

iSeries and High Availability

An e-Business Perspective

**Nadir Amra
Steven Janssen
Darin Scherer
Steve Simonson**

NOTE: Be sure to read the information in “Notices” on page ix before using the information presented in this document.

First Edition (May 2004)

This edition applies to Version 5 Release 2 Modification 0 of the OS/400 operating system and IBM WebSphere Application Server Network Deployment V5.0 for iSeries.

(C) Copyright International Business Machines Corporation 2004. All rights reserved.

Note to U.S. Government Users - Documentation related to restricted rights - Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Table of Contents

Notices	ix	Notification and Activation Procedures	38
Trademarks	x	Notification Procedures	38
Prefacexi	Activation Plan	39
Summary of Changesxi	Recovery Procedures	39
About the Authorsxi	Sequence of Recovery Activities	39
Chapter 1. Introduction	1	Recovery Procedure Highlights	40
Objectives	2	Reconstitution	41
Base Technologies Referenced in this Document.	2	Data Center Availability Planning	41
Document Organization	3	Infrastructure, the First Line of Defense	42
Chapter 2. High Availability Concepts	5	Data Center Planning	42
Availability Defined	5	Space and Growth Considerations	42
Levels Of Availability	6	Fundamental Elements of Data Center	
Process (Application) and Data Availability	7	Design	43
Data Center vs. Disaster Center	8	Preventative Controls	44
Factors Impacting Availability	8	Onsite Parts Maintenance Locker	44
How Much Availability Do I Need?	9	Data Center Facilities Security	44
High Availability Principles	10	Disaster Recovery Planning and	
High Availability Rules for Runtime		Infrastructure Assessment	44
Environments	10	Distributed Systems Architecture	
Redundancy	11	Considerations	45
Request Distribution	11	Server Architecture	45
Resources for Learning	12	Server Operating System Clustering	
Chapter 3. Developing an Availability Plan	15	Technologies	45
One Size Does Not Fit All	15	Capacity Planning	45
Shapers of Recovery/Availability Strategies	15	System Software	45
Types of Contingency Plans	16	Configuration Management	46
Business Continuity Planning (BCP)	17	Problem and Change Management	46
Business Recovery Plan (BRP)	17	Disk Storage Architecture Considerations	46
Continuity of Operations Plan (COOP)	18	Advanced Storage Subsystem Functionality .	47
Incident Response Plan (IRP)	18	Network Architecture Considerations	47
Occupant Emergency Plan (OEP)	18	Network Architecture	47
Disaster Recovery Plan (DRP)	18	Network Documentation	48
Disaster Recovery Planning	19	Network Management	48
Disaster Recovery Planning Project Phases.	20	Remote Access Considerations	49
Project Initiation and Team Selection Phase	21	WAN Considerations	49
Data Collection and Critical Needs Phase	24	Network Security Considerations	49
Risk Analysis Phase	26	Network Components	49
Data Protection Phase	29	Resources for Learning	51
Recovery Plan Phase	31	Chapter 4. Availability and The iSeries	
Testing and Training Phase	35	Server	53
Change Management Phase	36	Single Server Environment	53
Disaster Recovery Plan Execution	37	Hardware Availability Features	53
		Power Subsystem	53
		Disk Subsystem	54

I/O Subsystem	54	The plug-in Configuration File	91
Memory	55	HTTP Plug-in Workload Management	
Hardware Service	55	Policies	93
OS/400 and System Software Availability		EJB Workload Management (WLM)	95
Features	55	Looking Up an EJB Home	96
Database	56	How EJBs Participate in Workload	
Storage Management	61	Management	98
Save/Restore	62	EJB Workload Management Server Selection	
TCP/IP	63	Policy	100
Security	63	Session Management	102
System Software Maintenance	64	Session Affinity	102
Logical Partitions (LPARs)	64	Session Persistence	103
Multiple Server Environments	65	Session Identifiers	105
Clusters	65	Web Services	107
How a Cluster Works	65	Web Service Client (requester)	108
Cluster Basics	66	Web Service Provider	108
The Importance of Having a Backup and Recovery		IBM WebSphere UDDI Registry	108
Strategy	72	Web Service Gateway	109
Business Continuity and Recovery Services	73	Resource Providers	109
Resources for Learning	73	JCA Resources Adapters	109
Chapter 5. Programming Architectures 77		JDBC Resources	110
WebSphere Application Server Network		JMS Providers	112
Deployment V5 Architecture	77	Security	112
Network Deployment Configuration	77	User Registry	114
Cells, Nodes, and Servers	78	Authentication	114
Servers	78	Authorization	115
Nodes	79	Security Components	116
Cells	79	Security Flows	117
Servers	79	Administration	118
Application Server	79	Administrative Tools	118
JMS Server	79	Configuration Repository	119
Clusters	80	Centralized Administration	119
Containers	81	Master configuration repository	120
Web Container	81	The Flow of an Application	120
EJB Container	82	Common Gateway Interface (CGI) Architecture	122
Client Application Container	84	CGI Programming Model	122
Application Server Services	84	Persistent CGI	123
JCA Services	84	The Flow of an Application	124
Transaction Service	85	Resources for Learning	125
Object Request Broker (ORB) Service	85	Chapter 6. Runtime Component	
Name Service	86	 Considerations for High Availability 127	
Security Service	88	Identifying Runtime Components in an e-Business	
Dynamic Caching	88	Environment	127
Message Listener Service	88	Client (User) Component	127
Admin Service	89	Firewall Component	128
PMI Service	89	Load Balancer Component	128
Virtual Hosts	89	Web Server Component	128
HTTP Plug-in Workload Management (WLM)	90	Application Server Component	128
Request Processing	90		

Database Server Component	129
Shared File System Component	129
Runtime Components and High Availability	129
Chapter 7. Client Considerations for High Availability	133
Web-based Clients	133
Stand-alone Java Application Clients	133
Chapter 8. Firewall Considerations for High Availability	135
Resources for Learning	135
Chapter 9. Load balancer Considerations for High Availability	137
WebSphere Edge Server Load Balancer	137
Dispatcher and Server Affinity	139
“Stickyness” to Source IP Address Affinity	140
Passive Cookie Affinity	140
SSL Session ID Affinity	141
Dispatcher Forwarding Methods	141
Media Access Control (MAC) Forwarding	141
Network Address Translation (NAT) Forwarding	144
Network Address Port Translation (NAPT) Forwarding	146
Dispatcher Content-Based Routing (cbr forwarding)	148
Advisors	149
Custom Advisors	150
Dispatcher Availability Features	150
Dispatcher High Availability	151
Dispatcher Mutual High Availability	152
Resources for Learning	154
Chapter 10. HTTP Server Considerations for High Availability	155
Highly Available Web Server	155
How It Works	155
Heartbeat Monitoring	155
Session Data	157
Highly Available Web Server Models	157
Primary/Backup with Takeover IP Model	157
Primary/Backup with a Dispatcher Model	159
Peer Model	160
Model Considerations	160
Enabling CGI Programs for High Availability	161
Interaction Between HA CGI Program and HTTP Server	161
Resources for Learning	163

Chapter 11. WebSphere Application Server Considerations for High Availability	165
Overview	165
Web Container Clustering and Failover	167
RetryInterval and Operating System TCP Timeout	168
ConnectTimeout Setting	169
Network Failures and the Node Agent	170
Stream and Overloading Failover	170
Primary and Backup Servers Cluster Two-level Failover	171
HTTP Session Failover	172
Session Data Considerations	173
Session ID Considerations	173
Session Affinity and Failover	173
No Session Support Needed	173
No Session Information in the Client	174
Session Affinity Without Session Persistence	174
Session Persistence and Failover	174
Session Update Methods and Failover Session Data Loss	176
At the End of Servlet Service	176
Manually sync() in the Code Using the IBM Extension	176
Time-based (at a specified time interval)	176
EJB Container Clustering and Failover	177
EJB Client Redundancy and Bootstrap Failover Support	179
EJB WLM Routing	181
Cluster Aware Clients	181
Cluster Unaware Clients	181
Routing Algorithm	182
Location Service Daemon (LSD) Failover	183
EJB WLM Failover Behavior and Tuning	183
EJB Caching and Failover	187
Resource Redundancy	188
WebSphere and IP-Based Database Failover	188
WebSphere Connection Manager	188
TCP/IP and Java Toolbox Configuration	191
Application Considerations for StaleConnectionException	192
Impact of Database Failures	195
Administrative Servers and Administrative Actions	195
Persistent Session	195

Java/C++ Applications and Application Reconnecting	195	Tier 6: Zero or Little Data Loss	231
Enterprise beans	196	Tier 7: Highly Automated, Business Integrated Solution.	231
Web-based Clients, Applets, Servlets, and JSPs	196	Selecting the Optimum Disaster Recovery Solution 231	
Naming Service and Security Service .	196	Resources for Learning	232
Workload Management	196	Chapter 15. Recommended Topologies 233	
Deployment Manager and Node Agent High Availability	197	LPAR	233
Node Agent High Availability Considerations	198	HA Database Topologies	234
Impact of Node Agent Failures	199	Recommended Topologies (Data Center) . . .	235
Enhancing Node Agent High Availability	203	Peer Cells using Network Dispatcher	236
Deployment Manager High Availability Considerations	203	Primary/Backup Cells using Network Dispatcher 238	
Impact of Deployment Manager Failures	204	Primary/Backup Cells using HA Apache. . .	239
Enhancing Deployment Manager High Availability	207	Recommended Topologies (Disaster Recovery)	240
Resources for Learning	208	Data Centers with Single Server.	240
Chapter 12. Data Considerations for High Availability	209	Data Centers with Multiple Servers	241
Independent Disk Pools (aka Independent ASPs) 209		Data Centers with Multiple Servers and Intelligent Application Routing	242
Replication Solutions for High Availability . . .	211	Chapter 16. Configuring the Load Balancer	245
Software Replication Products	214	Configuring Dispatcher for MAC-Based Forwarding Method	245
iSeries SAN Solutions	215	Step 1. Prepare the Servers	246
ESS Copy Services	215	Step 2. Create Configuration Script	247
Remote OS/400 Mirroring with ESS	216	Step 3. Load Configuration Script into Dispatcher	250
Summary	217	Configuring Dispatcher for cbr Forwarding Method . 250	
Resources for Learning	217	Step 1. Prepare the Servers	251
Chapter 13. Back-End Application Considerations for High Availability	221	Step 2. Create Configuration Script	252
Programming Techniques for High Availability	222	Step 3. Load Configuration Script into Dispatcher	255
Commitment Control	222	Custom Advisors	255
Application Checkpointing	222	Sample Code for Custom Advisor	257
OS/400 Clusters	223	Chapter 17. Configuring HTTP Server	265
Chapter 14. Disaster Recovery Considerations for High Availability	225	Highly Available HTTP Server Using Primary/ Backup Model	265
Disasters: The Old, the New, and the Planned	226	Step 1. Prepare the iSeries Servers	266
Disaster Cost	227	Step 2. Create the Cluster	267
The Tiers of Disaster Recovery	227	Step 3. Update the HTTP Configuration Files . . 268	
Tier 0: Do Nothing, No Off-Site Data.	229	Chapter 18. Configuring WebSphere Application Server	273
Tier 1: Offsite Vaulting (PTAM)	229	Dual Cell Topology Implementation	274
Tier 2: Offsite Vaulting with a Hot Site (PTAM + Hot Site)	229	Step 1. Install Required Products	275
Tier 3: Electronic Vaulting	230		
Tier 4: Point-in-time Copies	230		
Tier 5: Transaction Integrity	230		

