The script should be run by the mqm user. A wrapper script is also
# provided to switch userids from HACMP's root authority.

# Only argument is the queue manager name. Save it as QM variable
QM=$1

if [ -z "$QM" ]
then
  echo "hamqm_start_su: ERROR! No queue manager name supplied"
  exit 1
fi

# Check that the QM doesn't already think it's running
# The best way to do this is to look for any processes that
# may be vestiges of a previous run of the QM and kill them.
There's little point enquiring what the contents of the qmstatus.ini file are - since we should not trust them anyway at this point - even if it said the QM is running we have to be robust to the fact that if the EC died first it may not have had a chance to update the status file. Therefore, just torch the QM processes, if any. This is exactly the same code that we use for the brutal (terminate) severity of stop command.

The regular expression in the next line contains a tab character. Edit only with tab-friendly editors.

```
# The regular expression in the next line contains a tab character
# Edit only with tab-friendly editors.
srchstr="( |-m)\$QM\[ \]*.*$"
for process in amqpcsea amqhasmx amqharmx amqz1lp0 \  
amqzlaa0 runmqchi amqrrmfa amqzxma0 amqrmppa \  
amqzfuma amqzdmaa  
do  
   ps -ef | grep $process | grep -v grep | \  
      egrep "$srchstr" | awk '{print $2}' | \  
      xargs kill -9

done
```

Should now be safe to start the QM

```
strmqm ${QM}
```

```
rc=+$
# Fire post-online script if good return from strmqm.
if [ $rc -eq 0 ]
then  
# Launch post-online script in background. Note that we  
# run this entire script under the instance userid (e.g.  
# "mqm") as specified by the administrator. The post-online  
# script is therefore run under this same id.  
#  
# This is always done in the background so we don't delay the  
# HA methods, which need to stay responsive and may need to  
# start other instances.  
#  
if [ -x /MQHA/bin/rc.local ]
then  
   echo "hamqm_start: launch post_online script for ${QM}"
   /MQHA/bin/rc.local ${QM} post_online &
   # exit code from online script is deliberately ignored
fi
fi

exit $rc
hamqm_status

This script checks the status of a MQ queue manager in the qmstatus.ini file.

#!/bin/ksh
# @(#) public/aix/hamqm_status, supportpacks, MQHA, MQHA-001108 1.2
00/11/08 09:57:18
#
# (C) Copyright IBM Corporation 2000
#
# MC63: MQSeries for AIX - Implementing with HACMP
#
# PLEASE NOTE - This script is supplied "AS IS" with no
# warranty or liability. It is not part of
# any product. Please ensure that you read
# and understand it before you run it. Make
# sure that by running it you will not
# overwrite or delete any important data.
#
# DESCRIPTION:
# /MQHA/bin/hamqm_status <qmname>
# This function will return the status of the QM as reported in
# the qmstatus.ini file. It is used by other scripts, and is not
# called directly by users or HACMP.
#
QM=$1

# Find the status file....
QMDIR="/usr/mqm/hascripts/hamqm_qm_directory ${QM}"

STATUS="/usr/bin/cat ${QMDIR}/qmstatus.ini | \
    grep "CurrentStatus" | awk -F= '{print $2}'"

print $STATUS
hamqm_stop

This script is used to start hamqm_stop_su with the mqm user.

#!/bin/ksh
# @(#) public/aix/hamqm_stop, supportpacks, MQHA, MQHA-001108          1.2
00/11/06 16:58:23
#
# (C) Copyright IBM Corporation 2000
#
# MC63: MQSeries for AIX - Implementing with HACMP
#
# PLEASE NOTE - This script is supplied "AS IS" with no
# warranty or liability. It is not part of
# any product. Please ensure that you read
# and understand it before you run it. Make
# sure that by running it you will not
# overwrite or delete any important data.
#
# DESCRIPTION:
# Because HACMP runs scripts with root authority, and MQSeries prefers
# to be managed by the mqm userid, we've got here a simple wrapper
# which will switch to the correct account before running the real
# command.

su mqm -c /usr/mqm/hascripts/hamqm_stop_su $*
This script stops the MQ queue manager.

```
#!/bin/ksh
# @(#) public/aix/hamqm_stop_su, supportpacks, MQHA 1.3 00/11/07
18:16:51
#
# (C) Copyright IBM Corporation 2000
#
# MC63: MQSeries for AIX - Implementing with HACMP
#
# PLEASE NOTE - This script is supplied "AS IS" with no
# warranty or liability. It is not part of
# any product. Please ensure that you read
# and understand it before you run it. Make
# sure that by running it you will not
# overwrite or delete any important data.
#
# DESCRIPTION:
# /MQHA/bin/hamqm_stop_su <qmname> <timeout-secs>
#
# Stops the QM. If no timeout has been provided ($2) then the
# endmqm is issued synchronously and this script blocks until it
# has completed. If a timeout is provided, this script uses a
# phased approach to stopping the QM - each phase is given
# timeout-secs to complete and is then escalated to the next
# phase. The first phase uses "immediate", the next uses
# "preemptive", and the third (and final) phase uses kill -9 on
# the QM processes.
#
# Run this script as the userid that should control the QM. e.g.
# su - mqm -c /usr/sbin/cluster/utils/hamqm_stop_su <qmname>

QM=$1
TIMEOUT=$2

if [ -z "$QM" ]
then
    echo "hamqm_stop: ERROR! No queue manager name supplied"
    exit 1
fi

if [ -z "$TIMEOUT" ]
then
    echo "hamqm_stop: ERROR! No timeout specified"
    exit 1
fi
```
echo "hamqm_stop_su is running: "$QM" $TIMEOUT"

# Check to see if the QM is already stopped. If so, just make # sure no processes are lying around.
# online=`/usr/mqm/hascripts/hamqm_running ${QM}`
if [ ${online} != "1" ]
then
  # QM is reported as offline; ensure no processes remain

  # Note that this whole script should be executed under su, # which is why there's no su in the following loop.

  # The regular expression in the next line contains a tab character # Edit only with tab-friendly editors.
srchstr="( |-m)$QM[ ].*$"
for process in amqpcsea amqhasmx amqharmx amqz1lp0 \
  amqz1la0 runmqchi amqrrmfa amqzxma0 amqrmpa \
  amqzfuma amqzdmaa
do
  ps -ef | grep $process | grep -v grep | \
    egrep "$srchstr" | awk '{print $2}'| \
    xargs kill -9
done
exit 0
fi

# Invoke Pre-Offline user exit, if one exists
if [ -x /MQHA/bin/rc.local ]
then
  echo "hamqm_stop_su: launch pre-offline script for ${QM}" /MQHA/bin/rc.local ${QM} pre_offline &
  # Exit code from pre_offline script is deliberately ignored
fi

for severity in immediate preemptive terminate
do
  # Issue the stop method in the background - we don't # want to risk having it hang us up, indefinitely. We # want to be able to run our TIMEOUT timer at # the same time to give up on the attempt, and try a # more forceful version. If the kill version fails then # there is nothing more we can do here anyway.
  echo "hamqm_stop_su: attempting ${severity} stop of ${QM}"
case $severity in
    immediate)
# Minimum severity of stop is an Immediate stop
# i.e. we sever connections - HACMP should not be delayed
# by clients
endmqm -i ${QM} &

preemptive)
# This is a preemptive stop. We have already tried -i.
endmqm -p ${QM} &

terminate)
# This is a brutal means of mopping up QM processes.
# We stop the processes in accordance with the order
# specified in Appendix E of the System Management Guide,
# except that surely the repository controller
# should go before the EC, so I have reversed those entries.

# The regular expression in the next line contains a tab character
# Edit only with tab-friendly editors.
srchstr="( |-m)$QM[ ]*.*$"
for process in amqpcsea amqhasmx amqharmx amqzllp0 \
  amqzlaa0 runmqchi amqrrmfa amqzxma0 amqrmppa \
  amqzfuma amqzdmaa
do
  ps -ef | grep $process | grep -v grep | \ 
  egrep "$srchstr" | awk '{print $2}' | \ 
  xargs kill -9
done
esac

echo "hamqm_stop_su: waiting for ${severity} stop of ${QM} to complete"
TIMED_OUT=yes
SECONDS=0
while ((( $SECONDS < ${TIMEOUT} )))
do
  # Invoke hamqm_running utility
  online="/usr/mqm/hascripts/hamqm_running ${QM}"
  case $online in
    "1"
      # It's still running...wait for timeout
      ;;
    "0"
      # EC has updated status, but wait for
      # EC to cleanup and terminate. If it

# fails to terminate inside 10 secs
# then escalate to next level of
# stop processing.
echo "hamqm_stop_su: \{QM\} is stopping"
TIMED_OUT=yes
i=0
while [ $i -lt 10 ]
do
    # Check for EC termination
    # The regular expression in the next line contains a tab character
    # Edit only with tab-friendly editors.
srchstr="( |-m)\{QM\}[ ].*$
cnt=`ps -ef | grep amqzxma0 | grep -v grep | \n    egrep "$srchstr" | awk '{print $2}' | wc -l`
i=`expr $i + 1`
sleep 1
    if [ $cnt -eq 0 ]
    then
        TIMED_OUT=no
        break
    fi
done
break
;;
*

*)
    # Bad result from hamqm_running
    echo "hamqm_stop_su: invalid result (\{online\}) from hamqm_running"
    exit 1
    break
    ;;
esac

sleep 1
done # timeout loop

if [ ${TIMED_OUT} = "yes" ]
then
    continue        # to next level of urgency
else
    break           # instance is stopped, job is done
fi
done # next phase of stop
WBI Message Broker

hamqsicreatebroker

This script is used to create an HA broker.

#!/bin/ksh
# @(#) public/aix/hamqsicreatebroker, SupportPacs, H000, H000-010403
1.1 01/03/30 15:10:27
#
# (C) Copyright IBM Corporation 2001
#
# IC61: Configuring MQSeries Integrator for AIX with HACMP
#
# PLEASE NOTE - This script is supplied "AS IS" with no
# warranty or liability. It is not part of
# any product. Please ensure that you read
# and understand it before you run it. Make
# sure that by running it you will not
# overwrite or delete any important data.
#
#
# Module: hamqsicreatebroker
#
# Args:
# Arguments are the same as for mqsicreatebroker
#
# Description:
# This script creates an MQSI broker so that it can run
# in an HA cluster.
#
# A broker relies on a database, which is run within
# a database instance. This script assumes that the
# database instance name is the same as the userid
# under which the instance is run.
#
# Runs as root and uses su when creating broker
#
#
# Must be run as root
if [ `id -u` -ne 0 ]
then
echo "Must be running as root"
exit 1
fi
# Save the supplied arguments - we'll need it below
ORIG_ARGS=$*

# Parse the args for $BROKER, $QM, $DBINST & $MQUSER
BROKER=$1
# Parse the remaining args to find the QM, DBINST and MQUSER
shift # to get past the broker name argument
while getopts :i:q:n: arguments
done

case $arguments in
  i) MQUSER=$OPTARG;;
  q) QM=$OPTARG;;
  n) DBINST=$OPTARG;;
  :) echo "Missing argument to $OPTARG switch";;
  \?) shift;; # ignore other (valid) switch and shift past its argument
esac

done

# Test that we have QM, DBINST and MQUSER args set
if [ -z "$QM" ]
then
  echo "hamqsicreatebroker: ERROR! No queue manager name supplied"
  echo "hamqsicreatebroker: Specify parameters as with mqsicreatebroker"
  exit 1
fi

if [ -z "$DBINST" ]
then
  echo "hamqsicreatebroker: ERROR! No database instance name supplied"
  echo "hamqsicreatebroker: Specify parameters as with mqsicreatebroker"
  exit 1
fi

if [ -z "$MQUSER" ]
then
  echo "hamqsicreatebroker: ERROR! No userid supplied"
  echo "hamqsicreatebroker: Specify parameters as with mqsicreatebroker"
  exit 1
fi

# Create the broker
su - $MQUSER -c "mqsicreatebroker $ORIG_ARGS"
if [ $? -ne "0" ]
then
  echo "hamqsicreatebroker: Error from mqsicreatebroker command"
  exit 1
fi

# Now find out where the datapath of the QM is.
DATAPATH=`./hamqsi_qm_datapath $QM`

echo "Broker directories will be moved to be under $DATAPATH"

# The brokers/$BROKER and registry/$BROKER directories for this broker
# will be moved to the QM data directory, which is on shared disk. Symbolic
# links are then created.
#
curdir=`pwd`

cd /var/mqsi/brokers
echo "hamqsicreatebroker: Moving brokers/$BROKER directory to $QM/data
directory"
if [ ! -d $DATAPATH/brokers ]
then
  mkdir $DATAPATH/brokers
fi
cp -Rp $BROKER $DATAPATH/brokers
rm -fr $BROKER
ln -fs $DATAPATH/brokers/$BROKER $BROKER

cd /var/mqsi/registry
echo "hamqsicreatebroker: Moving registry/$BROKER directory to $QM/data
directory"
if [ ! -d $DATAPATH/registry ]
then
  mkdir $DATAPATH/registry
fi
cp -Rp $BROKER $DATAPATH/registry
rm -fr $BROKER
ln -fs $DATAPATH/registry/$BROKER $BROKER

cd $curdir

# Create an Application Monitor
# HACMP/ES does not seem to support passing parameters to application
# monitoring scripts. So we need to create a unique monitor script for
# the combination of broker, qm and dbinst. The auto-created script
# is called /MQHA/bin/hamqsi_applmon.$BROKER and it is this script which
# is entered in the HACMP/ES configuration panels.
#
apmon=/MQHA/bin/hamqsi_applmon.$BROKER
cat > $apmon << EOF
#!/bin/ksh
/MQHA/bin/hamqsi_monitor_broker_as $BROKER $QM $DBINST $MQUSER
EOF
EOF
chmod +x $apmon

echo "hamqsicreatebroker:"
echo "\t You need to run hamqsiaddbrokerstandby $BROKER $QM $MQUSER on standby nodes"

exit 0
This script is used to prepare a standby node for a broker.

#!/bin/ksh
# @(#) public/aix/hamqsiaddbrokerstandby, SupportPacs, H000, H000-010403
1.1 01/03/30 15:10:25
#
# (C) Copyright IBM Corporation 2001
#
# IC61: Configuring MQSeries Integrator for AIX with HACMP
#
# PLEASE NOTE - This script is supplied "AS IS" with no
# warranty or liability. It is not part of
# any product. Please ensure that you read
# and understand it before you run it. Make
# sure that by running it you will not
# overwrite or delete any important data.
#
# Module: hamqsiaddbrokerstandby
#
# Args:
# broker = name of broker
# qm = name of queue manager broker uses
# mquser = user id under which queue manager and broker run
#
# DESCRIPTION:
# This script configures a standby node so that it can
# host the broker
#
# You run this on any nodes which may host the broker, except the
# one on which the broker exists.
#
# This script must be run as root.

# Check running as root
if [ `id -u` -ne 0 ]
then
  echo "Must be running as root"
  exit 1
fi

# Parse the args for $BROKER, $QM & $MQUSER
BROKER=$1
QM=$2
MQUSER=$3
# Test that all necessary parameters are set

if [ -z "$BROKER" ]
then
    echo "hamqsiaddbrokerstandby: ERROR! No broker name supplied"
    echo "    Usage: hamqsiaddbrokerstandby <BROKER> <QM> <MQUSER>"
    exit 1
fi

if [ -z "$QM" ]
then
    echo "hamqsiaddbrokerstandby: ERROR! No queue manager name supplied"
    echo "    Usage: hamqsiaddbrokerstandby <BROKER> <QM> <MQUSER>"
    exit 1
fi

if [ -z "$MQUSER" ]
then
    echo "hamqsiaddbrokerstandby: ERROR! No userid supplied"
    echo "    Usage: hamqsiaddbrokerstandby <BROKER> <QM> <MQUSER>"
    exit 1
fi

# Perform a check that the broker does not already exist on this
# machine - and that the /var/mqsi/brokers and /var/mqsi/registry
# directories do not already have entries (either real or links)
# for this broker. If they have, then warn the user to clean up
# and retry.

if [ -d /var/mqsi/brokers/$BROKER -o -h /var/mqsi/brokers/$BROKER ]
then
    echo "hamqsiaddbrokerstandby: "
    echo "    Broker already has an entry in /var/mqsi/brokers"
    echo "    Please ensure this is a standby machine and remove the"
    echo "    $BROKER directory or symlink under /var/mqsi/brokers"
    exit 1
fi

if [ -d /var/mqsi/registry/$BROKER -o -h /var/mqsi/registry/$BROKER ]
then
    echo "hamqsiaddbrokerstandby: "
    echo "    Broker already has an entry in /var/mqsi/registry"
    echo "    Please ensure this is a standby machine and remove the"
    echo "    $BROKER directory or symlink under /var/mqsi/registry"
    exit 1
fi

# Now find out where the datapath of the QM is.
DATAPATH="./hamqsi_qm_datapath $QM"
if [ -z "$DATAPATH" ]
then
    echo "hamqsiaddbrokerstandby:"
    echo "\t Directory structure for $QM not found in mqs.ini"
    echo "\t Please check your configuration is ready to issue this command"
    exit 1
fi

echo "Symbolic links will be created which link to directories under $DATAPATH"

# Create the symlinks to the broker directories from this machine.
# The brokers and registry directories for this broker are in the QM data
# directory, which is on shared disk. They are not currently mounted on
# this machine, but that's OK. Create symlinks to the directories from
# /var/mqsi/brokers and /var/mqsi/registry.
#
curdir=`pwd`
cd /var/mqsi/brokers
 echo "hamqsiaddbrokerstandby: Creating symbolic link to brokers/$BROKER"
 ln -fs $DATAPATH/brokers/$BROKER $BROKER
 cd /var/mqsi/registry
 echo "hamqsiaddbrokerstandby: Creating symbolic link to registry/$BROKER"
 ln -fs $DATAPATH/registry/$BROKER $BROKER
 cd $curdir

# Create an Application Monitor on this machine.
# HACMP/ES does not support passing parameters to application
# monitoring scripts. So we need to create a unique monitor script for
# the combination of broker and QM. The auto-created script
# is called /MQHA/bin/hamqsi_applmon.$BROKER. It is this script which
# should be entered in the HACMP/ES configuration panels.
#
echo "hamqsiaddbrokerstandby: Creating application monitor script " \
"/MQHA/bin/hamqsi_applmon.$BROKER"
apmon=/MQHA/bin/hamqsi_applmon.$BROKER
cat > $apmon « EOF
#!/bin/ksh
/MQHA/bin/hamqsi_monitor_broker_as $BROKER $QM $MQUSER
EOF
chmod +x $apmon

echo "hamqsiaddbrokerstandby: Completed successfully"

# Remind the administrator to synchronise the users and ODBC config on
# on this node.
 echo "hamqsiaddbrokerstandby:"
 echo "\t For broker $BROKER to run correctly on this node you need to "

156   Highly Available WebSphere Business Integration Solutions
echo "\t * Ensure that user $MQUSER exists on this machine, with "
echo "\t matching username, uid, password and groups as on the node "
echo "\t on which broker $BROKER was created. "
echo "\t * Ensure that the /var/mqsi/users/$MQUSER exists and matches "
echo "\t the directory of the same name on the machine on which "
echo "\t broker $BROKER was created."

echo "\t * Ensure that the /var/mqsi/odbc/.odbc.ini file contains "
echo "\t an entry for the database used by broker $BROKER."
This script is used to delete a broker created using `hamqsicreatebroker`.