Step 2. Create Network Deployment Instances 275	Unplanned Outages	350
Step 3. Create Base Application Server Instances	Verification	351
Step 4. Federate Base Application Server Instances	Chapter 21. End-to-End Monitoring	353
Step 5. Create Horizontal Clusters	Glossary	357
Step 6. Configure Resources		
Step 7. Install Applications		
Step 8. Enable Session Persistence		
Step 9. Generate Plug-in File		
Step 10. Update TCP/IP configuration values . 304		
Chapter 19. Configuring to Make Data Highly Available		307
Configuring Switched Disk (IASP).		307
Step 1. System Hardware and Software Requirements Planning.		309
Step 2. Physical Cabling of HSL Loop		312
Step 3. I/O Change Resource Ownership for LPAR		313
Step 4. Create the OS/400 Cluster.		319
Step 5. Enable the Tower to be Switchable		320
Step 6. Create a Switchable Hardware Group with a New Disk Pool Called 'HADB'		322
Step 7. Restore INVEST Database Collection into the iASP		336
Step 8. Enable WebSphere Persistent Sessions Database to be Stored in the iASP		336
Step 9. Enable WebSphere Distributed Application Access to the IASP.		336
Step 10. Verify Switchable Disk Pool Configuration		337
Factors that can affect IASP failover or switchover time		338
Useful IASP tasks.		339
Resources for Learning		344
Chapter 20. Testing		347
Preparation		347
Applications Used to Drive Workload		347
Tool Used to Drive Workload.		347
The Tests		348
Infrastructure Availability Tests		348
Planned Outages		348
Unplanned Outages.		348
Application Availability Tests		349
Upgrading an Application		349
Planned Outages		350

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:

AS/400®	Enterprise Storage Server®	Seascape®
ibm.com®	FlashCopy®	Tivoli®
iSeries™	IBM®	TotalStorage®
ClusterProven®	Net.Data®	WebSphere®
DB2 Universal Database™	OS/400®	
DB2®	Redbooks™	

The following terms are trademarks of other companies:

Intel, Intel Inside (logos), MMX, and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

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

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

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

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

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

Preface

No time for down time. Things like e-business, globalization, industry consolidation, merger mania, and server consolidation are the forces setting new standards for server availability. Both the iSeries and AS/400 are known for legendary availability. This server's success in the commercial computing arena has made it the central database or enterprise server for a significant share of the industries around the world.

The corporate clients using the iSeries or AS/400 as the enterprise server are demanding minimal downtime, if not continuous availability. No time for planned outages and certainly no time for unplanned outages is fast becoming the norm. The luxury of having a window of time to perform system backups has vanished, along with the idea that you can risk the possibility of an unplanned outage. In the increasingly seamless world of e-business, where financial transactions depend on a chain of events, no link in the availability chain can be broken and certainly, the central engine of commerce, the server, cannot be down.

Availability is a discipline. It is a technology. It is a corporate culture. It involves systems management, application design, even organizational structure and above all, a commitment from the IT organization and the business to make it a focus. This document will help you understand how you can achieve world class results. Our goal is to arm you with knowledge so you are the informed decision maker in this critical area of IT management.

Summary of Changes

This section describes the technical changes made in this edition of the book and in previous editions. This edition may also include minor corrections and editorial changes that are not identified.

Summary of Changes for *iSeries and High Availability* as created or updated on May 1, 2004.

May 2004, First Edition

This revision reflects the addition, deletion, or modification of new and changed information described below.

New information

- Document created.

Changed information

- Not applicable.

About the Authors

This document was produced by a team from the Rochester, MN lab.

Nadir Amra is an advisory software engineer with IBM Systems and Technology Group. He is a member of the WebSphere Application Server for iSeries development team, focusing on WebSphere WLM and high availability, and has been involved in the design and development of OS/400 Internet technologies for more than eight years. He also is the team leader for Net.Data for OS/400.

Steven Janssen is a staff software engineer with IBM Systems and Technology Group. He is the team member for internal testing of selective iSeries high-availability solutions within the IBM eServer Customer Solutions Test organization. He has been involved with testing a variety of iSeries technologies throughout his IBM career. WebSphere products, DB2 UDB for iSeries and hardware have been his focus areas during the past several years.

Darin Scherer is an advisory software engineer with IBM Systems and Technology Group. He is the team leader for internal testing of selective iSeries high-availability solutions within the IBM eServer Customer Solutions Test organization. He has been involved with high-availability technologies throughout his IBM career. WebSphere Application Server and DB2 UDB for iSeries have been his focus areas during the past several years.

Steve Simonson is a senior software engineer with IBM Systems and Technology Group. He currently is a member of the OS/400 Work Management team. Steve has many years of development experience within the OS/400 database and communications areas. Most recently he spent 3 years working on eCommerce high availability configuration and testing with a focus on iSeries and OS/400.

INTRODUCTION

Availability is measured in terms of *outages*, which are periods of time when the server is not available to users. During a *planned outage* (also called a scheduled outage), you deliberately make your system unavailable to users. You might use a scheduled outage to run batch work, back up your server, or apply fixes.

An *unplanned outage* (also called an unscheduled outage) is usually caused by a failure. A failure can be hardware related, such as disk failures, or software related, such as an application not responding to user requests. In addition, a failure can be a result of disasters such as a tornado or fire.

The goal in making your business infrastructure, data, and applications highly available is the minimization, even elimination, of these outages.

Now you may say this sounds well and good, but the iSeries servers are rock-solid and have proven over the years to be a reliable work-horse, and with a detailed backup and disaster recovery plan, why is high availability so important? It is true that iSeries servers are reliable (later in this document we detail the features that make it so reliable). However, simply put, high availability is important because it minimizes the risk of an outage and increases the availability of the system. A system outage can be costly in terms of lost productivity, lost revenue, and lost customers. In today's fiercely competitive environment, it takes years to gain a good solid reputation, and only seconds to tarnish it. With the increasing acceptance of e-business, companies can't afford to risk impacting customer service levels due to systems outages, both planned and unplanned. Customers want to do business, when it is convenient for them. Batch and backup windows are shrinking due to global business and the need for 24 X 7 X 365 access to the system. The demands of the customers and end users require round the clock system access. These are the factors driving the high availability phenomena.

Still not convinced? The level of availability can translate into being down hours or days. For example, a 90% level of availability means that the total downtime per year is approximately 36.5 days, while a 99.9% level of availability translates to approximately 8.76 hours of downtime per year. A 99.9% level of availability may sound good at first glance, unless you are in a business that can potentially lose up to \$13,000 a minute (as quoted in past press articles). This is a \$6,832,800 a year lost potential!

Objectives

It can be a daunting task to architect and to implement a highly-available solution which spans many different technologies. Moreover, heterogeneous server environments can add extra complexity due to differences that may exist across server platforms. Much documentation exists describing the “how to” for various individual components such as WebSphere Application Server, clustering, databases, etc. The sheer amount of this documentation can become overwhelming, and in many cases that information may not highlight the best choices for a particular server platform such as iSeries.

Our objectives in writing this document on the subject of iSeries high availability were the following:

- 1 To explain the strengths of the iSeries brand with regard to high-availability capabilities and characteristics
- 2 To provide technical information on proven techniques to maximize availability for each the components typically associated with e-business distributed application environments
- 3 To provide solid, practical reference material to assist those who are responsible for the planning and deployment of iSeries high-availability solutions
- 4 To supply specific examples of the actual configuration steps involved for each high-availability component within an overall, end-to-end high-availability topology

The effort to produce this document involved researching many manuals and Redbooks on the subject of high availability. From this research a few preferred topologies were selected, tested, and documented for iSeries. In addition, we included any information in the various IBM Redbooks and resources that we thought was pertinent to your understanding of high availability. However, we do not give all the various options researched or all the information available. Rather, we point the reader to the existing sources of information that one can utilize for additional investigation.

Despite all the information contained within this document, please bear in mind that the topic of high availability is not a trivial subject no matter what technologies, products, or computer platforms are used in the overall solution. Successful deployments require careful planning and coordination by skilled professionals who are well versed in the technical considerations for high availability. IBM consulting organizations and IBM business partners are available to assist you with any or all phases of a high-availability implementation for your particular computing environment.

Base Technologies Referenced in this Document

The major base technologies that are referenced in this document include the following:

- Version 5 Release 2 Modification 0 of the OS/400 operating system
- The IBM WebSphere Application Server Network Deployment V5.0 for iSeries
- HTTP Server for iSeries (powered by Apache)
- DB2 UDB for iSeries
- IBM Enterprise Storage Server

Document Organization

The list which follows is a high-level breakdown of the organization of this document:

- Chapters 2 through 3 focus on high availability concepts and principles.
- Chapter 4 gives details on what the iSeries server has to offer in the area of availability.
- Chapter 5 provides a good high-level view of the most common e-business programming architectures that are in use today.
- Chapters 6 to 13 are about high availability considerations for various components of an e-business environment, such as the load balancer, the HTTP server, etc.
- Chapter 14 covers disaster recovery considerations.
- Chapter 15 shows the recommended, highly available topologies for an e-business environment.
- Chapters 16 through 19 goes over the configuration steps for various e-business components.
- Chapter 20 covers testing a high availability environment.
- Chapter 21 shows how monitoring your e-business environment plays an important part in recovering from component failures.

At the end of most chapters, you will find a section entitled “Resources for Learning” that contain links to relevant supplemental information about various topics discussed in the chapter. The information resides on IBM and non-IBM Internet sites, whose sponsors control the technical accuracy of the information.

These links are provided for convenience. Often, the information is useful all or in part for understanding a topic discussed in the chapter. When possible, links are provided to technical papers and Redbooks that supplement the broad coverage of the release documentation with in-depth examinations of particular product areas.

HIGH AVAILABILITY CONCEPTS

Before you can plan for the availability of your iSeries server, it is important for you to understand some of the key concepts associated with this topic.

Availability Defined

In the previous chapter, we defined high availability as the minimization of planned and unplanned outages. We will elaborate on this definition and discuss how we measure availability.

Availability is a measure of the time that a server is functioning normally as well as a measure of the time the recovery process requires after the system fails. In other words, it is the downtime that defines system availability. This downtime includes both planned and unplanned downtime.

Let A be an index of system availability expressed as a percentage, $MTBF$ the mean time between failures, and $MTTR$ the maximum time to recover the system from failures; then we have:

$$A = \text{MTBF} / (\text{MTBF} + \text{MTTR})$$

As $MTBF$ gets larger, A increases and $MTTR$ has less impact on A . As $MTTR$ approaches zero, A increases toward 100 percent. This means that if we can recover from failures very quickly, we will have a highly available system. The time to recover a system includes fault detection time and system recovery time. By using fault detection mechanisms and the automatic fail-over of a service or component to a healthy host one can minimize the fault detection time and the service recovery time. The $MTTR$ is minimized because the fault detection time is minimized and no repair attempt is needed. Therefore, A is significantly raised. Any repairs to the failed node and any upgrades of software and hardware will not impact the service availability. This is the so-called *hot replacement* or rolling upgrade.

The availability issue is not as simple as the formula discussed above. First, $MTBF$ is just a trend. For example, if a CPU has an $MTBF$ of 500,000 hours, it does not mean that this CPU will fail after 57 years of use; this CPU can fail at any time. Second, there

are many components in an e-business environment, and every component has a different MTBF and MTTR. These variations make the availability of an e-business environment unpredictable using the formula above. We can build a simulation model to determine an e-business environment's availability with random process theory such as Markov chains, but this is beyond the scope of this book.

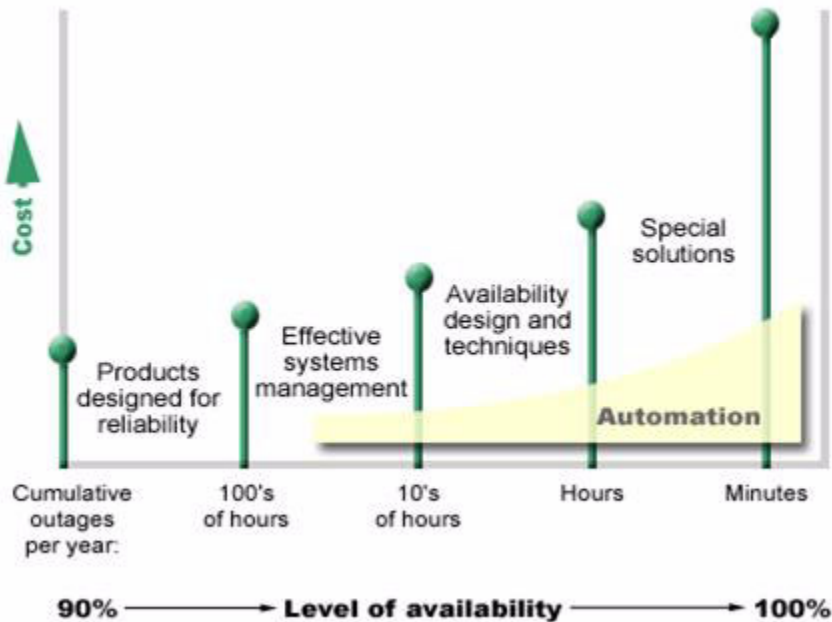
Availability becomes much more complicated in an e-business environment since many components are involved, such as firewall, Load Balancer (LB), Web server, WebSphere Application Server and database.

Our goal is to minimize the MTTR through various techniques; if MTTR=0, A=100% no matter what the MTBF is. Using this approach, availability of an e-business environment becomes predictable and manageable.

Levels Of Availability

First of all, availability is closely related to cost, as shown in Figure 2.1 on page 6. It is important to balance the downtime with cost. The more you invest, the less downtime there is. Therefore, it is also very important for you to evaluate what you will lose if your e-business is temporarily unavailable. Different businesses have different costs for downtime, and some businesses such as financial services may lose millions of dollars for each hour of downtime during business hours. Costs for the downtime include not only direct dollar losses but also reputation and customer relationships losses.

Figure 2.1 Availability techniques vs. cost



There are several levels of availability. These levels differ in the type and duration of outages that they tolerate. These levels are as follows:

Single system availability: All components are non redundant and run on a single system.

High availability: Minimal loss of service to meet the availability needs of a particular business, including handling of unplanned and planned outages. This usually involves the usage of redundant components and failover capabilities to ensure a more predictable and shorter recovery time for unplanned outages.

Continuous availability: 24 X 365 operations, no planned or unplanned outages that affect applications that are deemed critical to be accessible at all times. Fault tolerant hardware and customizations to mission-critical applications are required to help achieve the zero down time goal for continuous availability. Includes disaster recovery, which, implies maintaining systems in different sites. When the primary site becomes unavailable due to disasters, the backup site can become operational within a reasonable time.

NOTE The goal of continuous availability (0% down time), while laudable, is never achieved in the real world. High availability in the context of this document does not imply continuous availability, although by performing the various recommendations in this document you can get very close to continuous availability.

Whatever the availability level you desire, the iSeries platform has solutions available to meet your specific requirements.

Process (Application) and Data Availability

High availability can be further broken down into a discussion of process (application) and data availability.

Process availability simply means that redundant processes exist which are capable of servicing requests should another process fail. Process availability is usually achieved through operating system or middleware (e.g. WebSphere) clustering, depending upon the type of process.

Below is a list of some of the e-business middleware or component failures which must be considered when implementing an overall high-availability solution for an e-business environment:

- Network dispatcher failure
- HTTP server not responding
- Application server simply fails to respond
- WebSphere administrative server failure (Administration not possible)
- Load Balancer failure

Of course process availability is of little use if critical data is not available. Data availability for critical data residing on iSeries servers is assured using one of two schemes:

- Data replication technologies
- Switchable IASP (Independent ASP) usage involving OS/400 clustering

Careful planning must be done to minimize or eliminate the effects of unplanned outages. Just as important to consider are the planned outages such as application and operating system upgrades, PTF installations, and others.

Data Center vs. Disaster Center

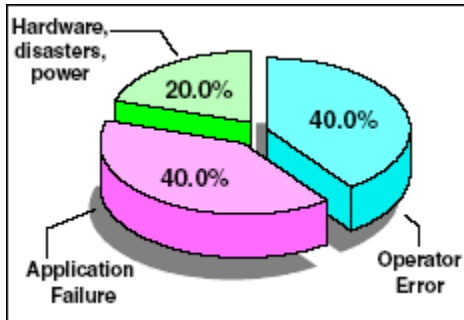
A *data center* solution is local. It typically addresses the day to day planned outage scenarios and the locally recoverable unplanned outage scenarios. Servers are in close proximity to each other. Normally one would deploy a data center solution to do back-ups that don't impact production or to recover from a locally recoverable outage such as a memory card failure.

However, making your data center highly available may not do you any good if for some reason a disaster occurs that disables the data center in such a way that makes communications with the data center impossible.

A *disaster center* solution is remote and requires remote systems to take over in the event of a site outage. The distance between systems is very important to insure no single catastrophic event affects both sites. However, the price for distance is loss of performance due to the latency time for the signal to travel the distance. The most common and simple solution is to vault back-up tapes at some distant facility. More well developed firms deploy a business partner solution whereby the second machine is kept real-time updated. In this scenario, many customers use that second server for supporting certain types of business transactions and also doing nightly back-ups so as to not impact the production environment.

Factors Impacting Availability

The IBM iSeries and AS/400 systems are renowned for their availability due to a number of factors:



Designed for availability: A single iSeries server delivers an average of 99.9+% availability. According to data collected by IBM between 1999 and 2001, AS/400 and iSeries owners have experienced an average of less than nine hours of unplanned down time per year. The diagram on the left indicates two out of three factors of unplanned outages that can be affected by proper design. IBM delivers a very reliable server because the IBM Development team

designs, creates, builds, tests, and services the iSeries and AS/400 systems as a single entity.

Effective system management process: As noted in figure to the right, 90 percent available translates to 36 days. Lack of attention to system management disciplines and processes affect the availability achieved. Availability solutions, such as clusters, are undermined when system management processes are lacking or nonexistent.

An effective system management strategy ties heavily into automation, such as an

<u>Availability Percentage</u>	<u>Total Outage Per Year</u>
99.9999	= 32 seconds
99.999	= 5 minutes
99.99	= 53 minutes
99.9	= 8.8 hours
99	= 87 hours (3.6 days)
90	= 876 hours (36 days)

automatic archival of data, continuous system auditing, responding to security exposures, and monitoring error logs, backup and restores, and so on.

An investment in system management disciplines, automation, and application recovery is necessary. Just a few additional hours of yearly downtime reduces availability from 99.99% availability to 99.9%.

Increased automation: Increased availability means a reduction in the possibility of errors and recovery delays, and an increase in consistency. Human errors can create more down time than hardware failures or power outages. More effective automation through the use of automation software and tools can help offset an overburdened staff and allow them to attend to more unique and critical decisions and tasks. As availability requirements increase, investments in automation must also increase.

Exploitation of availability techniques and applications designed for availability: Decrease unplanned outages and their effects by utilizing server availability options (for example, disk protection) and OS/400 functions, such as access path protection, file journaling, user auxiliary storage pools (ASPs) and independent ASPs.

Target a phased approach at increasing application resiliency (recoverability) and availability. As a general rule, an application on a non-clustered system is difficult to recover. A cluster solution cannot overcome a poor application design.

Use applications that incorporate commitment control or other transaction methods, and use application recovery to decrease application recovery times and database integrity issues (incomplete transactions). You must use journaling and application recovery techniques to achieve high availability at a minimum. More sophisticated and highly available applications also use commitment control. Each technique is a building block foundation to all highly available environments.

Implementation of special solutions to meet availability goals: To reach your availability goals, special solutions, such as iSeries or AS/400 clusters with monitoring, automatic switchover, and recovery automation, are implemented to control both planned and unplanned outages. If you sidestep the issues described above, even sophisticated options like clusters may not provide the highest possible availability levels. Small outages, such as recovering or reentering transactions, add up.

How Much Availability Do I Need?

No one would argue the importance of availability. However, when asked to justify the cost of additional hardware, software, or consulting services to support improved availability, many people do not know how to build a case. We recommend that you use the following steps when planning and evaluating your end-to-end e-business HA solutions:

- 1 Analyze your requirements:
 - a Do you need continuous operations? Most customers do not need continuous operations, therefore, upgrades of software/hardware can be performed offline.
 - b What kind of availability do you need during your hours of operation?
 - c What is the performance requirement during a failover? For example, the performance may be impacted because fewer nodes/servers serve client requests after some of the nodes/servers fail.

- 2 Analyze the cost factors. How much will you lose when your system is unavailable, and how much can you invest in your system availability? If downtime costs you more, you should invest more to improve your system availability.
- 3 Estimate the setup and administrative complexity. More complicated systems require skilled people and more configuration and administration effort.
- 4 Consider all the components in an e-business environment. Usually, the overall availability is dominated by the weakest component in the chain. Consider the possibility of failure in each component and its failover time.
- 5 Analyze failover time, which mainly includes fault-detection time and recovery time. For different failover mechanisms/techniques, the failover time is different.
- 6 Analyze the recovery point, where your processing resumes after a failover. It is directly related to the amount of work that is lost during a failover.
- 7 Understand the programming models. This concerns whether the failover is transparent to clients and whether one component's failover is transparent to other components in an e-business environment. Some services are able to perform failover transparently, since the failover is masked from the application. Others have options for adding application code that can retry when necessary.
- 8 Finally, know that there is usually more than one solution to address a given failure. Some solutions may have special restrictions. Analyze the trade-offs between the different solutions.