```
#!/bin/ksh
# @(#) public/aix/hamqsideletebroker, SupportPacs, H000, H000-010403
1.1  01/03/30 15:10:30
#
# (C) Copyright IBM Corporation 2001
#
# IC61: Configuring MQSeries Integrator for AIX with HACMP
#
# PLEASE NOTE - This script is supplied "AS IS" with no warranty or liability. It is not part of any product. Please ensure that you read and understand it before you run it. Make sure that by running it you will not overwrite or delete any important data.
#
# Module: hamqsideletebroker
#
# Args:
#   broker  = name of broker to delete
#   mquser  = userid under which broker service runs
#   dbinst  = database instance name use by broker
#
# Description:
#   This script deletes a broker. It has to be run on the node on which the broker is currently hosted. The script detects whether the broker is currently present or whether this node is acting only as a standby. If the broker is present it is deleted. If it is not present an error message is generated.
#
# The broker must be removed from HACMP control and stopped before calling this script.
#
# Runs as root and uses su when deleting broker
#
#
# Check running as root
if [ `id -u` -ne 0 ]
then
  echo "Must be running as root"
  exit 1
fi

# Parse the args for BROKER, MQUSER
```
BROKER=$1
MQUSER=$2
DBINST=$3

if [ -z "$BROKER" ]
then
    echo "hamqsideletebroker: ERROR! No broker supplied"
    echo "Usage: hamqsideletebroker <BROKER> <MQUSER> <DBINST>"
    exit 1
fi

if [ -z "$MQUSER" ]
then
    echo "hamqsideletebroker: ERROR! No userid supplied"
    echo "Usage: hamqsideletebroker <BROKER> <MQUSER> <DBINST>"
    exit 1
fi

if [ -z "$DBINST" ]
then
    echo "hamqsideletebroker: ERROR! No dbinst supplied"
    echo "Usage: hamqsideletebroker <BROKER> <MQUSER> <DBINST>"
    exit 1
fi

# Up to this point, this script has assumed that the Database Instance id
# and Database Instance name are the same. If this is not the case for the
# database manager you are using then modify the args to this script and
# the following line. The remainder of the script will treat DBUSER and
# DBINST as the user id and instance name respectively.
DBUSER=$DBINST

# Work out whether the broker is locally present
# Try and cd to broker directory
curdir=`pwd`
cd /var/mqsi/brokers/$BROKER
rc=$?
if [ $rc -ne 0 ]
then
    echo "hamqsideletebroker:"
    echo "\tBroker is not local"
    echo "\tYou should only run this script on the node which has the broker"
    exit 1
fi
cd $curdir

# The brokers/$BROKER and registry/$BROKER directories for this broker
# are mounted and are accessed by symlinks. Destroy the symlinks,
# move the directories back to internal disk, clearing up the shared disk.
echo "hamqsideletebroker: Moving brokers/$BROKER and registry/$BROKER " \ 
"back from QM data directory"
curdir=`pwd`

# Handle the brokers/$BROKER directory
cd /var/mqsi/brokers
sharedir=`ls -e $BROKER | awk '{print $NF}' ` if [ -z $sharedir ] then
  echo "hamqsideletebroker: Could not find path to brokers/$BROKER directory"
  echo "hamqsideletebroker: Please check parameters and directory structure"
  exit 1
else
  rm -f $BROKER               # delete the symlink
  cp -Rp $sharedir $BROKER   # copy the dir from shared to internal
  rm -fr $sharedir           # delete the sharedir
fi

# Handle the registry/$BROKER directory
cd /var/mqsi/registry
sharedir=`ls -e $BROKER | awk '{print $NF}' ` if [ -z $sharedir ] then
  echo "hamqsideletebroker: Could not find path to registry/$BROKER directory"
  echo "hamqsideletebroker: Please check parameters and directory structure"
  exit 1
else
  rm -f $BROKER               # delete the symlink
  cp -Rp $sharedir $BROKER   # copy the dir from shared to internal
  rm -fr $sharedir           # delete the shareddir
fi

cd $curdir

# The broker has been returned to normal directory layout, ready for deletion.
# In order to delete broker, the database must be available. Since the
# AS should have been stopped, it is reasonable to assume that the database
# has been stopped. Restart the database instance, using the DB2 sample script.
# Because we are assuming a clean shutdown of instance took place, the DBNAME
# is not supplied (it is an optional arg to the sample script).
/usr/bin/su - $DBUSER /usr/lpp/db2_06_01/samples/hacmp/hacmp-s1.sh
rc=$?
if [ $rc -ne 0 ] then
  echo "hamqsi_start_broker_as: Could not start the database instance"
  exit $rc
fi
# Delete the broker

```bash
su - $MQUSER -c "mqsideletebroker $BROKER"
if [ $? -ne "0" ]
then
    echo "hamqsideletebroker: Error from mqsideletebroker command"
    exit 1
fi

# Advise user that they can delete Application Monitor
# echo "hamqsideletebroker: You could optionally delete
/MQHA/bin/hamqsi_applmon.$BROKER"
echo "hamqsideletebroker: Completed"

exit 0
```
This script is used to remove the links created using the `hamqsicreatebrokerstandby` script.

```
#!/bin/ksh
# @(#) public/aix/hamqsiremovebrokerstandby, SupportPacs, H000, H000-010403
1.1  01/03/30 15:10:33
#
# (C) Copyright IBM Corporation 2001
#
# IC61: Configuring MQSeries Integrator for AIX with HACMP
#
# PLEASE NOTE - This script is supplied "AS IS" with no warranty or liability. It is not part of any product. Please ensure that you read and understand it before you run it. Make sure that by running it you will not overwrite or delete any important data.
#
# Module: hamqsiremovebrokerstandby
#
# Args:
#   broker = name of broker to delete
#
# Description:
#   This script deletes broker standby information from a node.
#   It detects that the node is a standby before removing it.
#
# Runs as root.