High Availability Principles

The basic principles of high availability (HA) are not overly complex. You must consider the possible failure of each component in the architecture of your solution. This will have an impact on your decisions all the way down from the architecture to the design, technology, topology, and products used to build the overall solution. Once the solution as a whole is defined to meet high availability requirements, you must identify dependencies on which you may, or may not have short-term influence. Your Internet Service Provider (ISP) is an example of this. If your connection to the Internet goes down, all you can do is to wait for it to come back. Dependence on a single ISP is a potential high availability bottleneck. Planning and designing a configuration using multiple ISPs for high availability requires a considerable amount of time to arrange. You must consider and plan for this well in advance.

The skills of your staff and availability of vendor support are crucial assets in supporting the high availability requirements of your configuration.

High Availability Rules for Runtime Environments

In this section, we will give you simple rules and guidelines when planning for high availability of your runtime environment and for verifying existing high availability configurations.

There are three basic rules:

- 1 Every entity must be redundant.
- 2 Each entity must be monitored for failure.

3 A failed entity must not receive any work.

There is a set of high availability terms that relate to the rules listed above in defining “the anatomy of a failure”. These terms are:

- Detect - continually monitor your configuration for component failures or problems
- Recover - use of redundant entities to continue processing during component failure
- Decide - make a decision to not send work to a failed entity and then decide to send work to that entity after repair
- Repair - fixing the problem component and bringing it back online

Redundancy

In order to make entities useful for high availability, they must be grouped together for redundancy. There are three major groups: high availability pair, cluster, and pool.

High availability pair

Some products have the capability to act in pairs, complying with all three high availability rules. There are two kinds of pairs. *Primary/Backup* and *Peer* pairs.

With the *Primary/Backup* type, one does the work (Primary role) while the other one monitors it (Backup or standby role). The Backup takes over in case of a detected failure. For example, iSeries OS/400 clustering achieves this at an operating system level.

In a peer high availability configuration, both members are actively working and may take over the work from its peer in case of a detected failure.

Cluster

Many products that are grouped into a cluster are not capable of monitoring their peers. Usually, they are not even aware of being grouped. They need a third party to distribute requests between them and to comply with high availability rules 2 and 3 listed above. Each member of a cluster must be able to serve identical requests. Members of a cluster are typically capable of serving multiple requests in parallel (multithreaded). So they can have different levels of load assigned to them. Adding members to a cluster and deleting them from the cluster is an administrative task that cannot be done on the distributor instance’s own initiative.

Pool

If the members of a group are threads, like servlet instances or JDBC connection instances, it is typically referred to as a *pool*. Like members of a cluster, they require a third party to distribute requests and to comply with high availability rules 2 and 3 listed above. In this example of a group, the distributing instance can create and delete threads in the pool to adjust to the overall load on the pool.

Request Distribution

To distribute requests to the members of a cluster or pool, there must be an entity that is aware of all members. This entity must address high availability rules 2 and 3. If a distributing entity is enhanced with the capability to measure the load on the cluster members and distribute the load equally among them, it is referred to as a *load balancer*.

There are two distinct configurations using load balancers, the cluster configuration and the high availability configuration.

Cluster configuration

In a cluster configuration, you have multiple entities that perform redirection or basic load balancing functions that are not aware of each other. Therefore, they are grouped in clusters themselves. To comply with high availability rules 2 and 3, these entities would need a third party.

The third party load balancer has the following role:

- Monitor members of a downstream cluster and detect a failure
- Balance requests between the downstream servers in a given cluster
- Given multiple requests from a unique client, optionally preserve an affinity to one of the cluster members

The advantage of a configuration like this is the separation of processing functions. The separation of processing functions allows the machine selection to be made to suit the appropriate processing request (for example Web serving as opposed to application serving).

Examples of technologies that would fit into this category are:

- Caching proxy node enhanced with a content-based routing component (for example, WebSphere Edge Server CBR component)
- Web server redirector node (for example, WebSphere Application Server Plug-in)
- Java Virtual Machine (servlet pool, JDBC connection pool, etc.)
- Stand-alone Load Balancer node (for example, WebSphere Edge Server Dispatcher)

High availability configuration

In a high availability configuration, the load balancers act in pairs, capable of complying with all three high availability rules. Each load balancer has the following role:

- Capability to monitor its peer load balancer and detect a failure
- Capability to take over the balancing task from its peer load balancer
- Capability to monitor members of a downstream load balancer cluster and detect a failure
- Capability to balance requests between the downstream servers in a given load balancer cluster

An example of this would be the WebSphere Edge Server Load Balancer in high availability mode.

Resources for Learning

Use the following links to find relevant supplemental information about various topics discussed in this chapter. The information resides on IBM and non-IBM Internet sites, whose sponsors control the technical accuracy of the information.

These links are provided for convenience. Often, the information is useful all or in part for understanding a topic discussed in this chapter. When possible, links are provided to technical papers and Redbooks that supplement the broad coverage of the release documentation with in-depth examinations of particular product areas.

High availability

- *IBM WebSphere V5.0 Performance, Scalability, and High Availability: WebSphere Handbook Series* (July 2003)

<http://www.redbooks.ibm.com/redbooks/pdfs/sg246198.pdf>

This IBM Redbook discusses various options for scaling applications based on IBM WebSphere Application Server Network Deployment V5.0. It also discusses high availability. A must-read. A lot of the information in this document was obtained from this Redbook.

- *Patterns for the Edge of Network* (November 2002)

<http://www.redbooks.ibm.com/redbooks/pdfs/sg246822.pdf>

This IBM Redbook describes guidelines and options for the selection of Runtime patterns that include high availability and high performance considerations in the design process.

- Availability

http://publib.boulder.ibm.com/series/v5r2/ic2924/index.htm?info/rzalw/rzalw_avail_overview.htm

This iSeries Information Center topic contains information about how to decide what level of availability you need and technologies and techniques you can use to achieve your availability goals.

IBM offerings and services

- iSeries for Capacity BackUp Offering

<http://www.ibm.com/servers/eserver/series/hardware/is4cbu/>

The iSeries for Capacity BackUp offering is intended for companies requiring an off-site, disaster recovery machine at an extremely affordable price. Using iSeries On/Off Capacity on Demand capabilities, the iSeries for Capacity BackUp offering has a minimum set of startup processors that can be used for any workload and a large number of standby processors that can be used at no-charge in the event of a disaster.

Standby processors cannot be permanently activated. In addition to their no-charge disaster usage, standby processors can be used on a chargeable basis for production role swapping during an unscheduled outage, tape backup, failover testing, and role swapping during upgrades or PTF installations.

The iSeries for Capacity BackUp offering is not intended or priced as a backup server for 24x7 high availability solutions that require day-to-day full operation of the backup server. High usage of standby processors for purposes other than a disaster may add significant cost to an I.T. operation.

- iSeries for High Availability Offering

<http://www.ibm.com/servers/eserver/series/hardware/is4ha/>

The iSeries for High Availability offering is intended to help eliminate planned downtime while also helping to reduce unplanned outages. This offering is valuable for customers who require 24x7 availability. To achieve the highest possible availability, multiple iSeries are tied together with high function, high availability software. Role swapping and running production work on both primary and secondary servers is typical. In this multi-iSeries environment, an iSeries for High Availability server would typically be linked to an iSeries server Enterprise Edition of equal or larger size (processor performance - CPW). An iSeries for High Availability offering is aggressively priced compared to an Enterprise Edition; however, a Standard Edition carries an even lower price than the iSeries for High Availability offering.

- IBM Business Continuity and Recovery Services

<http://www.ibm.com/services/continuity/recover1.nsf/documents/home>

IBM Business Continuity and Recovery Services can help you map your current infrastructure and design a customized recovery plan for a variety of disaster scenarios. IBM professionals also have the knowledge and experience to help you recover from an actual disaster situation.

DEVELOPING AN AVAILABILITY PLAN

For maximum availability, it is very important that planning is performed for the operating system, hardware, and database, as well as for applications and operational processes. The need for higher availability can be relaxed by determining what a business actually needs and what is possible using the available technology. In reality, most applications can withstand some planned outage, either for batch work, backups, and reorganization of files, or to affect application changes. In most cases, the application is expected to be available at the host, or perhaps on the network that is owned and controlled by the organization. However, continuous availability in the environment outside the control of the organization (that is, on the Internet) is not normally included in the business case.

One Size Does Not Fit All

Each organization has its own unique signature. In the process of planning for the unexpected, *one size definitely does not fit all*. Each organization has unique business processes necessary for its survival. Each has a unique culture and must face different regulatory and environmental limitations. Each must confront unique risks that it must avoid, transfer, or mitigate. Each has informational assets that it must protect and insure, and these assets differ from organization to organization in terms of dollar value, quantity, format, and volatility. Additionally, applications and their data have higher or lower availability requirements, such that some applications may require a higher tier of availability/recovery considerations than others which are less critical to that organization.

Shapers of Recovery/Availability Strategies

Since each organization is unique and has, therefore, unique requirements for its data recoverability/availability strategy, let us quickly examine some of the forces that are currently impacting the recovery strategies that organizations must implement:

- The exponential growth in the amount of data that must be stored has created the need for a comprehensive program to manage that growth. Data, like any other

organizational asset (such as financial, physical, personnel) must be managed with the same due diligence. The storage management discipline is concerned with managing application requirements, forecasting future growth needs (capacity planning), performance and tuning, and the recovery and availability of data.

- The rate of technological change enormously complicates the recovery planning process. This accelerated rate of technological innovation requires companies to invest heavily in information-based solutions to business processes or run the risk of becoming uncompetitive and losing market share. It seems that almost daily, advances in the networking arena require organizations to address connectivity issues, Virtual Private Network deployments, faster, more efficient switching technologies, and demand for larger and larger bandwidths. Rapid change and adjustments further complicate staying online or being able to come back online.
- Closely related to the above point, the mixture of disparate platforms, operating systems, and communication protocols found within most organizations intensifies the already complex task of recovering/preserving data. Gone forever are the days of the water-cooled monolith. Today, management must have reliable methods of recovering not only the iSeries server, but perhaps multiple flavors of UNIX, Windows, not to mention the network infrastructure. It is not surprising that server/storage consolidation has come to the forefront in many organizations' strategic IT planning, not only to more effectively manage day-to-day operations, but to also ease the difficulties of recovery planning.
- A number of constituencies (customers, suppliers, management, and others) expect (in other words, demand) that data be available at anytime from anywhere. Web-based applications, for many, have to be available without exception. Revenue and the preservation of corporate image require it. The explosive growth of the Internet has drastically altered the traditional mechanisms of the marketplace, and have created the need for Web site redirection and load-balancing as availability assurance methods.
- The concept of what is a threat has also changed and now must include: terrorism; vandalism from outside as well as from within the organization; industrial espionage; compromised privacy; advanced viruses which can corrupt and modify mission critical data; and fraud. This list is, unfortunately, not all-inclusive, and each can threaten your enterprise's survivability.
- New regulatory requirements have forced many to rethink their approach to data survivability. The Health Insurance Portability and Accountability Act (HIPAA) is an example of a requirement that determines how an entire industry, the US health care industry, must handle and account for patient-related data.
- Finally, globalization of a firm creates new opportunities for some firms, but it can also create additional recovery and availability concerns that did not exist in the centralized computing model.

Types of Contingency Plans

IT Contingency Planning represents a broad scope of activities designed to sustain and recover critical IT services following an emergency. IT Contingency Planning fits into a much broader emergency preparedness environment that includes Organizational and Business Process Continuity And Recovery Planning. Ultimately, an organization would use a suite of plans to properly prepare response, recovery, and continuity activities for disruptions affecting the organization's IT systems, business processes, and

the facility. Because there is an inherent relationship between an IT system and the business process it supports, there should be coordination between each plan during development and updates to ensure that recovery strategies and supporting resources neither negate each other nor duplicate efforts.

In general, universally accepted definitions for Contingency Planning and these related planning areas have not been available. Occasionally, this has led to confusion regarding the actual scope and purpose of various types of plans. To provide a common basis of understanding regarding IT Contingency Planning, this section identifies several other types of plans and describes their purpose and scope relative to IT Contingency Planning. Because of the lack of standard definitions for these types of plans, in some cases the scope of actual plans developed by organizations may vary from the descriptions below.

Business Continuity Planning (BCP)

Business Continuity Planning confronts the likelihood of a disaster, how the disaster interrupts the business process, and how the business can continue in operation. An interruption could be something related to a winter storm, the loss of electricity to the general area, or the complete inaccessibility of a facility for an extended period of time. The cause of the interruption does not matter; what matters is gaining management control and processing capacity just after the interruption.

Data integrity is one of our major concerns, and the type of disaster that has the most impact is the rolling disaster. Here the disaster unfolds along a significant amount of time (as in fire or water flood). In such an event, even using techniques such as remote copy by itself may not preserve data integrity, as the hardware components do not stop at the same time.

Business Continuity Planning relates to considerations that effect the whole of the business process. If a building is unusable where will workers be relocated? How will their personal business tools be replaced? How will these workers access the system by the network?

The BCP focuses on sustaining an organization's business functions during and after a disruption. An example of a business function may be an organization's payroll process or consumer information process. A BCP may be written for a specific business process or may address all key business processes. Information systems are considered in the BCP only in terms of their support to the larger business processes. In some cases, the BCP may not address long-term recovery of processes and return to normal operations, but only cover interim business continuity requirements.

An extreme example of a piece of a BCP is to keep a paper copy of the payroll documents of the previous month; then, when everything fails in the event of a disaster, a photocopy may address the business continuity by repeating the same salaries from the previous month.

Business Recovery Plan (BRP)

The BRP (also Business Resumption Plan) addresses the restoration of business processes after an emergency. The BRP is similar to the BCP, but unlike that plan, the BRP typically lacks procedures to ensure continuity of critical processes throughout an emergency or disruption.

Continuity of Operations Plan (COOP)

The COOP focuses on restoring an organization's (usually the headquarters element) essential functions at an alternate site and performing those functions for up to 30 days before returning to normal operations.

Incident Response Plan (IRP)

The Incident Response Plan establishes procedures to address cyber attacks against an organization's IT server and workstation systems. These procedures are designed to enable security personnel to identify, mitigate, and recover from malicious computer incidents, such as unauthorized access to a system or data, denial of service, or unauthorized changes to system hardware or software (such as, malicious logic such as a virus, worm, or Trojan horse).

Occupant Emergency Plan (OEP)

The OEP provides the Response Procedures for occupants of a facility in the event of a situation posing a potential threat to the health and safety of personnel, the environment, or property. Such events would include a fire, hurricane, criminal attack, or a medical emergency. OEPs are developed at the facility level, specific to the geographic location and structural design of the building.

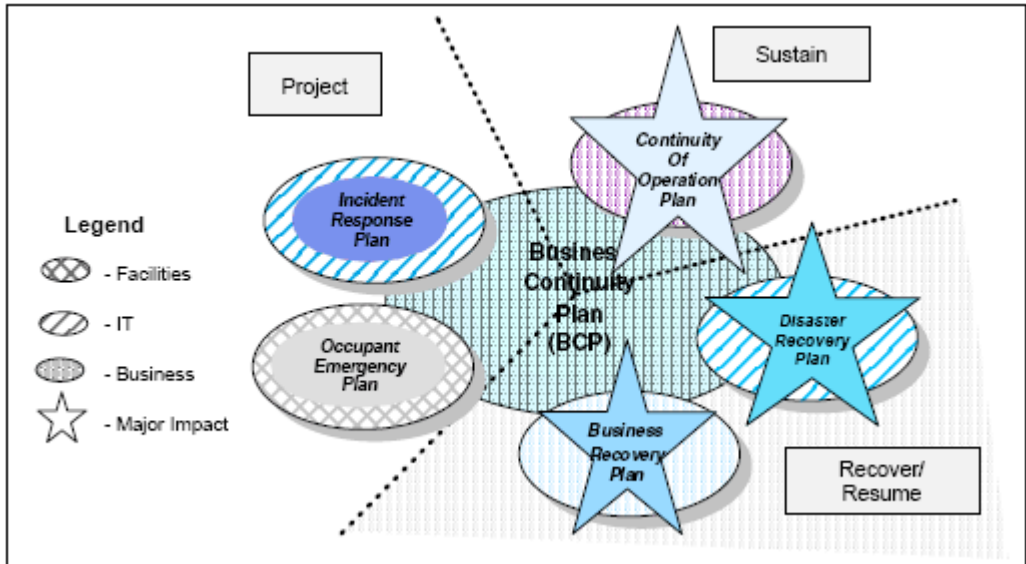
Disaster Recovery Plan (DRP)

As suggested by its name, the Disaster Recovery Plan applies to major events that deny access to the normal facility for an extended period. Frequently, the Disaster Recovery Plan refers to an IT-focused plan designed to restore operability of the target system, application, data, or computer facility at an alternate site after an emergency.

The Disaster Recovery Plan scope may overlap that of an IT Contingency Plan; however, the Disaster Recovery Plan is narrower in scope and does not address minor disruptions that do not require relocation. Nevertheless, logical data errors such as viruses, have no need for relocation, but can also be seen as a big disaster for a customer. These cases are covered in the Incident Response Plan and are very much linked to the topics discussed in the following chapters.

The Figure 3-1 on page 18 depicts how the various plans relate to each other.

Figure 3.1 Interrelationship of Emergency Preparedness Plans



Disaster Recovery Planning

A Disaster Recovery Plan is a comprehensive statement of consistent actions to be taken *before, during, and after* a disaster. The planning should be documented and tested to ensure the continuity of operations and availability of critical resources in the event of a disaster.

The Disaster Recovery Plan applies to all, usually catastrophic (or not), events that deny access to the normal facility for an extended period, as the orderly shutdown of a facility due to a one-day long fixing of the energy cables.

Depending on the time span or the severity of the interruption, significant consequences or the very survivability of the corporation may depend on management's ability to reestablish critical business functions. Usually these business functions have required years to create and establish, but management must reestablish these functions within hours or days.

This is a difficult problem, and reestablishing the complex business environment in a timely manner requires a well thought-out plan in place ready to be executed. This plan also predicts actions to be executed after the reestablishment of the damaged site, in order to return the original situation.

The Disaster Recovery Plan involves more than off-site storage or backup processing actions.

Organizations should also develop written, comprehensive items that address all the critical operations and functions of the business. The plan should include documented and tested procedures, which, if followed, ensure the ongoing availability of critical resources and continuity of operations.

The probability of a disaster occurring in an organization is highly uncertain. As a customer you want to know when a disaster occurs, precisely how the system and how your staff will react to this disaster. Predictability of the reaction to a disaster is the goal. This can only be accomplished by having a combination of automation functions and well documented and regularly tested procedures. In other words, you do not want to wait until a disaster occurs to find out whether or not your plan will work.

A Disaster Recovery Plan, however, is similar to liability insurance: It provides a certain level of confidence in knowing that if a major catastrophe occurs, it will not result in financial disaster. However, just having an insurance policy by itself is not enough because it may not be a compensation for the incalculable loss of business during the interruption or for the business that goes bankrupt because of the disaster. On the other hand, to have an understandable Disaster Recovery Plan may help reduce fees when insuring an enterprise.

It is interesting to note that in a disaster situation, customers, providers, and employees normally are aware that an outage has happened to the central computer facility. In this situation these people are not so demanding, and in many cases they can wait for the duration of the outage. Clearly, the outage interval is mainly dependent on the recovery solution.

Usually this duration is measured in two different non-overlapping components:

- Service loss

This represents the loss of computing time you experience from the moment of disaster up to the moment when your system has been recovered.

- Data loss

This represents the loss of data you have, that is, how much time it takes to re-execute the work once your system is recovered to reach the same level of data as before the disaster.

Organizations that have the most revenue and are the most heavily dependent on online systems have the highest potential loss of revenue from (global) application and network outages. Depending on the industry the revenue per hours lost by a disaster is different. It is not a surprise that energy, telecommunications, manufacturing industries and financial institutions have the highest revenues per hour.

The percentage of the revenue actually lost depends on the criticality of the system that experiences the outage (such as, degree of customer interaction, existing work arounds, peak periods) and the number of users affected by the outage or slowdown. Also, significant immediate losses can result in bad publicity and loss of customer trust that affects future revenues.

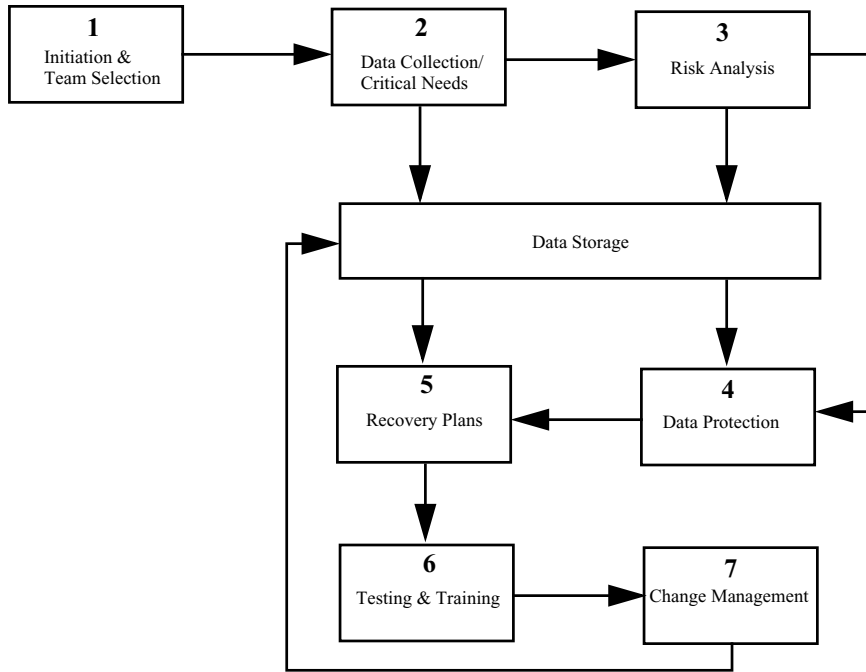
Preparedness is the key. The planning process should minimize the disruption of operations and ensure some level of organizational stability and an orderly recovery after a disaster.