# Check running as root
if [ `id -u` -ne 0 ]
then
  echo "Must be running as root"
  exit 1
fi

# Parse the args for BROKER
BROKER=$1

if [ -z "$BROKER" ]
then
  echo "hamqsiremovebrokerstandby: ERROR! No broker supplied"
  echo "   Usage: hamqsiremovebrokerstandby <BROKER>"
  exit 1
fi
```
# Work out whether the broker is locally present, by trying to cd to
# broker directory. If it is, then report to user that this is not
# a standby node! Output is redirected because we expect this to fail.
curdir="pwd"
cd /var/mqsi/brokers/$BROKER > /dev/null 2>&1
rc=$?
if [ $rc -eq 0 ]
then
  echo "hamqsiremovebrokerstandby:"
  echo \"\tThis node currently has the broker files locally mounted\"
  echo \"\tYou should only run this script on a standby node\"
  exit 1
fi
cd $curdir

cd /var/mqsi/brokers
flo}$BROKER $curdir
for brokerdir in brokri
  do
    echo "Removing symlink for brokers/$BROKER"
    rm -f $BROKER
    # delete the symlink
    echo "Removing symlink for registry/$BROKER"
    rm -f $BROKER
    # delete the symlink
  done

cd $curdir

echo "hamqsiremovebrokerstandby: Completed"

echo "$curdir"

exit 0
hamqsi_qm_datapath

This script is used for finding the location of the MQ queue manager's data files.

#!/bin/ksh
# @(#) public/aix/hamqsi_qm_datapath, SupportPacs, H000, H000-010403
1.1  01/03/30 15:10:40
#
# (C) Copyright IBM Corporation 2001
#
# IC61: Configuring MQSeries Integrator for AIX with HACMP
#
# PLEASE NOTE - This script is supplied "AS IS" with no
# warranty or liability. It is not part of
# any product. Please ensure that you read
# and understand it before you run it. Make
# sure that by running it you will not
# overwrite or delete any important data.
#
# Module:
#    hamqsi_qm_datapath
#
# Parameters:
#    QM  = name of queue manager whose data path is
#          to be found.
#
# Description:
#   Accepts the name of a QM and returns (prints) the fully
#   qualified path of the directory which contains the qmgrs
#   directory for that QM.
#
# This script is used by other hamqsi scripts and is not
# called directly by users or HACMP.

NAME_TO_FIND=$1

# The regular expression in the next line contains a tab character
# Edit only with tab-friendly editors.
cat /var/mqm/mqs.ini | grep -v "^[ ]*#" | \n   awk -F= -v name_to_find=$NAME_TO_FIND \n' BEGIN {
   in_qmstanza="no";
   correct_stanza="no";
   name="not set";
   prefix="not set";
   directory="not set";
}

# body
{ 
    if (in_qmstanza=="no" && $1=="QueueManager:"
    {
        in_qmstanza="yes";
    }
    else if (in_qmstanza=="yes")
    {
      if ($1=="   Name")
      {
        #print "found name "$2
        name=$2;
      }
      if ($1=="   Prefix")
      {
        #print "found prefix "$2
        prefix=$2;
      }
      if ($1=="   Directory")
      {
        #print "found directory "$2
        directory=$2;
      }
      if ( match($1,"[*]正在进行":") ) # start of a stanza
      {
        if (name==name_to_find)
        {
          result=sprintf("%s",prefix);
          #print "result set to "result
        }
        if ($1=="QueueManager:"
        {
          in_qmstanza="yes";
          name="not set";
          prefix="not set";
          directory="not set";
        }
      else
      {
        in_qmstanza="no";
        name="not set";
        prefix="not set";
        directory="not set";
      }
    }
    }  
  }
}
if (name==name_to_find)
{
    result=sprintf("%s",prefix);
    #print "result set to "result
}
}
if (result!="not set")
{
    print result;
}
else
{
    print "not found";
}
,
hamqsi_start_broker

This script is used to start the Message Broker.

#!/bin/ksh
# @(#) public/aix/hamqsi_start_broker, SupportPacs, H000, H000-010403
1.1  01/03/30 15:10:41
#
# (C) Copyright IBM Corporation 2001
#
# IC61: Configuring MQSeries Integrator for AIX with HACMP
#
# PLEASE NOTE - This script is supplied "AS IS" with no
# warranty or liability. It is not part of
# any product. Please ensure that you read
# and understand it before you run it. Make
# sure that by running it you will not
# overwrite or delete any important data.
#
# Module:
#   hamqsi_start_broker
#
# Args:
#   BROKER = name of broker to start
#
# Description:
#   This script attempts to start the MQSI Broker
#
#   Runs as the userid which runs broker, and must have
#   the user's environment (i.e. invoke from "su - $MQUSER ..")
#
BROKER=$1

if [ -z "$BROKER" ]
then
  echo "hamqsi_start_broker: ERROR! No Broker name supplied"
  echo "   Usage: hamqsi_start_broker <BROKER>"
  exit 1
fi

# Ensure that the broker is not already running. In this test
# we look for any broker-related processes, which may have
# been left around after a previous failure. Any that remain
# must now be terminated. This is a brutal means of mopping
# up broker processes.
# We stop the processes in the following order:
#   bipservice       - first so it cannot issue restarts
#   bipbroker        - next for same reason
# DataFlowEngines - last

# echo "hamqsi_start_broker: Ensure $BROKER not already running"
for process in bipservice bipbroker DataFlowEngine
do
  # Output of kill redirected to /dev/null in case no processes
  ps -ef | grep "$process $BROKER" | grep -v grep | \  
      awk '{print $2}' | xargs kill -9 > /dev/null 2>&1
done

# Start the Broker
echo "hamqsi_start_broker: Start Broker " $BROKER
/usr/bin/mqsistart $BROKER > /dev/null 2>&1
if [ $? -ne "0" ]
then
  echo "hamqsi_start_broker: Bad result from mqsistart for $BROKER"
  exit 1
fi

# Check to see if the broker service has started. This loop
# uses a fixed online timeout of approx. 10 seconds.
TIMED_OUT=yes
i=0
while [ $i -lt 10 ]
do
  # Check for Broker start. We look for bipservice and
  # bipbroker to be running; there may be no message flows
  # deployed.
  # Look to see whether bipservice is running
  cnt=`ps -ef | grep "bipservice $BROKER" | grep -v grep | wc -l`
  if [ $cnt -gt 0 ]
  then
    # Look to see whether bipbroker is running
    cnt=`ps -ef | grep "bipbroker $BROKER" | grep -v grep | wc -l`
    if [ $cnt -gt 0 ]
    then
      # Broker is online
      echo "hamqsi_start_broker: ${BROKER} is running"
      TIMED_OUT=no
      break  # out of timing loop
    fi
  fi
  # Manage the loop counter
  i=`expr $i + 1`
  sleep 1
done

# Report error if broker failed to start in time
if [ ${TIMED_OUT} = "yes" ]
then
    echo "hamqsi_start_broker: Broker service failed to start: " $BROKER
    exit 1
else
    # Broker was started and came online within timeout.
    # Launch post-online script in background. We are running
    # as the broker userid and run the script as this userid.
    # This is always done in the background so we don't delay the
    # HA methods, which need to stay responsive. A synchronous
    # call to the user method might delay (indefinitely) the
    # completion of our start_broker method, which would upset
    # other actions being taken by the cluster.
    
    # Is there an executable method for us to run?
    if [ -x /MQHA/bin/rc.mqsi ]
then
    echo "hamqsi_start_broker: run post_online script for ${BROKER}"
    /MQHA/bin/rc.mqsi ${BROKER} post_online &
    # The exit code from online script is deliberately ignored
    fi
fi

exit 0
hamqsi_start_broker_as

This is the main script for starting up all the components required for the Message Broker.

#!/bin/ksh -x
# @(#) public/aix/hamqsi_start_broker_as, SupportPacs, H000, H000-010403
1.1  01/03/30 15:10:42
#
# (C) Copyright IBM Corporation 2001
#
# IC61: Configuring MQSeries Integrator for AIX with HACMP
#
# PLEASE NOTE - This script is supplied "AS IS" with no
# warranty or liability. It is not part of
# any product. Please ensure that you read
# and understand it before you run it. Make
# sure that by running it you will not
# overwrite or delete any important data.
#
#
# Module:
# hamqsi_start_broker_as
#
# Args:
# broker = broker name
# qm     = name of broker queue manager
# mquser = user account under which QM and Broker are run
#
# Description:
# Starting an MQSI Broker requires the following services:
# (1) The MQSeries Queue Manager which supports the Broker
# (2) The MQSI Broker service
# This script provides a single source to initiate the required
# services in sequence.
#
# Queue Manager:
# This script uses the hamqm_start_su script supplied by MQSeries
# SupportPac MC63: MQSeries for AIX - Implementing with HACMP,
# to start the MQSeries Queue Manager. The hamqm_start_su script
# ensures that the QM is fully stopped before starting it.
#
# Broker:
# This script then invokes the hamqsi_start_broker script which
# checks that the broker is fully stopped and then starts it.
#
# The hamqsi_start_broker_as script should be run as root.
# Check running as root
if [ `id -u` -ne 0 ]
then
  echo "Must be running as root"
  exit 1
fi

BROKER=$1
QM=$2
MQUSER=$3
PORT=$4

# Check all parameters exist
if [ -z "$BROKER" ]
then
  echo "hamqsi_start_broker_as: ERROR! No Broker name supplied"
  echo "   Usage: hamqsi_start_broker_as <BROKER> <QM> <MQUSER> <PORT>"
  exit 1
fi

if [ -z "$QM" ]
then
  echo "hamqsi_start_broker_as: ERROR! No queue manager name supplied"
  echo "   Usage: hamqsi_start_broker_as <BROKER> <QM> <MQUSER> <PORT>"
  exit 1
fi

if [ -z "$MQUSER" ]
then
  echo "hamqsi_start_broker_as: ERROR! No Userid supplied"
  echo "   Usage: hamqsi_start_broker_as <BROKER> <QM> <MQUSER> <PORT>"
  exit 1
fi

if [ -z "$PORT" ]
then
  echo "hamqsi_start_broker_as: ERROR! No Listener Port supplied"
  echo "   Usage: hamqsi_start_broker_as <BROKER> <QM> <MQUSER> <PORT>"
  exit 1
fi

#-----------------------------------------
# If Listeners have stopped restart
#

cnt=`ps -ef | grep runmqslsr | grep ${QM} | grep -v grep | wc -l`
if [ $cnt -eq 0 ]
then

# Start the Listeners
#
# Ensure that the Broker is not already running and start the Broker
su - $MQUSER -c "/usr/mqm/hascripts/hamqm_lsnr_start_su $QM $PORT &"
rc=$?
if [ $rc -ne 0 ]
then
    echo "hamqsi_start_broker_as: Could not start the QM Listener"
    exit $rc
fi
else
    echo "hamqsi_start_broker_as: QM Listener already running"
fi

#--------------------------------------------------
# If queue manager was stopped then bring down broker
# before restarting
#
su - $MQUSER -c "echo "ping qmgr" | runmqsc $(QM)" > /dev/null 2>&1
pingresult=$?
# pingresult will be 0 on success; non-zero on error (man runmqsc)
brkcnt=`ps -ef | grep "bipservice $BROKER" | grep -v grep | wc -l`
# If it was just listener stopped then only restart listener
if [ $pingresult -eq 0 ]  &&  [ $brkcnt -ne 0 ]
then
    echo "hamqsi_start_broker_as: QM and Broker already running so not
    restarting"
    exit 0
fi

if [ $pingresult -ne 0 ]
then

#--------------------------------------------------
# If the broker is running stop it
#
cnt=`ps -ef | grep "bipservice $BROKER" | grep -v grep | wc -l`
if [ $cnt -ne 0 ]
then
    echo "hamqsi_start_broker_as: Stopping Broker "$BROKER" as QM is down"
su $MQUSER -c "/usr/mqm/hascripts/hamqsi_stop_broker $BROKER 5"
rc=$?
    if [ $rc -ne 0 ]
then
echo "hamqsi_start_broker_as: Could not stop Broker before restarting"
exit $rc
fi
fi

# Start the Queue Manager by using the MC63 start script
#
echo "hamqsi_start_broker_as: Start Queue manager "$QM
su $MQUSER -c "/usr/mqm/hascripts/hamqm_start_su $QM"
rc=$?
if [ $rc -ne 0 ]
then
echo "hamqsi_start_broker_as: Could not start the queue manager"
exit $rc
fi

# Start the Queue Manager command server
#
echo "hamqsi_start_broker_as: Start Queue manager Command Server "$QM
su $MQUSER -c "/usr/bin/strmqcsv $QM"
rc=$?
if [ $rc -ne 0 ]
then
echo "hamqsi_start_broker_as: Could not start the queue manager command server"
exit $rc
fi
else
  echo "hamqsi_start_broker_as: Queue Manager already running"
  echo "hamqsi_start_broker_as: Queue Manager will NOT be restarted"
fi

# Start the Broker
#
# Ensure that the Broker is not already running and start the Broker
su - $MQUSER -c "/usr/mqm/hascripts/hamqsi_start_broker $BROKER"
rc=$?
if [ $rc -ne 0 ]
then
echo "hamqsi_start_broker_as: Could not start the broker"
exit $rc
fi

exit $rc
hamqsi_stop_broker

This script is used to stop the Message Broker.

#!/bin/ksh
# @(#) public/aix/hamqsi_stop_broker, SupportPacs, H000, H000-010403
1.1  01/03/30 15:10:46
#
# (C) Copyright IBM Corporation 2001
#
# IC61: Configuring MQSeries Integrator for AIX with HACMP
#
# PLEASE NOTE - This script is supplied "AS IS" with no
# warranty or liability. It is not part of
# any product. Please ensure that you read
# and understand it before you run it. Make
# sure that by running it you will not
# overwrite or delete any important data.
#
#
# Module:
#   hamqsi_stop_broker
#
# Args:
#   broker = name of broker
#   timeout = max time to allow for each phase of termination
#
# Description:
#   This script stops the broker, forcibly if necessary.
#   The script should be run by the user account under which
#   the broker is run, including environment.

BROKER=$1
TIMEOUT=$2

if [ -z "$BROKER" ]
then
    echo "hamqsi_stop_broker: ERROR! No broker name supplied"
    echo "   Usage: hamqsi_stop_broker <BROKER> <TIMEOUT>"
    exit 1
fi

if [ -z "$TIMEOUT" ]
then
    echo "hamqsi_stop_broker: ERROR! No timeout supplied"
    echo "   Usage: hamqsi_stop_broker <BROKER> <TIMEOUT>"
    exit 1
fi
# Invoke Pre-Offline user exit, if one exists
if [ -x /MQHA/bin/rc.mqsi ]
then
    echo "hamqsi_stop_broker: Run pre-offline script for ${BROKER}"
    /MQHA/bin/rc.mqsi ${BROKER} pre_offline &
    # Exit code from pre_offline script is deliberately ignored
fi

for severity in normal immediate terminate
do
    # Issue the stop method in the background - we don't
    # want to risk having it hang us up, indefinitely. We
    # want to be able to concurrently run a TIMEOUT timer
    # to give up on the attempt, and try a more forceful
    # stop. If the kill version fails then there is nothing
    # more we can do here anyway.

    echo "hamqsi_stop_broker: Attempting ${severity} stop of ${BROKER}"
    case $severity in
        normal)
            # Minimum severity of stop is to issue mqsistop
            mqsistop $BROKER > /dev/null 2>&1 &
            ;;
        immediate)
            # This is an immediate stop.
            mqsistop $BROKER -i > /dev/null 2>&1 &
            ;;
        terminate)
            # This is a brutal means of mopping up Broker processes.
            # We stop the processes in the following order:
            #   bipservice       - first so it cannot issue restarts
            #   bipbroker        - next for same reason
            #   DataFlowEngines  - last
            for process in bipservice bipbroker DataFlowEngine
do
                # Output of kill redirected to /dev/null in case no processes
                ps -ef | grep "$process ${BROKER}" | grep -v grep | \
                    awk '{print $2}'| xargs kill -9 > /dev/null 2>&1
done
            ;;
        esac

    echo "hamqsi_stop_broker: Waiting for ${severity} stop of ${BROKER} to complete"
TIMED_OUT=yes
SECONDS=0
while (( $SECONDS < ${TIMEOUT} ))
do
    # See whether there are any broker processes still running
    cnt="ps -ef | \ 
        grep -E "bipservice $BROKER|bipbroker $BROKER|DataFlowEngine $BROKER" | \ 
        grep -v grep | wc -l"
    if [ $cnt -gt 0 ]
    then
        # It's still running...wait for timeout
        sleep 1 # loop granularity
    else
        # It's stopped, as desired
        echo "${BROKER} has stopped"
        TIMED_OUT=no
        break # out of while ..offline timeout loop
    fi
done # timeout loop

if [ ${TIMED_OUT} = "yes" ]
then
    continue # to next level of urgency
else
    break # instance is stopped, job is done
fi

done # next level of urgency

if [ ${TIMED_OUT} = "no" ]
then
    echo "hamqsi_stop_broker: Completed"
    exit 0
else
    echo "hamqsi_stop_broker: Completed with errors"
    exit 1
fi
hamqsi_stop_broker_as

This script is used to stop all the components required for the Message Broker.

#!/bin/ksh -x
# @(#) public/aix/hamqsi_stop_broker_as, SupportPacs, H000, H000-010403
1.1 01/03/30 15:10:48
#
# (C) Copyright IBM Corporation 2001
#
# IC61: Configuring MQSeries Integrator for AIX with HACMP
#
# PLEASE NOTE - This script is supplied "AS IS" with no
# warranty or liability. It is not part of
# any product. Please ensure that you read
# and understand it before you run it. Make
# sure that by running it you will not
# overwrite or delete any important data.
#
#
# Module:
#   hamqsi_stop_broker_as
#
# Arguments are:
#   broker = name of broker
#   qm     = name of broker queue manager
#   mquser = user account under which QM and broker run
#   timeout = max time to allow each phase of stop processing
#
# Description:
#   This script stops the Broker, Queue Manager and Database Instance
# in that sequence.
#
# Broker:
#   The script invokes the hamqsi_stop_broker script to stop the
#   broker, which checks that the broker is fully stopped.
#
# Queue Manager:
#   This script uses the hamqm_stop_su script supplied by MQSeries
# SupportPac MC63: MQSeries for AIX - Implementing with HACMP,
# to manage the MQSeries Queue Manager termination.
# The hamqm_stop_su script ensures that the QM is fully stopped.
#
# The hamqsi_stop_broker_as script should be run as root.

# Check running as root
if [ `id -u` -ne 0 ]
then
```

Appendix A. HA scripts

179

echo "Must be running as root"
exit 1

BROKER=$1
QM=$2
MQUSER=$3
TIMEOUT=$4

# Check all parameters

if [ -z "$BROKER" ]
then
    echo "hamqsi_stop_broker_as: ERROR! No Broker name supplied"
    echo "   Usage: hamqsi_stop_broker_as <BROKER> <QM> <MQUSER> <TIMEOUT>"
    exit 1
fi

if [ -z "$QM" ]
then
    echo "hamqsi_stop_broker_as: ERROR! No queue manager name supplied"
    echo "   Usage: hamqsi_stop_broker_as <BROKER> <QM> <MQUSER> <TIMEOUT>"
    exit 1
fi

if [ -z "$MQUSER" ]
then
    echo "hamqsi_stop_broker_as: ERROR! No userid supplied"
    echo "   Usage: hamqsi_stop_broker_as <BROKER> <QM> <MQUSER> <TIMEOUT>"
    exit 1
fi

if [ -z "$TIMEOUT" ]
then
    echo "hamqsi_stop_broker_as: ERROR! No Timeout value supplied"
    echo "   Usage: hamqsi_stop_broker_as <BROKER> <QM> <MQUSER> <TIMEOUT>"
    exit 1
fi

METHOD_STATUS="OK"

# Stop the BROKER

 METHOD_STATUS="OK"

# Stop the BROKER

# "hamqsi_stop_broker_as: Stop Broker " $BROKER
su - $MQUSER -c "'/usr/mqm/hascripts/hamqsi_stop_broker $BROKER $TIMEOUT"
if [ $? -ne "0"
then
```

Appendix A. HA scripts  179
# Even if the above operation failed, just report and then continue by
# stopping other components
echo "hamqsi_stop_broker_as: Attempt to stop broker $BROKER failed"
METHOD_STATUS="Error"
fi

# Stop the Queue Manager, using script from MC63
# echo "hamqsi_stop_broker_as: Stop Queue Manager $QM"
su $MQUSER -c "/usr/mqm/hascripts/hamqm_stop_su $QM $TIMEOUT"
if [ $? -ne "0" ]
then
    # Even if the above operation failed, just report and then continue by
    # stopping other components
    echo "hamqsi_stop_broker_as: Attempt to stop queue manager $QM failed"
    METHOD_STATUS="Error"
    fi

# Stop the Listener
# echo "hamqsi_stop_broker_as: Stop Listeners on Queue Manager "$QM
su - $MQUSER -c "/usr/mqm/hascripts/hamqm_lsnr_stop_su $QM"
if [ $? -ne "0" ]
then
    # Even if the above operation failed, just report and then continue by
    # stopping other components
    echo "hamqsi_stop_broker_as: Attempt to stop Listeners on QM $QM failed"
    METHOD_STATUS="Error"
    fi

if [ ${METHOD_STATUS} = "OK" ]
then
    exit 0
else
    echo "hamqsi_stop_broker_as: Completed with errors"
    exit 1
fi
hamqm_lsnr_start

This script is used to run the hamqm_lsnr_start_su as the mqm user.

#!/bin/ksh
# @(#) public/aix/hamqm_lsnr_start
#
# (C) Copyright IBM Corporation 2000
#
# DESCRIPTION:
#   Because HACMP runs scripts with root authority, and MQSeries prefers
#   to be managed by the mqm userid, we've got here a simple wrapper
#   which will switch to the correct account before running the real
#   command.
#
su mqm -c /usr/mqm/hascripts/hamqm_lsnr_start_su $*
This script is used to start an MQ Listener.

```
#!/bin/ksh
# @(#) public/aix/hamqm_lsnr_start_su
#
# DESCRIPTION:
# /MQHA/bin/hamqm_lsnr_start_su <qmgr name> <port number>
#
# This script should start the qmgr listener
#
# The script should be run by the mqm user. A wrapper script is also
# provided to switch userids from HACMP's root authority.

# Argument queue manager name
QM=$1
# Argument port number
PORT=$2

if [ -z "$QM" ]
then
    echo "hamqm_lsnr_start_su: ERROR! No queue manager name supplied"
    exit 1
fi

if [ -z "$PORT" ]
then
    echo "hamqm_lsnr_start_su: ERROR! No listener port supplied"
    exit 1
fi

# Check the listener is not running and

cnt=`ps -ef | grep runmqlsr | grep $QM | grep $PORT | wc -l`
if [ $cnt -ne 0 ]
then
    echo "Listener is already running on $QM port: $PORT"
    exit 1
fi