Disaster Recovery Planning Project Phases

Disaster Recovery planning is typically undertaken as a multi phase project. Figure 3-2 is a diagram that illustrates the phases of a typical Disaster Recovery planning project.

The central *data storage* presented receives the information from the Collection Phase (and from subsequent testing and change management activity) creating centralized data storage that can be used in the other phases of planning.

Figure 3.2 Diagram of disaster recovery planning phases



A description of each phase follows.

Project Initiation and Team Selection Phase

Following are the major aspects of the project initiation and team selection phase.

Top management commitment

Top management must support and be involved in developing, coordinating, and ensuring the Disaster Recovery effectiveness within the organization.

Adequate time and resources must be committed to the development of an effective plan. Resources could include both financial considerations and the effort of all personnel involved.

The Establishment of a planning committee

A planning committee should be appointed to oversee the development and implementation of the plan. The planning committee should include representatives from all functional areas of the organization. Key committee members should include the operations manager and the data processing manager. The committee also should define the scope of the plan. Top Management should be informed regularly, because this is a very sensitive subject and because the expenses (in money and people) can be quite high.

Scope

Although most Disaster Recovery Plans address only data processing related activities, a comprehensive plan will also include areas of operation outside data processing. The plan should have a broad scope if it is to effectively address the many disaster scenarios that could affect the organization.

Assumptions

A worst case scenario should be the basis for developing the plan. The worst case scenario is the destruction of the main or primary facility. Because the plan is written based on this premise, less critical situations can be handled by using only the needed portions of the plan, with minor (if any) alterations required.

Every Disaster Recovery Plan has a foundation of assumptions on which the plan is based.

The assumptions limit the circumstances that the plan addresses. The limits define the magnitude of the disaster the organization is preparing to address. The assumptions can often be identified by asking the following questions:

- What equipment/facilities have been destroyed?
- What is the timing of the disruption?
- What records, files, and materials were protected from destruction?
- What resources are available following the disaster:
 - Staffing?
 - Equipment?
 - Communications?
 - Transportation?
 - Hot site/alternate site?

Following is a list of typical planning assumptions to be considered when writing the Disaster Recovery Plan:

- The main facility of the organization has been destroyed.
- Staff is available to perform critical functions defined within the plan.
- Staff can be notified and can report to the backup site(s) to perform critical processing, recovery, and reconstruction activities.
- The Disaster Recovery Plan is current.
- Subsets of the overall plan can be used to recover from minor interruptions.
- An alternate facility is available.
- An adequate supply of critical forms and supplies are stored off-site, either at an alternate facility or off-site storage.
- A backup site is available for processing the organization's work.
- The necessary long-distance and local communications lines are available to the organization.
- Surface transportation in the local area is possible.
- Vendors will perform according to their general commitments to support the organization in a disaster.

This list of assumptions is not all-inclusive, but it is intended as a thought-provoking process in the beginning stage of planning. The assumptions themselves will often dictate the makeup of the plan, therefore, management should carefully review them for appropriateness.

Teams and responsibilities

The Disaster Recovery planning should be structured using a team approach. Specific responsibilities should be assigned to the appropriate team for each functional area of the company. There should be teams responsible for administrative functions, facilities, logistics, user support, computer backup, restoration, and other important areas in the organization.

The structure of the contingency organization may not be the same as the existing organization chart. The contingency organization is usually structured in teams, that are responsible, such as, for major functional areas. The teams may include:

- Management team
- Business recovery team
- Departmental recovery team
- Computer recovery team
- Damage assessment team
- Security team
- Facilities support team
- Administrative support team
- Logistics support team
- User support team
- Computer backup team
- Off-site storage team
- Software team
- Communications team
- Applications team
- Human relations team
- Marketing/customer relations team

It is not necessary for the organization to have all these teams created exactly in that area, but we strongly recommend that functions related with each one of these organizational teams be executed.

Personnel should be chosen to staff these teams based on their skills and leadership. Ideally, teams would be staffed with the personnel responsible for the same or similar operation under normal conditions. For example, Server Recovery Team members should include the server administrators. Team members must understand not only the contingency plan purpose, but also the procedures necessary for executing the recovery strategy. Teams should be sufficient in size to remain viable even if some members are unavailable to respond (such as, due to vacations), or spare team members may be designated. Similarly, team members should be familiar with the goals and procedures of other teams to facilitate inter-team coordination.

Each team is led by a leader who directs overall team operations and acts as the team's representative to management and liaisons with other team leaders. The team leader disseminates information to team members and approves any decisions that must be made within the team. Team leaders should have a designated substitute to act as the leader if the primary leader is unavailable.

The most important team in Disaster Recovery planning is the Management Team, which is necessary for providing overall guidance following a major system disruption or emergency. The Management Team is typically led by a senior management official, such as the Chief Information Officer (CIO), who has the authority to make decisions regarding spending levels, acceptable risk, and intercompany coordination.

Following are the major functions of such a team:

- Responsible for activating the Contingency Plan and supervising the execution of contingency operations.
- Facilitates communications among other teams and supervises plan tests and exercises.
- All or some of its personnel may also lead specialized contingency teams.
- Coordinates the recovery process.
- Assesses the disaster, activates the Recovery Plan, and contacts team managers.
- Oversees, documents, and monitors the recovery process.
- Are the final decision-makers in setting priorities, policies, and procedures.

Another important team is the Disaster Recovery Management Team, who is called into action under the authority of the Top Administrative Committee.

Data Collection and Critical Needs Phase

To determine the critical needs of the organization, each department should document all the functions performed within that department.

An analysis over a period of two weeks to one month can indicate the principle functions performed inside and outside the department, and assist in identifying the necessary data requirements for the department to conduct its daily operations satisfactorily.

Some of the diagnostic questions that can be asked include:

- If a disaster occurred, how long could the department function without the existing equipment and departmental organization?
- What are the high-priority tasks, including critical manual functions and processes in the department? How often are these tasks performed, such as, daily, weekly, monthly, etc.?
- What staffing, equipment, forms, and supplies would be necessary to perform the high priority tasks?
- How would the critical equipment, forms and supplies be replaced in a disaster situation?
- Does any of the above information require long lead times for replacement?
- What reference manuals and operating procedure manuals, are used in the department?
- How would these be replaced in the event of a disaster?

- Should any forms, supplies, equipment, procedure manuals or reference manuals from the department be stored in an off-site location?
- Identify the storage and security of original documents. How would this information be replaced in the event of a disaster? Should any of this information be in a more protected location?
- What are the current computer backup procedures? Have the backups been restored?
- Should any critical backup copies be stored off-site?
- What would the temporary operating procedures be in the event of a disaster?
- How would other departments be affected by an interruption in the department?
- What effect would a disaster at the main computer have on the department?
- What outside services/vendors are relied on for normal operation?
- Would a disaster in the department jeopardize any legal requirements for reporting?
- Are job descriptions available and current for the department?
- Are department personnel cross-trained?
- Who would be responsible for maintaining the department's Contingency Plan?
- Are there other concerns related to planning for Disaster Recovery?

On top of that, there are some recommended data gathering materials and documentation to be included: backup position listing, critical telephone numbers, communications inventory, distribution register, documentation inventory, equipment inventory, forms inventory, insurance policy inventory, main computer hardware inventory, master call list, master vendor list, microcomputer hardware and software inventory, notification checklist, office supply inventory, off-site storage location inventory, software and data files backup/retention schedules, telephone inventory, temporary location specifications, other materials, and documentation.

IMPORTANT Each business unit should declare the maximum time of recovery of their application. There are two pieces of information that should be delivered: The *Recovery Time Objective* (RTO), how long should it take to recover to the point for an end user to be online again; and the *Recovery Point Objective* (RPO), how much data the user can afford to recreate after the disaster. Usually department personnel are biased in setting higher than needed figures for recovery. The other key piece of information is the *Network Recovery Objective* (NRO). This is the time to recover the network and move the users to the disaster site. There is no point in recovering the applications in 10 minutes if it takes several hours to connect the users to the new site. It is extremely helpful to develop pre-formatted forms to facilitate the data gathering process.

What the critical needs are can be obtained in a consistent manner by using a User Department Questionnaire. The questionnaire focuses on documenting critical activities in each department and identifying related minimum requirements for staff, equipment, forms, supplies, documentation, facilities, and other resources.

Setting priorities on processing and operations

Once the critical needs have been documented, management can set priorities within departments for the overall recovery of the organization. Activities of each department could be given priorities in the following manner:

- Essential activities (vital): A disruption in service exceeding one day would seriously jeopardize the operation of the organization.

- Recommended activities (critical): A disruption of service exceeding one week would jeopardize seriously the operation of the organization.
- Nonessential activities (non-critical): This information would be convenient to have but would not detract seriously from the operating capabilities if it were missing.

Although the classification for RTO and RPO will vary from company to company, some general guidelines can be shown to help qualifying the application:

- Vital: RTO < 8 hours after the outage occurred and RPO within 15 minutes of the time of outage
- Critical: RTO < 72 hours after the outage occurred and RPO start of the day of the outage
- Non-critical: RTO < 168 hours (1 week) after the outage occurred and RPO within 48 hours of the time of the outage

Risk Analysis Phase

The planning teams should prepare a risk management process and at the same time the Business Impact Analysis (BIA) that includes a range of possible disasters, including natural, technical, and human threats.

Each functional area of the organization should be analyzed to determine the potential consequence and impact associated with several disaster scenarios. The Risk Analysis Phase should also evaluate the safety of critical documents and vital records. IT systems are vulnerable to a variety of disruptions, ranging from mild (such as, short-term power outage, disk drive failure) to severe (such as, equipment destruction, fire). Many vulnerabilities may be eliminated through technical, management, or operational solutions as part of the organization's risk management or security controls; however, typically it is impossible to completely eliminate all risks. Disaster Recovery planning in this phase is designed to complement these risk management and security activities by focusing recovery solutions on addressed and residual risks. As a result, the Disaster Recovery Plan can provide a cost-effective means to ensure that essential IT services can be recovered quickly after an emergency.