# Should now be safe to start the QM
runmqlsr -m $QM -t TCP -p $PORT &
rc=$?

# Fire post-onlins script if good return from strmqm.
```
if [ $rc -eq 0 ]
then
  echo "Listener started on ${QM} port: ${PORT}"
fi

exit $rc
hamqm_lsnr_stop

This script is used to run the hamqm_lsnr_stop_su as the mqm user.

#!/bin/ksh  
# @(#) public/aix/hamqm_lsnr_statop  
#  
# (C) Copyright IBM Corporation 2000  
#  
# DESCRIPTION:  
#   Because HACMP runs scripts with root authority, and MQSeries prefers  
#   to be managed by the mqm userid, we've got here a simple wrapper  
#   which will switch to the correct account before running the real  
#   command.

su mqm -c /usr/mqm/hascripts/hamqm_lsnr_stop_su $*
hamqm_lsnr_stop_su

This script is used to stop a MQ Listener.

#!/bin/ksh
# @(#) public/aix/hamqm_lsnr_stop_su
#
# DESCRIPTION:
# /MQHA/bin/hamqm_lsnr_stop_su <qmname> <portnumber>
#
# Stops the QM listener on a given port
#
# The script should be run by the mqm user. A wrapper script is also
# provided to switch userids from HACMP's root authority.

QM=$1

if [ -z "$QM" ]
then
    echo "hamqm_stop: ERROR! No queue manager name supplied"
    exit 1
fi

# Check to see if the QM is already stopped. If so, just make
# sure no processes are lying around.
#
cnt=`ps -ef | grep runmqlsr | grep $QM | grep -v grep | wc -l`
if [ $cnt -eq 0 ]
then
    echo "No Listener is not running for QM $QM" 
    exit 0
fi

dmqlsr -m $QM

echo "Listener Stopped"

exit 0
hamqsi_start_broker_as.MBBRK

This is the main HACMP start script.

#!/bin/ksh
/usr/mqm/hascripts/hamqsi_start_broker_as MBBRK MB_QM superman 1414

hamqsi_stop_broker_as.MBBRK

This is the main HACMP stop script.

#!/bin/ksh
/usr/mqm/hascripts/hamqsi_stop_broker_as MBBRK MB_QM superman 15
echo "Sleeping for 10 seconds"
sleep 10

hamqsi_start_broker_as.MB2BRK

This is the main HACMP start script for the second Message Broker.

#!/bin/ksh
/usr/mqm/hascripts/hamqsi_start_broker_as MB2BRK MB_QM superman 2020

hamqsi_stop_broker_as.MBBRK

This is the main HACMP stop script for the second Message Broker.

#!/bin/ksh
/usr/mqm/hascripts/hamqsi_stop_broker_as MB2BRK MB_QM superman 15
echo "Sleeping for 10 seconds"
sleep 10
hamqsi_monitor_broker_as

This is the script containing the generic command for monitoring all the components needed by the broker.

#!/bin/ksh
# @(#) public/aix/hamqsi_monitor_broker_as, SupportPacs, H000, H000-010403
1.1  01/03/30 15:10:36
#
# (C) Copyright IBM Corporation 2001
#
# IC61: Configuring MQSeries Integrator for AIX with HACMP
#
# PLEASE NOTE - This script is supplied "AS IS" with no warranty or liability. It is not part of any product. Please ensure that you read and understand it before you run it. Make sure that by running it you will not overwrite or delete any important data.
#
# Module:
#   hamqsi_monitor_broker_as
#
# Args:
#   BROKER = name of broker in AppServer
#   QM     = name of queue manager in AppServer
#   MQUSER = userid under which queue manager and broker run
#   PORT   = port number for the listener
#
# Description:
#   This is the application monitor script used with HACMP/ES. It needs to be invoked by a parameter-less wrapper script because HACMP does not allow parameters to be passed to application monitor scripts.
#
# This hamqsi_monitor_broker_as script is run as root, and uses su as needed to monitor the 3 components of the application server.
#
# This script is tolerant of a queue manager that is still in startup or a database instance that is still in startup. If either the queue manager or database is still starting this application monitor script will exit with 0 - which indicates to HACMP that there's nothing wrong. This is to allow for startup time for the queue manager and database instance which may exceed the Stabilisation Interval set for the Application Monitor in HACMP/ES.
#
#
# Exit codes:
# 0 => Broker, QM are all running OK or starting
# >0 => One or more components are not responding.
#

# Check running as root
if [ `id -u` -ne 0 ]
then
   echo "Must be running as root"
   exit 1
fi

BROKER=$1
QM=$2
MQUSER=$3
PORT=$4

# Check the parameters
if [ -z "$BROKER" ]
then
   echo "hamqsi_monitor_broker_as: ERROR! No broker name supplied"
   exit 1
fi

if [ -z "$QM" ]
then
   echo "hamqsi_monitor_broker_as: ERROR! No queue manager name supplied"
   exit 1
fi

if [ -z "$MQUSER" ]
then
   echo "hamqsi_monitor_broker_as: ERROR! No mquser supplied"
   exit 1
fi

if [ -z "$PORT" ]
then
   echo "hamqsi_monitor_broker_as: ERROR! No Listener port supplied"
   exit 1
fi

# Use a state variable to reflect the state of components as they
# are tested. Valid values are "stopped", "starting" and "started"
# Initialise it to "stopped" for safety.
STATE="stopped"
# Check that the queue manager is running or starting.
# We can't use the MC63 applmon for this as it has to return a
# boolean and we need to be able to differentiate between
# starting and started so that we know whether to go on and
# test database and broker.

su - $MQUSER -c "echo "ping qmgr" | runmqsc ${QM}" > /dev/null 2>&1
pingresult=$?  
# pingresult will be 0 on success; non-zero on error (man runmqsc)
if [ $pingresult -eq 0 ]
then
    # ping succeeded
    echo "hamqsi_monitor_broker_as: Queue Manager ${QM} is responsive"
    STATE="started"
else
    # ping failed
    # Don't condemn the QM immediately, it may be in startup.
    # The following regexp includes a space and a tab, so use tab-friendly
    # editors.
    srchstr=" $QM[ ]*"$
    cnt=`ps -ef | grep strmqm | grep "$srchstr" | grep -v grep 
    | awk '{print $2}' | wc -l`
    if [ $cnt -gt 0 ]
    then
        # It appears that QM is still starting up, tolerate
        echo "hamqsi_monitor_broker_as: Queue Manager ${QM} is starting"
        STATE="starting"
    else
        # There is no sign of QM start process
        echo "hamqsi_monitor_broker_as: Queue Manager ${QM} is not responsive"
        STATE="stopped"
    fi
fi

# Decide whether to continue or to exit
case $STATE in
    stopped)
        echo "hamqsi_monitor_broker_as: Queue manager ($QM) is not running
correctly"
        exit 1
        ;;
    starting)
        echo "hamqsi_monitor_broker_as: Queue manager ($QM) is starting"
        echo "hamqsi_monitor_broker_as: WARNING - Stabilisation Interval may be too shorten"
echo "hamqsi_monitor_broker_as: WARNING - No test of broker $BROKER will be conducted"
    echo "hamqsi_monitor_broker_as: WARNING - No test of Listeners will be conducted"
    exit 0
;;
started)
    echo "hamqsi_monitor_broker_as: Queue manager ($QM) is running"
    continue # proceed by testing database instance
    ;;
esac

# ---ująuszng (1u (with (a) (b) (c) (d)) (with (a) (b) (c) (d)) (with (a) (b) (c) (d)))
# Check the MQSI Broker is running
#
# Re-initialise STATE for safety
STATE="stopped"
#
# The broker runs as a process called bipservice which is responsible for starting and re-starting the admin agent process (bipbroker).
# The bipbroker is responsible for starting any DataFlowEngines. If no execution groups have been assigned to the broker there will be no DataFlowEngine processes. There should always be a bipservice and bipbroker process pair. This monitor script only tests for bipservice, because bipservice should restart bipbroker if necessary - the monitor script should not attempt to restart bipbroker and it may be premature to report an absence of a bipbroker as a failure.
#
cnt=`ps -ef | grep "bipservice $BROKER" | grep -v grep | wc -l`
if [ $cnt -eq 0 ]
then
    echo "hamqsi_monitor_broker_as: MQSI Broker $BROKER is not running"
    STATE="stopped"
else
    echo "hamqsi_monitor_broker_as: MQSI Broker $BROKER is running"
    STATE="started"
fi

# Decide how to exit
case $STATE in
    stopped)
        echo "hamqsi_monitor_broker_as: Broker ($BROKER) is not running correctly"
        exit 1
        ;;
        started)
        echo "hamqsi_monitor_broker_as: Broker ($BROKER) is running"
        continue
        ;;
esac

# Check the MQ Listener is running
#
# Re-initialise STATE for safety
STATE="stopped"

cnt=`ps -ef | grep runmqlsr | grep $QM | grep -v grep | grep $PORT | wc -l`
if [ $cnt -eq 0 ]
 then
  echo "hamqsi_monitor_broker_as: MQ Listener on port $PORT is not running"
  STATE="stopped"
else
  echo "hamqsi_monitor_broker_as: MQ Listener on port $PORT is running"
  STATE="started"
fi

# Decide how to exit
case $STATE in
  stopped)
    echo "hamqsi_monitor_broker_as: Listener $QM - $PORT is not running correctly"
    exit 1
  ;;
  started)
    echo "hamqsi_monitor_broker_as: Listener $QM - $PORT is running"
    exit 0
  ;;
esac
hamqsi_applmon.MBBRK

This is the script for monitoring the Message Broker.

```
#!/bin/ksh
/usr/mqm/hascripts/hamqsi_monitor_broker_as MBBRK MB_QM superman 1414
```

hamqsi_applmon.MB2BRK

This is the script for monitoring the second Message Broker.

```
#!/bin/ksh
/usr/mqm/hascripts/hamqsi_monitor_broker_as MB2BRK MB2_QM superman 2020
```
Message Broker adapter scripts

This section lists the start, stop, and monitor scripts for the JText adapter. These scripts can be applied to all distributed adapters. To use these scripts you must edit the set_remote_adapter_variables file to match your environment, and also change the monitor, start, and stop scripts to call the variable file.

remote_adapter_HA_monitor

This script is used by HACMP to monitor the status of the adapter and its underlying queue manager.

```bash
#!/usr/bin/sh
# WBI Remote Adapters Monitor Script for HACMP 5.1
#set -x

#set Variables
#change this to your root directory
./opt/IBM/MBAdapters/bin/hascripts/set_remote_adapter_variables

# Monitor Adapter QM
su - mqm -c "echo "ping qmgr" | runmqsc ${QM_NAME}" > /dev/null 2>&1
pingresult=$?
# pingresult will be 0 on success; non-zero on error (man runmqsc)
if [ $pingresult -ne 0 ]
then
  echo "Adapter QM is unavaible"
  exit 1
fi

#Check MQ Listener and Command Server running
cnt=`ps -ef | grep runmqtsr | grep -v grep | grep $QM_NAME | grep $PORT | wc -l`
if [ $cnt -eq 0 ]
then
  echo "QM Listener is unavaible"
  exit 1
fi

cnt=`ps -ef | grep amqpcsea | grep -v grep | grep $QM_NAME | wc -l`
if [ $cnt -eq 0 ]
then
  su - mqm -c strmqcsv $QM_NAME
  resultvar=$?
  if [ ${resultvar} -ne 0 ]
  then
    echo "$PROGID - $HOST - MQ Command Server could not start. `date`"
    echo "$PROGID - $HOST - please check MQ logs. `date`"
  fi
fi
```
fi
fi

# Monitor Adapter process
$SCRIPT_LOCATION/remote_adapter_restart $ADAPTER_ROOT $ID $PROGID $HOST
resultvar=$?
if [ ${resultvar} -ne 0 ]
then
    echo "Not able to restart Remote Adapters"
    echo "Adapter failover does not trigger failover"
fi
exit 0
remote_adapter_ha_start

This script starts all the components of the JText adapter. It first checks if the components are available, and if not then attempts to start them. This script is used both as a start and restart.

#!/usr/bin/sh
# Remote Adapter Start Script for HACMP 5.1
#set -x

#set Variables
. /opt/IBM/MBAdapters/bin/hascripts/set_remote_adapter_variables

# Start MQSeries
su - mqm -c "echo "ping qmgr" | runmqsc ${QM_NAME}" > /dev/null 2>&1
pingresult=$?
# pingresult will be 0 on success; non-zero on error (man runmqsc)
if [ ${pingresult} != "0" ]
then
    echo "$PROGID - $HOST: Starting MQSeries. `date`"
    su mqm -c $SCRIPT_LOCATION/MQ/hamqm_start_su $QM_NAME $SCRIPT_LOCATION
    resultvar=$?
    if [ ${resultvar} -ne 0 ]
    then
        echo "$PROGID - $HOST - Return code $resultvar: Did not start MQ.
        `date`"
        exit 1
    fi
else
    echo "$PROGID - $HOST - MQ Queue Manager already started. `date`"
fi

# Start MQ Listener and Command Server if not already started
# if listener isn't running exit with return code

# echo "$PROGID - $HOST: Checking MQ Listener and Command Server. `date`"
cnt=`ps -ef | grep runmqlsr | grep -v grep | grep $QM_NAME | grep $PORT | wc -l`
if [ $cnt -eq 0 ]
then
    echo "$PROGID - $HOST: Starting MQSeries Listener. `date`"
    su - mqm -c runmqlsr -m $QM_NAME -t TCP -p $PORT &
    resultvar=$?
    if [ ${resultvar} -ne 0 ]
    then
        echo "$PROGID - $HOST - Return code $resultvar: Could not start MQ
        Listener. `date`"
        #Remote Adapter not-starting does not cause a failover
        exit ${resultvar}
# Start Adapters

$SCRIPT_LOCATION/remote_adapter_restart $ADAPTER_ROOT $ID $PROGID $HOST
resultvar=$?
if [ ${resultvar} -ne 0 ]
then
  echo "$PROGID - $HOST - Return code $resultvar: Could not start Remote Adapters. `date`"
  #Remote Adapter not-starting does not cause a failover
fi
exit 0

### remote_adapter_ha_stop

This script stops all components of the JText adapter.

#!/usr/bin/sh
# Remote Adapter Stop Script for HACMP 5.1

#set Variables
. /opt/IBM/MBAdapters/bin/hascripts/set_remote_adapter_variables

# Stop Connectors
echo "$PROGID - $HOST: Stopping Connector JTXT. `date`"
su - $ID -c "$ADAPTER_ROOT/bin/connector_manager_JText -stop"
notStopped=0
resultvar=$?
if [ ${resultvar} -ne 0 ]
then
  echo "$PROGID - $HOST - Return code $resultvar: Did not stop Connector JTXT. `date`"
  notStopped=$resultvar

# Stop MQSeries

```bash
fi

# Stop MQSeries
echo "$PROGID - $HOST: Stopping MQSeries. `date`"
su mqm -c $SCRIPT_LOCATION/MQ/hamqm_stop_su $QM_NAME 10 $SCRIPT_LOCATION
resultvar=$?
if [ ${resultvar} -ne 0 ]
then
  echo "$PROGID - $HOST - Return code $resultvar: Did not stop MQSeries. `date`"
  notStopped=$resultvar
fi

# Stop MQ Listener
echo "$PROGID - $HOST - Stopping Listener for Adapter QM. `date`"
su mqm -c endmqlsr -m $QM_NAME
resultvar=$?
if [ ${resultvar} -ne 0 ]
then
  echo "$PROGID - $HOST - Return code $resultvar: Could not stop MQSeries Listener. `date`"
  notStopped=$resultvar
fi

exit $notStopped
```
remote_adapter_restart

This script is called by the monitor and start script to start the adapter if needed.

# you would ps - grepping for the adapter process since
# connector_manager_JText -IsConnectorAgentAlive does not work with
# non-ICS brokers
cnt=`ps -ef | grep JTextConnector | grep -v grep | wc -l`
if [ $cnt -eq 0 ]
then
    echo "$PROGID - $HOST - Starting JText Connector. `date`"
    su - $ID -c $ADAPTER_ROOT/bin/connector_manager_JText -start
else
    echo "$PROGID - $HOST - JTextConnector already started. `date`"
fi

set_remote_adapter_variables

These are the variables which set up each time one of the above scripts is called.

#set Variables
# Initialize variables
APP_WAIT=30
HOST=`/bin/hostname -s`
PROGID=`echo $0 | sed 's%/usr/bin/%%g'`

# Root directory of the adapters
ADAPTER_ROOT=/opt/IBM/MBAdapters
SCRIPT_LOCATION=$ADAPTER_ROOT/bin/hascripts

#MQ Variables
QM_NAME=ADAPTER_QM
PORT=1515

#User ID for remote adapters
ID=superman
InterChange Server scripts

This section lists all the scripts that are required to run WebSphere Interchange Server under HACMP control. These tests have been developed for HACMP clustering software on the AIX operating system. These scripts start, stop, and monitor all the applications that are part of the WICS deployment. These applications are WebSphere Interchange Server V4.2.2, MQ V5.3, IBM ORB, and WICS adapters.

For MQ start, stop, and monitoring of these scripts, refer the MQ HA scripts that are part of MQ support pac.

wics_ha_start

#!/usr/bin/sh
# WICS Start Script for HACMP 5.1

# Initialize variables
APP_WAIT=60
ADAPTER_WAIT=30
MQ_WAIT=15
HOST=/bin/hostname -s`
PROGID=`echo $0 | sed 's%/usr/bin/%%g'`
CROSSWORLDS=/opt/IBM/WebSphereICS
export CROSSWORLDS
QM_NAME=ics422.queue.manager
PORT=1415

#Start the IBM ORB's name service
echo "$PROGID - $HOST: Starting IBM ORB's name service. `date`"
su - cwadmin '-c $CROSSWORLDS/bin/hascripts/wics_orb_start'
resultvar=$?
echo "out of ORB script $resultvar"
if [ ${resultvar} -ne 0 ]
then
  echo "$PROGID - $HOST - Return code $resultvar: IBM ORB's name service is not running. `date`"
  exit $resultvar
fi

# Start MQSeries
su - cwadmin -c "echo "ping qmgr" | runmqsc ${QM_NAME}" > /dev/null 2>&1
pingresult=$?
# pingresult will be 0 on success; non-zero on error (man runmqsc)
if [ $(pingresult) != "0"
then
  echo "$PROGID - $HOST: Starting MQSeries. `date`"
su - cwadmin -c $CROSSWORLDS/bin/hascripts/mq/hamqm_start_su $QM_NAME
$CROSSWORLDS/bin/hascripts/mq
resultvar=?
    if [ $(resultvar) -ne 0 ]
    then
        echo "$PROGID - $HOST - Return code $resultvar: Did not start MQ.
            `date`"
        exit 1
    fi
else
    echo "$PROGID - $HOST - MQ Queue Manager already started. `date`"
fi

# Start MQ Listener and Command Server if not already started
    # if listener isn't running exit with return code
    cnt=`ps -ef | grep runmqlsr | grep -v grep | grep $QM_NAME | grep $PORT | wc -l`
    if [ $cnt -eq 0 ]
    then
        echo "$PROGID - $HOST: Starting MQSeries Listener. `date`"
        su - cwadmin -c runmqlsr -m $QM_NAME -t TCP -p $PORT &
        resultvar=?
        if [ $(resultvar) -ne 0 ]
        then
            echo "$PROGID - $HOST - Return code $resultvar: Could not start MQ
            Listener. `date`"
            exit $resultvar
        else
            echo "$PROGID - $HOST - Return code $cnt: MQ Listener started. `date`"
            fi
    fi
    # if the command server isn't running do not fail over
    cnt=`ps -ef | grep amqpcsea | grep -v grep | grep $QM_NAME | wc -l`
    if [ $cnt -eq 0 ]
    then
        echo "$PROGID - $HOST - Starting MQ Command Server. `date`"
        su - cwadmin -c strmqcsv $QM_NAME
        else
            echo "$PROGID - $HOST - MQ Command Server already started. `date`"
        fi
    sleep $MQ_WAIT
    # Test for Server being active
    su - cwadmin '-c $CROSSWORLDS/bin/ics_manager -IsServerAlive'
    resultvar=?
    if [ $(resultvar) -ne 0 ]
    then
# Start CrossWorlds ICS
echo "$PROGID - $HOST: Starting CrossWorlds ICS. `date`"
su - cwadmin '-c $CROSSWORLDS/bin/ics_manager -start
-c$CROSSWORLDS/config/InterchangeSystem.cfg'

###############
# This value must change based on the length of time repository loads
echo "$PROGID - $HOST: Must wait for repository to load. Sleeping $APP_WAIT
seconds"
sleep $APP_WAIT
fi

# Test for Server being active
su - cwadmin '-c $CROSSWORLDS/bin/ics_manager -IsServerAlive'
resultvar=?
if [ ${resultvar} -ne 0 ]
then
echo "$PROGID - $HOST - Return code $resultvar: Interchange Server process is
not running. `date`"
exit $resultvar
fi

# Test for ORDJDBC Connector being active
echo "$PROGID - $HOST: Testing CrossWorlds ORDJDBC Connector Alive. `date`"
su - cwadmin '-c $CROSSWORLDS/bin/connector_manager_ORDJDBC
-IsConnectorAgentAlive'
resultvar=?
if [ ${resultvar} -ne 0 ]
then
    # Start the ORDJDBC Connector
    echo "$PROGID - $HOST: Starting CrossWorlds ORDJDBC Connector. `date`"
su - cwadmin '-c $CROSSWORLDS/bin/connector_manager_ORDJDBC -start
-c$CROSSWORLDS/config/InterchangeSystem.cfg'
sleep $ADAPTER_WAIT
fi

# Test for Connector being active
su - cwadmin '-c $CROSSWORLDS/bin/connector_manager_ORDJDBC
-IsConnectorAgentAlive'
resultvar=?
if [ ${resultvar} -ne 0 ]
then
    echo "$PROGID - $HOST - Return code $resultvar: Not connected to ORDJDBC
Connector. `date`"
fi

# Test for WBIMB Connector being active
echo "$PROGID - $HOST: Testing CrossWorlds WBIMB Connector Alive. `date`"
su - cwadmin '-c $CROSSWORLDS/bin/connector_manager_WBIMB
-IsConnectorAgentAlive'
resultvar=$?
if [ ${resultvar} -ne 0 ]
then
    # Start the WMQI Connector
    echo "$PROGID - $HOST: Starting CrossWorlds WBIMB Connector. `date`"
    su - cwadmin '-c $CROSSWORLDS/bin/connector_manager_WBIMB -start
    -c$CROSSWORLDS/config/InterchangeSystem.cfg'
    sleep $ADAPTER_WAIT
fi

# Test for Connector being active
su - cwadmin '-c $CROSSWORLDS/bin/connector_manager_WBIMB
-IsConnectorAgentAlive'
resultvar=$?
if [ ${resultvar} -ne 0 ]
then
    echo "$PROGID - $HOST - Return code $resultvar: Not connected to WBIMB
    Connector. `date`"
fi

# Test for REQJDBC Connector being active
echo "$PROGID - $HOST: Testing CrossWorlds REQJDBC Connector Alive. `date`"
su - cwadmin '-c $CROSSWORLDS/bin/connector_manager_REQJDBC
-IsConnectorAgentAlive'
resultvar=$?
if [ ${resultvar} -ne 0 ]
then
    # Start the REQJDBC Connector
    echo "$PROGID - $HOST: Starting CrossWorlds REQJDBC Connector. `date`"
    su - cwadmin '-c $CROSSWORLDS/bin/connector_manager_REQJDBC -start
    -c$CROSSWORLDS/config/REQJDBC.cfg'
    sleep $ADAPTER_WAIT
fi

# Test for Connector being active
su - cwadmin '-c $CROSSWORLDS/bin/connector_manager_REQJDBC
-IsConnectorAgentAlive'
resultvar=$?
if [ ${resultvar} -ne 0 ]
then
    echo "$PROGID - $HOST - Return code $resultvar: Not connected to REQJDBC
    Connector. `date`"
fi

echo "WICSHA Start up Complete"
exit 0
wics_ha_stop

#!/usr/bin/sh
# WICS Stop Script for HACMP 5.1
# Initialize variables
QM_NAME=ics422.queue.manager
CROSSWORLDS=/opt/IBM/WebSphereICS
HOST=`/bin/hostname -s`
PROGID=`echo $0 | sed 's%/usr/bin/%%g'`

echo "Entering WICSHA Stop"
# Stop Connector REQJDBC
echo "$PROGID - $HOST: Stopping Connector REQJDBC. `date`"
su - cwadmin '-c $CROSSWORLDS/bin/connector_manager_REQJDBC -stop'
resultvar=$?
if [ ${resultvar} -ne 0 ]
then
  echo "$PROGID - $HOST - Return code $resultvar: Did not stop Connector
  REQJDBC. `date`"
fi

# Stop Connector ORDJDBC
echo "$PROGID - $HOST: Stopping Connector ORDJDBC. `date`"
su - cwadmin '-c $CROSSWORLDS/bin/connector_manager_ORDJDBC -stop'
resultvar=$?
if [ ${resultvar} -ne 0 ]
then
  echo "$PROGID - $HOST - Return code $resultvar: Did not stop Connector
  ORDJDBC. `date`"
fi

# Stop Connector WMQI
echo "$PROGID - $HOST: Stopping Connector WBIMB `date`"
su - cwadmin '-c $CROSSWORLDS/bin/connector_manager_WBIMB -stop'