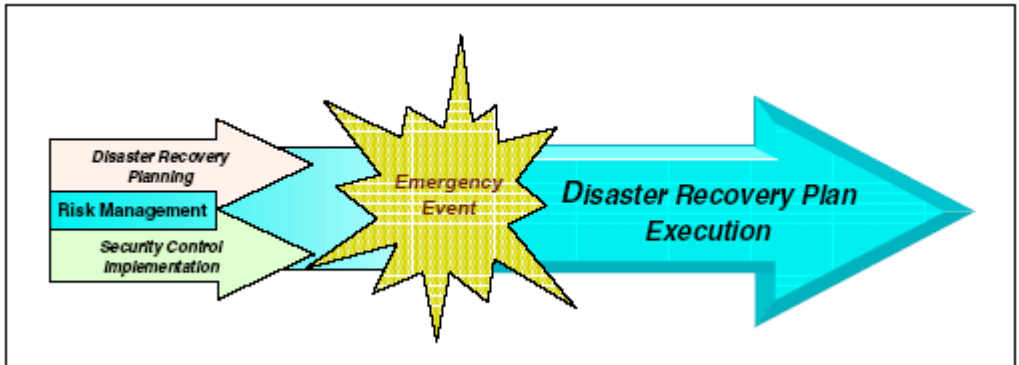
Risk management process

Risk management encompasses a broad range of activities to identify, control, and mitigate risks to an IT system. Risk management activities may be considered to have two primary functions:

- Prevent or reduce the likelihood of damaging incidents by reducing or eliminating risks. These preventive measures to reduce or eliminate risk typically form the security controls that protect a system against natural, human, and technological threats.
- Encompass actions to reduce or limit the consequences of threats in the event that they successfully disrupt a system. These measures are developed in anticipation of a possible event, are executed after that event has occurred, and form the basis for the Disaster Recovery Plan.

Figure 3.3 on page 27 illustrates the relationship between preemptive security controls and post-event Disaster Recovery Plan implementation.

Figure 3.3 Disaster Recovery Plan as an element of risk management implementation



Risks result from a variety of factors, although typically they are classified into three types:

- Natural: Hurricane, tornado, flood, and fire
- Human: Operator error, sabotage, and malicious code
- Technological: Equipment failure, software error, telecommunications network outage, and electric power failure

Not all risks are present with respect to a given IT department (or site). For example, depending on its location, a system may have no risk of damage by hurricane, but a reasonably high risk of effects from a tornado. To determine effectively the specific risks to a system, a risk assessment is required.

The risk assessment is critical because it enables the person responsible for Disaster Recovery to focus risk management efforts and resources in a prioritized manner only on identified risks. It is important to assess the impacts and consequences resulting from loss of information and services.

A thorough risk assessment should identify all potential risks and attempt to determine the probability of the risk actually occurring.

Traditionally, fire has posed the greatest threat to an organization. Intentional human destruction, however, should also be considered. The plan should provide for the worst case situation; such as, destruction of the main building. Usually the list of potential risks can be filtered to a list of identified risks including the ones with a measurable probability.

Items in determining the probability of a specific disaster should include, but not be limited to:

- Geographic location
- Topography of the area
- Proximity to power sources, water bodies, and airports
- Degree of accessibility to the organization history of local utility companies in providing uninterrupted services
- History of the area's susceptibility to natural threats

- Proximity to major highways that transport hazardous waste and combustible products
- Proximity to nuclear power plants
- Other factors, such as political instability

Ideally, all identified risks would be eliminated completely. However, rarely is this possible or cost-effective. Rather, an attempt is made to reduce risks to an acceptable level and remain aware of residual risks. Because these residual risks represent the complete set of situations that could affect system performance, availability, integrity, and security, the scope of the Contingency Plan may be reduced to address only these residual risks.

As a result, the Contingency Plan can be more narrowly focused, conserving company resources while ensuring an effective system recovery capability.

Because risks can vary over time and new risks may replace old ones as a system evolves, the risk management process must be ongoing and dynamic.

The Disaster Recovery Plan should also analyze the costs related to minimizing the potential exposures.

Business Impact Analysis (BIA)

The BIA is a key step in the Disaster Recovery planning process. It belongs to the Risk Analysis Phase. The BIA enables the Disaster Recovery group to characterize fully the system requirements, processes, and interdependencies, and use this information to determine contingency requirements and priorities.

The BIA process helps the Disaster Recovery group streamline and focus their plan development activities to achieve a more effective plan:

1 Identify critical resources.

The first BIA step evaluates the IT system to determine the critical functions performed by the system and to identify the specific system resources (hardware, network, software, data) required to perform them.

2 Identify disruption impacts and allowable outage times.

In this step you should analyze the critical resources identified in the previous step and determine the impact(s) on IT operations if a given resource were disrupted or damaged. The analysis should evaluate the impact of the outage in two ways: Over time and across related resources and dependent systems.

3 Develop recovery priorities.

The outage impact(s) and allowable outage times, characterized in the previous step, allow recovery strategies to be prioritized and developed to be implemented during Disaster Recovery Plan activation. For example, if the outage impacts step determines that the system must be recovered within four hours, the Disaster Recovery Plan will adopt measures to meet that need.

Creating an avoidance system

Because a goal of Business Recovery Planning is to ensure the safety of personnel and assets during and following a disaster, a critical aspect of the risk analysis process is to identify the preparedness and preventive measures in place at any point-in-time. Once the potential areas of high exposure to the organization are identified, additional preventative measures can be considered for implementation.

Disaster prevention and preparedness begins at the top of an organization. The attitude of senior management toward security and prevention should permeate the entire organization. Therefore, management's support of disaster planning can focus attention on good security and prevention techniques and better prepare the organization for the unwelcome and unwanted.

Disaster prevention techniques include two categories: Procedural prevention and physical prevention.

Procedural prevention

Procedural prevention relates to activities performed on a day-to-day, month-to-month, or annual basis, relating to security and recovery. Procedural prevention begins with assigning responsibility for overall security of the organization to an individual with adequate competence and authority to meet the challenges. The objective of procedural prevention is to define activities necessary to prevent various types of disasters and ensure that these activities are performed regularly.

Physical prevention

Preparedness for disaster begins when a site is constructed. It includes special requirements for building construction, as well as fire protection for various equipment components. Special considerations include: the computer area, fire detection and extinguishing systems, record(s) protection, air conditioning, heating and ventilation, electrical supply and UPS systems, dual power substation feeds, emergency procedures, vault storage area(s), and archival systems.

Data Protection Phase

Data Protection refers to all the measures adopted within an organization to safeguard assets, ensure the accuracy and reliability of records, and encourage operational efficiency and adherence to prescribed procedures. The system of internal controls also includes the measures adopted to safeguard the computer system and to determine which data needs which level of protection.

The nature of internal controls is such that certain Control Procedures are necessary for a proper execution of other Control Procedures. This interdependence of Control Procedures may be significant because certain control objectives that appear to have been achieved may, in fact, not have been achieved because of weaknesses in other Control Procedures upon which they depend.

Concern over this interdependence of Control Procedures may be greater with a computerized system than with a manual system because computer operations often have a greater concentration of functions, and certain Manual Control Procedures may depend on Automated Control Procedures, even though that dependence is not readily apparent. Adequate computer internal controls are a vital aspect of an automated system.

Security is an increasing concern because computer systems are increasingly complex. Particular security concerns result from the proliferation of PCs, local area networking, and online systems that allow more access to the mainframe and departmental computers. Modern technology provides computer thieves with powerful new electronic safe cracking tools.

Computer internal controls are especially important because computer processing can circumvent traditional security and control techniques. Important areas of concern related to general computer internal controls include: organization controls, systems development and maintenance controls, documentation controls, access controls, data and procedural controls, physical security, password security systems, and communications security.

Insurance coverage

Adequate insurance coverage is a key consideration when developing a Business Recovery Plan and performing a risk analysis. Having a Disaster Plan and testing it regularly may not, in itself, lower insurance rates in all circumstances. However, a good plan can reduce risks and address many concerns of the underwriter, in addition to affecting the cost or availability of the insurance.

Most insurance companies specializing in business interruption coverage can provide the organization with an estimate of anticipated business interruption costs. Many organizations that have experienced a disaster indicate that their costs were significantly higher than expected in sustaining temporary operations during recovery.

Most business interruption coverages include lost revenues following a disaster. Extra expense coverage includes all additional expenses until normal operations can be resumed. However, coverages differ in the definition of resumption of services. As a part of the risk analysis, these coverages should be discussed in detail with the insurer to determine their adequacy.

To provide adequate proof of loss to an insurance company, the organization may need to contract with a public adjuster who may charge between three and ten percent of recovered assets for the adjustment fee. Asset records become extremely important as the adjustment process takes place.

Types of insurance coverages to be considered include: computer hardware replacement, extra expense coverage, business interruption coverage, valuable paper and records coverage, errors and omissions coverage, fidelity coverage, and media transportation coverage.

With estimates of the costs of these coverages, management can make reasonable decisions on the type and amount of insurance to carry. These estimates also allow management to determine to what extent the organization should self-insure against certain losses.

Records classification

We classified the activities in “Setting priorities on processing and operations” on page 25, and, consequently, its' records, as one of the major critical resources identified in the “Business Impact Analysis (BIA)” on page 28, will have the same level of classification. Vital records are irreplaceable. Critical records can be obtained or reproduced at considerable expense and only after considerable delay. Non-critical records would cause inconvenience if lost, but can be replaced without considerable expense.

Vital and critical records should be duplicated and stored in an area protected from fire or its effects.

Records kept in the computer room should be minimized and should be stored in closed metal files or cabinets. Records stored outside the computer room should be in fire-resistant file cabinets with a fire resistance of at least two hours.

Protection of records also depends on the particular threat that is present. An important consideration is the speed of onset and the amount of time available to act. This could range from gathering papers hastily and exiting quickly to an orderly securing of documents in a vault. Identifying records and information is most critical for ensuring the continuity of operations.

Records should not be retained only as proof of financial transactions, but also to verify compliance with legal and statutory requirements. In addition, businesses must satisfy retention requirements as an organization and employer. These records are used for

independent examination and verification of sound business practices.

Federal and state requirements for record retention must be analyzed. Each organization should have its' legal counsel approve its own retention schedule. As well as retaining records, the organization should be aware of the specific Record Salvage Procedures to follow for different types of media after a disaster.

Recovery Plan Phase

The Recovery Plan phase deals with recovery strategies that provide a mean to restore IT operations quickly and effectively following a service disruption. The strategies should address residual risks identified in the Business Impact Analysis (BIA). Several alternatives should be considered when developing the strategy, including cost, allowable outage time, security, and integration with larger, organization-level Disaster Recovery Plans.

The recovery strategy selected should address the potential impacts identified in the BIA and should be integrated into the system architecture during the Design and Implementation Phases of the system life cycle.