resultvar=$?
if [ ${resultvar} -ne 0 ]
then
  echo "$PROGID - $HOST - Return code $resultvar: Did not stop Connector
  WBIMB. `date`"
fi

# Stop WICS
echo "$PROGID - $HOST: Stopping ICS. `date`"
su - cwadmin '-c $CROSSWORLDS/bin/ics_manager -stopgraceful'
resultvar=$?
if [ ${resultvar} -ne 0 ]
then
  su - cwadmin '-c $CROSSWORLDS/bin/ics_manager -stop'
  resultvar=$?
  if [ ${resultvar} -ne 0 ]
  then
echo "WICS -stop failed, attempting to kill WICS process"
su - cwadmin '-c $CROSSWORLDS/bin/ics_manager -kill'
if [ ${resultvar} -ne 0 ]
then
  echo "$PROGID - $HOST - Return code $resultvar: Could not kill ICS. `date`"
  exit $resultvar
fi
fi

# Stop MQSeries
echo "$PROGID - $HOST: Stopping MQSeries. `date`"
su cwadmin -c $CROSSWORLDS/bin/hascripts/mq/hamqm_stop_su $QM_NAME 10
$CROSSWORLDS/bin/hascripts/mq
resultvar=$?
if [ ${resultvar} -ne 0 ]
then
  echo "$PROGID - $HOST - Return code $resultvar: Did not stop MQSeries. `date`"
  notStopped=$resultvar
fi

# Stop IBM ORB
su - cwadmin '-c $CROSSWORLDS/bin/hascripts/wics_orb_stop'
resultvar=$?
if [ ${resultvar} -ne 0 ]
then
  echo "$PROGID - $HOST - Return code $resultvar: Could not stop IBM ORB. `date`"
  exit $resultvar
fi
sleep 60
echo "WICSHA Stop Complete"
exit 0
wics_ha_monitor

#!/usr/bin/sh
# WICS/MQ/WICS Adapter Monitor Script for HACMP 5.1

# Initialize variables
APP_WAIT=30
HOST="/bin/hostname -s"
PROGID="echo $0 | sed 's%/usr/bin/%%g'"
CROSSWORLDS=/opt/IBM/WebSphereICS
export CROSSWORLDS

echo "Entering WICSHA Monitoring"
# Monitor MQM
$CROSSWORLDS/bin/hascripts/wics_mq_restart
resultvar=0
if [ ${resultvar} -ne 0 ]
then
    echo "WICS QM is down, Invoking wics_ha_stop"
    exit ${resultvar}
else
    echo "WICS QM is Alive"
fi

# Monitor ORB
$CROSSWORLDS/bin/hascripts/wics_orb_restart
resultvar=0
if [ ${resultvar} -ne 0 ]
then
    echo "Not able to restart IBM ORB"
    exit ${resultvar}
else
    echo "ORB is Alive"
fi

# Monitor WICS
$CROSSWORLDS/bin/hascripts/ics_restart
resultvar=0
if [ ${resultvar} -ne 0 ]
then
    echo "Not able restart WICS"
    exit ${resultvar}
else
    echo "WICS is Alive"
fi

# Monitor WICS Adapters
$CROSSWORLDS/bin/hascripts/wics_adapter_restart
resultvar=0
if [ ${resultvar} -ne 0 ]
then
    echo "Not able restart WICS Adapters"
    echo "Adapter failure does not trigger failover"
    exit 0
else
    echo "WICS Adapters are Alive"
fi
echo "Exiting WICSHA Monitoring"
exit 0
wics_orb_start

#!/bin/sh
.
/opt/IBM/WebSphereICS/bin/CWSharedEnv.sh
#Start the IBM ORB's name service
# Need to pass repository file as argument to PersistentNameServer.sh
FILE=$1
ORB_WAIT=3

if [ -f ${CROSSWORLDS}/logs/NameServer.log ]; then
    mv -f ${CROSSWORLDS}/logs/NameServer.log
    ${CROSSWORLDS}/logs/NameServer.log_old
fi
${CROSSWORLDS}/bin/PersistentNameServer_ha ${FILE} >
${CROSSWORLDS}/logs>NameServer.log 2>&1
processid=`grep "Transient Name Server Process Id"
${CROSSWORLDS}/logs>NameServer.log | cut -d" " -f8`
echo 'Name Server Process Id = '${processid}
sleep $ORB_WAIT
status=`ps -ef | grep com.ibm.CosNaming.TransientNameServer | grep -v grep | wc -l`
if [{$status} -eq 1]
then
    echo 'Successfully started IBM ORB Name Server'
    exit 0;
else
    echo 'Unable to start IBM ORB Name Server'
    # mv ${CROSSWORLDS}/logs>NameServer.log ${CROSSWORLDS}/logs>NameServer.err
    # mv ${CROSSWORLDS}/logs>NameServer.log_old
    ${CROSSWORLDS}/logs>NameServer.log
    exit 1;
fi
exit 0
wics_orb_stop

#!/bin/sh
. /opt/IBM/WebSphereICS/bin/CWSharedEnv.sh

processid=`grep "Transient Name Server Process Id" ${CROSSWORLDS}/logs/NameServer.log | cut -d" " -f8`
echo 'Name Server Process Id = ' ${processid}
kill -9 ${processid}
resultvar=$?
if [ ${resultvar} -eq 0 ]
then
  echo 'Stopped Name Server'
  exit 0
else
  if [ ${resultvar} -eq 2 ]
  then
    echo 'Name server seems to be stopped already'
    exit 0
  else
    echo 'Name Server could not be stopped'
    exit 1
  fi
fi

wics_orb_test

#!/bin/sh
out=`ics_manager -IsServerAlive`
status=`echo $out|grep 'Cannot connect to the CORBA name server'| wc -l`
if [ ${status} -ne 1 ]
then
  echo 'IBM ORB Name Server is running'
  exit 0
else
  echo 'IBM ORB Name Server not running'
  exit 1
fi
wics_orb_restart

#!/usr/bin/sh
# WICS IBM ORB Restart Script for HACMP 5.1

# Initialize variables
APP_WAIT=40
HOST=`/bin/hostname -s`
PROGID=`echo $0 | sed 's%/usr/bin/%%g'`
CROSSWORLDS=/opt/IBM/WebSphereICS
export CROSSWORLDS

# Test the IBM ORB
echo "$PROGID - $HOST: testing IBM ORB. `date``
su - cwadmin '-c $CROSSWORLDS/bin/hascripts/wics_orb_test'
resultvar=$?
if [ ${resultvar} -ne 0 ]
then
  echo "$PROGID - $HOST - Restart IBM ORB. `date`"
su - cwadmin ' -c $CROSSWORLDS/bin/hascripts/wics_orb_start'
  resultvar=$?
  if [ ${resultvar} -ne 0 ]
  then
    echo "IBM ORB restart failed"
    exit $resultvar
fi
  echo "IBM ORB restarted"
fi
exit $resultvar
wics_mq_restart

#!/usr/bin/sh
# MQ Restart Script for HACMP 5.1
# This is called when MQ has died and we want to restart

# Initialize variables
HOST=`/bin/hostname -s`
PROGID=`echo $0 | sed 's%/usr/bin/%%g'`
CROSSWORLDS=/opt/IBM/WebSphereICS
export CROSSWORLDS
QM_NAME=ics422.queue.manager
PORT=1415

# Use the MQ Support Pack scripts to find out if ICS QM is running
su - cwadmin -c "echo "ping qmgr" | runmqsc ${QM_NAME}" > /dev/null 2>&1
pingresult=$?
# pingresult will be 0 on success; non-zero on error (man runmqsc)
if [ $pingresult -ne 0 ]
then
  echo "WICS QM is unavailable"
  exit 1
else
  echo "WICS QM is running"
fi

# Check MQ Listener and Command Server running
cnt=`ps -ef | grep runmqlsr | grep -v grep | grep $QM_NAME | grep $PORT | wc -l`
if [ $cnt -eq 0 ]
then
  echo "WICS QM Listener is unavailable"
  exit 1
else
  echo "WICS QM Listener is running"
fi

cnt=`ps -ef | grep amqpcsea | grep -v grep | grep $QM_NAME | wc -l`
if [ $cnt -eq 0 ]
then
  su - cwadmin -c strmqcsv $QM_NAME
  resultvar=$?
  if [ ${resultvar} -ne 0 ]
  then
    echo "$PROGID - $HOST - MQ Command Server could not start. `date`"
    echo "$PROGID - $HOST - please check MQ logs. `date`"
  fi
fi

exit 0
ics_restart

#!/usr/bin/sh
# WICS Only Restart Script for HACMP 5.1

# Initialize variables
HOST=`/bin/hostname -s`
PROGID=`echo $0 | sed 's%/usr/bin/%%g'`
APP_WAIT=30
CROSSWORLDS=/opt/IBM/WebSphereICS
export CROSSWORLDS

# Check if ICS is alive
echo "PROGID - HOST: Checking if WICS is alive. `date`"
su - cwadmin '-c $CROSSWORLDS/bin/ics_manager -IsServerAlive'
resultvar=$?
if [ ${resultvar} -ne 0 ]
then
  # Start WICS
  echo "PROGID - HOST: Starting ICS. `date`"
su - cwadmin '-c $CROSSWORLDS/bin/ics_manager -start -c$CROSSWORLDS/config/InterchangeSystem.cfg'
else
  echo "PROGID - HOST: WICS is alive. `date`"
  exit 0
fi
#!/usr/bin/sh

# WICS Start Script for HACMP 5.1

# Initialize variables
ADAPTER_WAIT=30
HOST="/bin/hostname -s"
PROGID=`echo $0 | sed 's%/usr/bin/%%g'`
CROSSWORLDS=/opt/IBM/WebSphereICS
export CROSSWORLDS

# Test for Connector being active
echo "$PROGID - $HOST: Testing WICS ORDJDBC Adapter Alive. `date`"
su - cwadmin '-c $CROSSWORLDS/bin/connector_manager_ORDJDBC -IsConnectorAgentAlive'
resultvar=$?
if [ ${resultvar} -ne 0 ]; then
    # Start the ORDJDBC Connector
    echo "$PROGID - $HOST: Restarting WICS ORDJDBC Connector. `date`"
su - cwadmin '-c $CROSSWORLDS/bin/connector_manager_ORDJDBC -start -c$CROSSWORLDS/config/InterchangeSystem.cfg'
else
    echo "ORDJDBC Adapter is Alive"
fi

# Test for Connector being active
echo "$PROGID - $HOST: Testing WICS WBIMB Adapter Alive. `date`"
su - cwadmin '-c $CROSSWORLDS/bin/connector_manager_WBIMB -IsConnectorAgentAlive'
resultvar=$?
if [ ${resultvar} -ne 0 ]; then
    # Start the WMQI Connector
    echo "$PROGID - $HOST: Restarting CrossWorlds WBIMB Connector. `date`"
su - cwadmin '-c $CROSSWORLDS/bin/connector_manager_WBIMB -start -c$CROSSWORLDS/config/InterchangeSystem.cfg'
else
    echo "WBIMB Adapter is Alive"
fi

# Test for Connector being active
echo "$PROGID - $HOST: Testing CrossWorlds REQJDBC Adapter Alive. `date`"
su - cwadmin '-c $CROSSWORLDS/bin/connector_manager_REQJDBC -IsConnectorAgentAlive'
resultvar=$?
if [ ${resultvar} -ne 0 ]; then
# Start the REQJDBC Connector

```
echo "\$PROGID - \$HOST: Restarting WICS REQJDBC Connector. `date`"

su - cwadmin '-c $CROSSWORLDS/bin/connector_manager_REQJDBC -start
-c$CROSSWORLDS/config/REQJDBC.cfg'

else

echo "REQJDBC Adapter is Alive"

fi

exit 0
```
InterChange Server remote adapter scripts

The below scripts are useful to set up WICS Adapters in an HA environment in a separate resource group using remote agent technology. These scripts start, stop, and monitor the WICS adapters and the Adapter QM.

For MQ start, stop, and monitoring, these scripts refer the MQ HA scripts that are part of the MQ support pac.

rmt_adap_ha_start

#!/usr/bin/sh
# WICS REMOTE ADAPTER Start Script for HACMP 5.1
# Initialize variables
ADAPTER_WAIT=30
MQ_WAIT=15
HOST=`/bin/hostname -s`
PROGID=`echo $0 | sed 's%/usr/bin/%%g'`
RMT_ADAP_HOME=/opt/IBM/WICSAdapters
export RMT_ADAP_HOME
QM_NAME=WICSADAPTERS
PORT=1421

# Start MQSeries
su - mqm -c "echo "ping qmgr" | runmqsc ${QM_NAME}" > /dev/null 2>&1
pingresult=$?
# pingresult will be 0 on success; non-zero on error (man runmqsc)
if [ ${pingresult} != "0" ]
then
    echo "$PROGID - $HOST: Starting MQSeries. `date`"
    su - mqm -c $RMT_ADAP_HOME/bin/hascripts/mq/hamqm_start_su $QM_NAME
    resultvar=$?
    if [ ${resultvar} -ne 0 ]
    then
        echo "$PROGID - $HOST - Return code $resultvar: Did not start MQ. `date`"
        exit 1
    fi
else
    echo "$PROGID - $HOST - WMQ Queue Manager already started. `date`"
fi

# Start MQ Listener and Command Server if not already started
# if listener isn't running exit with return code
# echo "$PROGID - $HOST: Checking MQ Listener and Command Server. `date`"
cnt=`ps -ef | grep runmqlsr | grep -v grep | grep $QM_NAME | grep $PORT | wc -l`
if [ $cnt -eq 0 ]
then
  echo "$PROGID - $HOST: Starting MQSeries Listener. `date`"
  su - mqm -c runmqlsr -m $QM_NAME -t TCP -p $PORT &
  resultvar=$?
  if [ ${resultvar} -ne 0 ]
  then
    echo "$PROGID - $HOST - Return code $resultvar: Could not start MQ Listener. `date`"
    exit $resultvar
  else
    echo "$PROGID - $HOST - Return code $cnt: MQ Listener started. `date`"
    fi
fi

# if the command server isn't running do not fail over
cnt=`ps -ef | grep amqpcsea | grep -v grep | grep $QM_NAME | wc -l`
if [ $cnt -eq 0 ]
then
  echo "$PROGID - $HOST - Starting MQ Command Server. `date`"
  su - mqm -c strmqcsv $QM_NAME
  else
    echo "$PROGID - $HOST - MQ Command Server already started. `date`"
  fi
sleep $MQ_WAIT

# Test for Connector being active
echo "$PROGID - $HOST: Testing CrossWorlds RMTHAORDJDBC Connector Alive. `date`"
$RMT_ADAP_HOME/bin/hascripts/rmt_adap_restart
resultvar=$?
if [ ${resultvar} -ne 0 ]
then
  echo "Not able start WICS Remote Adapters"
  exit 0
else
  echo "WICS Remote Adapters are Alive"
fi

echo "WICS REMOTE ADAPTER Start up Complete"
exit 0
#!/usr/bin/sh
# WICS Stop Script for HACMP 5.1
# Initialize variables
QM_NAME=WICSADAPTERS
RMT_ADAP_HOME=/opt/IBM/WICSAdapters
HOST="/bin/hostname -s"
PROGID=`echo $0 | sed 's%/usr/bin/%%g'`

echo "Entering WICS REMOTE ADAPTER HA Stop"
# Stop Connector ORDJDBC
echo "$PROGID - $HOST: Stopping Connector RMTHAORDJDBC. `date`"
su - cwadmin '-c $RMT_ADAP_HOME/bin/connector_manager_RMTHAORDJDBC -kill'
resultvar=$?
if [ ${resultvar} -ne 0 ]
then
  echo "$PROGID - $HOST - Return code $resultvar: Did not stop Remote Connector RMTHAORDJDBC. `date`"
fi

# Stop MQSeries
echo "$PROGID - $HOST: Stopping MQSeries. `date`"
su mqm -c $RMT_ADAP_HOME/bin/hascripts/mq/hamqm_stop_su WICSADAPTERS 20
$RMT_ADAP_HOME/bin/hascripts/mq
resultvar=$?
if [ ${resultvar} -ne 0 ]
then
  echo "$PROGID - $HOST - Return code $resultvar: Did not stop MQSeries. `date`"
  notStopped=$resultvar
fi

sleep 20
echo "WICS REMOTE ADAPTER HA Stop Complete"
exit 0
rmt_adap_ha_monitor

#!/usr/bin/sh
# WICS/MQ/WICS Adapter Monitor Script for HACMP 5.1
# Initialize variables
APP_WAIT=30
HOST="/bin/hostname -s"
PROGID="echo $0 | sed 's%/usr/bin/%%g'"
RMT_ADAP_HOME=/opt/IBM/WICSAdapters
export RMT_ADAP_HOME

echo "Entering WICS REMOTE ADAPTER HA Monitoring"
# Monitor MQM
$RMT_ADAP_HOME/bin/hascripts/rmt_mq_restart
resultvar=$?
if [ ${resultvar} -ne 0 ]
then
  echo "WICS REMOTE ADAPTER QM is down, Invoking Stop Script"
  exit ${resultvar}
else
  echo "WICS ADAPTER QM is Alive"
fi

# Monitor WICS Adapters
$RMT_ADAP_HOME/bin/hascripts/rmt_adap_restart
resultvar=$?
if [ ${resultvar} -ne 0 ]
then
  echo "Not able restart WICS Remote Adapters"
  echo "Adapter failure does not trigger failover"
  exit 0
else
  echo "WICS Remote Adapters are Alive"
fi

echo "Exiting WICS Remote Adapter HA Monitoring"
exit 0
#!/usr/bin/sh
# WICS REMOTE ADAPTER Restart Script for HACMP 5.1
# Initialize variables
HOST=`/bin/hostname -s`
PROGID=`echo $0 | sed 's%/usr/bin/%%g'`
RMT_ADAP_HOME=/opt/IBM/WICSAdapters
export RMT_ADAP_HOME

# Test for Connector being active
echo "$PROGID - $HOST: Testing WICS RMTHAORDJDBC Adapter Alive. `date`"
su - cwadmin '-c $RMT_ADAP_HOME/bin/connector_manager_RMTHAORDJDBC -IsConnectorAgentAlive'
out=`ps -ef | grep RMTHAORDJDBC | grep -v grep | wc -l`
#resultvar=$?
echo $out
if [ ${out} -ne 1 ]
then
  # Start the ORDJDBC Connector
  echo "$PROGID - $HOST: Restarting WICS ORDJDBC Connector. `date`"
  su - cwadmin '-c $RMT_ADAP_HOME/bin/connector_manager_RMTHAORDJDBC -start -c$RMT_ADAP_HOME/config/RMTHAORDJDBCConnector.cfg'
else
  echo "RMTHAORDJDBC Adapter is Alive"
  echo "No need to restart"
fi
exit 0
rmt_mq_restart

#! /usr/bin/sh
# REMOTE ADAPTER MQ Restart Script for HACMP 5.1
# This is called when MQ has died and we want to restart
# Initialize variables
HOST=/bin/hostname -s
PROGID=`echo $0 | sed 's%/usr/bin/%%g'`
QM_NAME=WICSADAPTERS
PORT=1421

# Use the MQ Support Pack scripts to find out if ICS QM is running
su - mqm -c "echo "ping qmgr" | runmqsc ${QM_NAME}" > /dev/null 2>&1
pingresult=$?
# pingresult will be 0 on success; non-zero on error (man runmqsc)
if [ $pingresult -ne 0 ]
then
    echo "WICS REMOTE ADAPTER QM is unavailable"
    exit 1
else
    echo "WICS REMOTE ADAPTER QM is running"
fi

# Check MQ Listener and Command Server running
cnt=`ps -ef | grep runmqlsr | grep -v grep | grep $QM_NAME | grep $PORT | wc -l`
if [ $cnt -eq 0 ]
then
    echo "WICS REMOTE ADAPTER QM Listener is unavailable"
    exit 1
else
    echo "WICS QM REMOTE ADAPTER Listener is running"
fi

cnt=`ps -ef | grep amqpcsea | grep -v grep | grep $QM_NAME | wc -l`
if [ $cnt -eq 0 ]
then
    su -cwadmin -c strmqcsv $QM_NAME
    resultvar=$?
    if [ ${resultvar} -ne 0 ]
    then
        echo "$PROGID - $HOST - MQ Command Server could not start. `date`"
        echo "$PROGID - $HOST - please check MQ logs. `date`"
    fi
fi
exit 0
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/SG246328

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, SG246328.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

<table>
<thead>
<tr>
<th>File name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SG246328Scripts.zip</td>
<td>Zipped HA scripts from sample implementation</td>
</tr>
</tbody>
</table>
System requirements for downloading the Web material

The following system configuration is recommended:

**Hard disk space:** 10 MB minimum  
**Operating System:** Windows 2000 with Service Pack 4  
**Processor:** 1.5 GHz or higher  
**Memory:** 512 MB

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.
# Abbreviations and acronyms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASBO</td>
<td>Application Specific Business Object</td>
</tr>
<tr>
<td>FASTT</td>
<td>Fibre Array Storage Technology</td>
</tr>
<tr>
<td>FIFO</td>
<td>First In First Out</td>
</tr>
<tr>
<td>GBO</td>
<td>Generic Business Object</td>
</tr>
<tr>
<td>HACMP</td>
<td>High Availability Cluster Multiprocessing</td>
</tr>
<tr>
<td>HACMP/ES</td>
<td>High Availability Cluster Multiprocessing / Enhanced Scalability</td>
</tr>
<tr>
<td>IBM</td>
<td>International Business Machines Corporation</td>
</tr>
<tr>
<td>IDL</td>
<td>Interface Definition Language</td>
</tr>
<tr>
<td>IPAT</td>
<td>IP Address Takeover</td>
</tr>
<tr>
<td>ITSO</td>
<td>International Technical Support Organization</td>
</tr>
<tr>
<td>JMS</td>
<td>Java Messaging Support</td>
</tr>
<tr>
<td>JVM</td>
<td>Java Virtual Machine</td>
</tr>
<tr>
<td>LPAR</td>
<td>Logical Partitioning</td>
</tr>
<tr>
<td>MSCS</td>
<td>Microsoft Cluster Server</td>
</tr>
<tr>
<td>NIC</td>
<td>Network Interface Card</td>
</tr>
<tr>
<td>ORB</td>
<td>Object Request Broker</td>
</tr>
<tr>
<td>QA</td>
<td>Quality Assurance</td>
</tr>
<tr>
<td>RAID</td>
<td>Redundant Arrays of Inexpensive Disks</td>
</tr>
</tbody>
</table>
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 226. Note that some of the documents referenced here may be available in softcopy only.

► Fibre Array Storage Technology A FASTT Introduction, SG24-6246

Other publications

These publications are also relevant as further information sources:

► IBM High Availability Services - for WebSphere MQ (IBM internal use only)
► WebSphere MQ Queue Manager Clusters, SC34-6061
► MQSeries for AIX - Implementing with HACMP (SupportPac MC63)
► Configuring MQSeries Integrator for AIX with HACMP (SupportPac IC61)
► WebSphere Interchange Server with HACMP (SupportPac XC63)
► IBM WebSphere InterChange Server System Administration Guide V4.2.2
► IBM WebSphere InterChange Server Implementation Guide for WebSphere InterChange Server V4.2.2
► High Availability Cluster Multi-Processing for AIX Planning and Installation Guide Version 5.1, SC23-4861

Online resources

These Web sites and URLs are also relevant as further information sources:

► WebSphere MQ family SupportPacs
► WebSphere Business Integration Information Center
How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services
Index

A
Active / Active configuration 18
Active / Passive configuration 18
adding applications to HACMP control 85–86
architecture, solution architecture 68–71
    queue manager clustering 71
availability
    continuous 13
    high 13
availability, degree of
    number of nines 12–13
availability, WBI components
    InterChange Server 30–33
    Message Broker 33
    WebSphere MQ 26–30

B
business challenge 56
business scenario 55–63
    See sample scenario

C
cluster design, HA WBI 34–45
cluster group 17
    See also resource group, service group
cluster topologies 17–18
    Active / Active configuration 18
    Active / Passive configuration 18
    Hot Standby configuration 18
    Mutual Takeover configuration 18
    N+1 Standby Takeover configuration 17
    One-sided Takeover configuration 18
cluster, HA 16
cluster, queue manager
    illustration 29
    overview 28–30
    use in solution 71, 77
    illustration 77
clustering
    high availability clustering 15
    advantages of 19
    cluster topologies 17–18
    failback 17
    failover 17
    heartbeat 19
    IP Address Takeover 18
    node 16
    resource group 16–17
    shared external disk device 19
high availability solutions
    HACMP 16
    HP Serviceguard 16
    Microsoft Cluster Server 16
    Sun Cluster 16
    VERITAS Cluster Server 16
queue manager clustering
    illustration 29
    overview 28–30
    use in solution 71
component availability, WBI
    InterChange Server 30–33
    Message Broker 33
    WebSphere MQ 26–30
component dependencies, WBI 22–26
    InterChange Server / WebSphere BI Adapters
    and IBM Java ORB 24
    InterChange Server and Message Broker 22
    InterChange Server and WebSphere MQ 22
    Message Broker and WebSphere BI Adapters
    24
    Message Broker and WebSphere MQ 24
    WebSphere BI Adapters and IBM Java ORB 24
    WebSphere BI Adapters and InterChange Server 23
    WebSphere BI Adapters and MQ 23
component, preparing for HA 72–80
    See also preparation, preparing components for HA
continuous availability 13
continuous operation 13

D
data store, sample implementation 62
dependencies, WBI components 22–26
    InterChange Server / WebSphere BI Adapters
and IBM Java ORB 24
InterChange Server and Message Broker 22
InterChange Server and WebSphere MQ 22
Message Broker and WebSphere BI Adapters 24
Message Broker and WebSphere MQ 24
WebSphere BI Adapters and IBM Java ORB 24
WebSphere BI Adapters and InterChange Server 23
WebSphere BI Adapters and MQ 23

design considerations, HA WBI 21–51
  capacity planning 46
  component dependencies 22–26
  HA cluster design 34–45
  InterChange Server availability 30–33
  maintenance planning 46–50
  performance 50–51
  WebSphere MQ availability 26–30

E
eexample scenario 55–63
  See sample scenario
  external disk device, shared 19

F
  failback 17
  failover 17
  fallback 17
  fault tolerant 14

H
  HACMP 16
    adding applications to HACMP control 85–86
    resource group 16–17
  hardware used in solution implementation 84
  heartbeat 19
  high availability 13
    business challenge 56
  hints and tips
    general recommendations 122
    InterChange Server 123
    WebSphere Business Integration Adapters 123
  Hot Standby configuration 18
  HP Serviceguard 16

I
  IC61: Configuring MQSeries Integrator for AIX with

HACMP 75, 80, 225
  implementing, solution implementation 65–86
    adding applications to HACMP control 85–86
    architecture 68–71
    cluster layout 66
      illustration 67
    hardware 84
    initial environment 72
    preparing components for HA
      See preparation, preparing components for HA
    queue manager clustering 71, 77
      illustration 77
    scripts, HA scripts
      See scripts, used in sample implementation
    software 84
    system setup 84–85
    volume group layout 67
      illustration 68
  IP Address Takeover 18

M
  marooned messages 27
  MC63: WebSphere MQ for AIX - Implementing with
    HACMP 72, 80, 84, 225
  message availability 27
    automatic availability 28
    continuous availability 28
    HA clustering 27
    inherent availability 28
    queue manager clustering 27
  message resilience 26
  Microsoft Cluster Server 16
    cluster group 17
  Mutual Takeover configuration 18

N
  N+1 Standby Takeover configuration 17
  node 16
  nonpersistent messages 27
  number of nines 12–13

O
  One-sided Takeover configuration 18

P
  persistent messages 27
preparation, preparing components for HA 72–80
counting Active / Active Message Broker 76–77
illustration 77
counting Active / Passive Message Broker 75–76
counting InterChange Server and adapters 78–79
moving InterChange Server and adapters to shared disk 78–79
counting remote adapters 75
counting system manager 78
counting the ORB 79
shared disk layouts 80
counts WebSphere MQ 72–74
manually moving queue managers to shared disk 72–74
object configuration 74
initial environment 72
queue manager cluster 77
illustration 77

Q
queue manager cluster
illustration 29
overview 28–30
use in solution 71, 77
illustration 77

R
Redbooks Web site 226
Contact us xiv
representative test plan 87–93
general introduction 88
InterChange Server and InterChange Server adapter failover 91
Message Broker adapter failover 90
verify basic machine failover 88
WebSphere MQ and Message Broker failover 89
resource group 16–17
See also cluster group, service group

S
sample scenario 55–63
communication 63
data store 62
description 56
design 56
implementation 65–86
See implementing, solution implementation
objective 56
overview 56
solution 57–62
Active / Active Message Broker takeover configuration 60–62
Active / Passive takeover configuration 58–60
cluster setup 57
transport mechanism 63
scripts, used in sample implementation
discussion 80–84
InterChange Server and adapters 81–83
InterChange Server remote adapter resource group 83
Message Broker 80
remote WebSphere Business Integration Adapters 81
WebSphere MQ 80
hacrtmqm 128
hadltmqm 131
halinkmqm 133
hamqm_lsnr_start 181
hamqm_lsnr_start_su 182
hamqm_lsnr_stop 184
hamqm_lsnr_stop_su 185
hamqm_qm_directory 136
hamqm_running 139
hamqm_start 141
hamqm_start_su 142
hamqm_status 144
hamqm_stop 145
hamqm_stop_su 146
hamqsi_applmon.MB2BRK 192
hamqsi_applmon.MBBRK 192
hamqsi_monitor_broker_as 187
hamqsi_qm_datapath 164
hamqsi_start_broker 167
hamqsi_start_broker_as 170
hamqsi_start_broker_as.MB2BRK 186
hamqsi_start_broker_as.MBBRK 186
hamqsi_stop_broker 175
hamqsi_stop_broker_as 178
hamqsi_stop_broker_as.MBBRK 186
hamqsiaddbrokerstandby 154
hamqsiscreatebroker 150
Highly Available WebSphere Business Integration Solutions

Develop a WebSphere Business Integration solution under HA control

Explore middleware behavior under HA

Example implementation with supporting HA scripts

Your company has customers on five continents and in every time zone in this on-demand world. Applications throughout the Enterprise run non-stop to support your business. 7x24x365. Or, maybe it is 5x12. If you’re going to satisfy your uptime requirement, high availability is an important part of your system design.

This IBM Redbook examines valuable design and implementation considerations involved in deploying your highly available WebSphere Business Integration solution, and details the design, implementation, and execution of a sample HA WebSphere Business Integration solution.

For more information: ibm.com/redbooks