For this phase automation is a key feature. It is too demanding for personnel staff to be able to execute all the needed functions at the verge of a disaster. However, all the automation policies and people interaction must be rehearsed to exhaustion to reach success.

The topics that can be considered when developing the Recovery Plans are:

- Systems recovery

A Systems Recovery Plan should address mission-critical application hosts, both centralized and distributed.

- Network recovery

Network Recovery Plans are developed to provide for the restoration of:

- Internal LAN and peripheral network supports for mission-critical business processes
- External WANs and telecommunications services
- Communications between recovered systems and end users

- Activation of Disaster Recovery Teams

The Disaster Recovery Management Team must coordinate all activities of all emergency recovery teams. Quick and valid decisions are key in an exceptional situation. The management team must have the authority for all decisions, including financial decisions. A notification plan must be available to activate all emergency teams. The experience of 9/11 disaster demonstrated, that when a major disaster occurs, many people can be affected and not available in the emergency teams. A successful DR plan will take this into account to ensure that the business can be recovered and maintained even if key people are not available.

- End user recovery

End User Recovery Plans, often ignored in traditional Disaster Recovery planning, consist of strategies to provide end users with a mechanism for interacting with restored systems and networks to perform useful work.

Alternative sites

The most practical alternatives for processing in case of a disaster should be researched and evaluated. It is important to consider all aspects of the organization such as: facilities, hardware, software, communications, data files, customer services, user operations, end-user systems, and other processing operations.

The alternatives, dependent upon the evaluation of the computer function, may include: hot, warm, or cold sites reciprocal agreements, two data centers, multiple computers, service centers, consortium arrangement, vendor supplied equipment, or a combinations of all of these.

Although major disruptions with long-term effects may be rare, they should be accounted for in the Disaster Recovery Plan. Thus, the plan must include a strategy to recover and perform system operations at an alternate facility for an extended period. In general, three types of alternate sites are available:

- Dedicated site owned or operated by the company
- Reciprocal agreement with an internal or external entity
- Commercially leased facility

Regardless of the type of alternate site chosen, the facility must be able to support system operations as defined in the Contingency Plan. The three alternate site types also may be categorized in terms of their operational readiness. Based on this factor, sites may be classified as cold sites, warm sites, hot sites, mobile sites, and mirrored sites. Progressing from basic to advanced, the sites are described below:

- Cold sites

Cold sites typically consist of a facility with adequate space and infrastructure (electric power, telecommunications connections, and environmental controls) to support the IT system. The space may have raised floors and other attributes suited for IT operations. The site does not contain IT equipment and usually does not contain office automation equipment, such as telephones, facsimile machines, or copiers. The organization using the cold site is responsible for providing and installing the necessary equipment and telecommunications capabilities.

- Warm sites

Warm sites are partially equipped office spaces that contain some or all of the system hardware, software, telecommunications, and power sources. The warm site is maintained in an operational status ready to receive the relocated system. The site may need to be prepared before receiving the system and recovery personnel. In many cases, a warm site may serve as a normal operational facility for another system or function, and in the event of Contingency Plan activation, the normal activities are displaced temporarily to accommodate the disrupted system.

- Hot sites

Hot sites are office spaces appropriately sized to support system requirements and configured with the necessary system hardware, supporting infrastructure, and support personnel. Hot sites are typically staffed 24 hours a day, 7 days a week. Hot site personnel begin to prepare for the system arrival as soon as they are notified that the Contingency Plan has been activated.

If the RTO is short the data will be mirrored to the site over fibre links either synchronously or asynchronously depending on how far apart the two sites are. If the RTO is long, then the data can be stored on backup tapes and installed prior to the site going live.

- Mobile sites

Mobile sites are self-contained, transportable shells custom-fitted with specific telecommunications and IT equipment necessary to meet system requirements. These are available for lease through commercial vendors. The facility often is contained in a tractor-trailer and may be driven to and set up at the desired alternate location. In most cases, to be a viable recovery solution, mobile sites should be designed in advance with the vendor, and a service-level agreement (SLA) should be signed between the two parties. This is necessary because the time required to configure the mobile site can be extensive, and without prior coordination, the time to deliver the mobile site may exceed the system's allowable outage time.

- Mirrored sites

Mirrored sites are fully redundant facilities with full, real-time information mirroring. Mirrored sites are identical to the primary site in all technical respects. These sites provide the highest degree of availability because the data is processed and stored at the primary and alternate site simultaneously. These sites typically are designed, built, operated, and maintained by the organization.

A requirement for these mirrored sites is the availability of synchronous remote copy among the I/O controllers. Usually companies implementing mirrored sites use both sites to produce work to decrease the expenses - not just a hot stand-by site. In this case we may have workload distribution and continuous availability on top of the Disaster Recovery solution.

There are obvious cost and setup-time differences among the five options. The mirrored site is the most expensive choice, but it ensures virtually 100 percent availability. Cold sites are the least expensive to maintain, however, they require substantial time to acquire and install the necessary equipment. Partially equipped sites, such as warm sites, fall in the middle of the spectrum. In many cases, mobile sites may be delivered to the desired location within 24 hours. However, installation time can increase this setup time. Table 3.1 on page 3-33 summarizes the criteria that can be employed to determine which type of alternate site meets the organization's requirements. Sites should be analyzed further by the organization based on the specific requirements defined in the BIA.

As sites are evaluated, the Disaster Recovery planning coordinator should ensure that the system's security, management, operational and technical controls (such as firewalls and physical access controls) are compatible with the prospective site.

Table 3.1 Alternate site characteristics

Site	Cost	Hardware Equipment	Telecommunications	Setup Time	Location
Cold site	Low	None	None	Long	Fixed
Warm Site	Medium	Partial	Partial/full	Medium	Fixed
Hot site	Medium/high	Full	Full	Short	Fixed
Mobile site	High	Dependent	Dependent	Dependent	Not fixed
Mirrored site	High	Full	Full	None	Fixed

These alternate sites may be owned and operated by the organization (internal recovery) or may be contracted for commercially. If contracting for the site with a commercial vendor, adequate testing time, work space, security requirements, hardware requirements, telecommunications requirements, support services, and recovery days (how long the organization can occupy the space during the recovery period) must be negotiated and clearly stated in the contract.

Customers should be aware that multiple organizations may contract with a vendor for the same alternate site; as a result, the site may be unable to accommodate all of the customers if a disaster affects enough of those customers simultaneously. The vendor's policy on how this situation should be addressed and how the priority status is determined should be negotiated.

Two or more organizations with similar or identical IT configurations and backup technologies may enter a formal agreement to serve as alternate sites for each other or to enter a joint contract for an alternate site. This type of site is set up via a reciprocal agreement or memorandum of understanding. A reciprocal agreement should be entered into carefully because each site must be able to support the other, in addition to its own workload, in the event of a disaster. This type of agreement requires the recovery sequence for the applications from both organizations to be prioritized with a joint perspective. Testing should be conducted at the partner-sites to evaluate the extra processing thresholds, compatible system and backup configurations, sufficient telecommunications connections, and compatible security measures, in addition to functionality of the recovery strategy.

Written agreements for the specific recovery alternatives selected should be prepared, including the following special considerations:

- Contract duration
- Termination conditions
- Testing
- Costs
- Special security procedures
- Notification of systems changes
- Hours of operation
- Specific hardware and other equipment required for processing
- Personnel requirements
- Alternate personnel available if necessary to do the recovery at the recovery site
- Physical space at the recovery site
- Circumstances constituting an emergency
- Process to negotiate extension of service
- Guarantee of compatibility
- Availability
- System source requirements
- Contracted network switchover to recovery site
- Priorities
- Other contractual issues

The strategy should include a combination of methods that complement one another to provide recovery capability of a full spectrum of identified risks. A wide variety of recovery approaches may be considered; the appropriate choice depends on the type of systems and their operational requirements.

Cost considerations

The Disaster Recovery planning coordinator should ensure that the strategy chosen can be implemented effectively with available personnel and financial resources. The cost of each type of alternate site, equipment replacement, and storage option under consideration should be weighed against budget limitations. The coordinator should determine known Disaster Recovery planning expenses, such as alternate site contract fees, and those that are less obvious, such as the cost of implementing a company-wide contingency awareness program and contractor support.

The budget must be sufficient to encompass software, hardware, travel and shipping, testing, plan training programs, awareness programs, labor hours, other contracted services, and any other applicable resources (such as, desks, telephones, fax machines, pens, and paper). The company should perform a cost-benefit analysis to identify the optimum recovery strategy.

Testing and Training Phase

Plan testing is a critical element of a viable contingency capability. Testing enables plan deficiencies to be identified and addressed. Testing also helps evaluate the ability of recovery staff to implement the plan quickly and effectively. Each element of the Disaster Recovery Plan should be tested to confirm the accuracy of the individual recovery procedures and the overall effectiveness of the plan. The following are areas that should be addressed in a contingency test:

- System recovery on an alternate platform from backup tapes
- Coordination among recovery teams
- Internal and external connectivity
- System performance using alternate equipment
- Restoration of normal operations

To derive the most value from the test, explicit test objectives and success criteria should be identified. For example, one test objective might be the recovery of a database, database server, and operating system at an alternate site within eight hours, and database recovery with no errors. The use of test objectives and success criteria enable the effectiveness of each plan element and the overall plan to be assessed. Test results and lessons learned should be documented and reviewed by test participants and other personnel as appropriate. Information collected during the test and post-test reviews that improve plan effectiveness should be incorporated into the Disaster Recovery Plan.

A suggested testing process would have the primary Disaster Recovery personnel to execute the Disaster Recovery plan on a cyclical basis. In doing so they would update all of the documentation for any steps which might have been missed or changed over time. Then the primary personnel would be *declared on vacation* and the Disaster Recovery drill would be executed by a group of alternate personnel at the recovery site following the current documented process. Alternate personnel may have to be utilized at the time of a disaster as the primary Disaster Recovery team may not be available. It is desirable to have the plan *customer environment skills neutral*. Automation of the recovery procedures further helps this process as it can reduce the impact if some of the Disaster Recovery team are unavailable due to the disaster.

Training for personnel with Disaster Recovery Plan responsibilities should complement testing. Training should be provided at least annually; new hires who will have plan responsibilities should receive training shortly after they are hired. Ultimately,

Contingency Plan personnel should be trained to the extent that they are able to execute their respective Recovery Procedures without aid of the actual document. This is an important goal in the event that paper or electronic versions of the plan are unavailable due to the extent of the disaster situation. Recovery personnel should be trained on the following plan elements:

- Purpose of the plan
- Cross-team coordination and communication
- Reporting Procedures
- Security requirements
- Team-specific processes (Activation/Notification, Recovery, and Reconstitution Phases)
- Individual responsibilities (Activation/Notification, Recovery, and Reconstitution Phases)

Change Management Phase

To be effective, the plan must be maintained in a ready state that accurately reflects system requirements, procedures, and policies. IT systems undergo frequent changes because of shifting business needs, technology upgrades, or new internal or external policies. Therefore, it is essential that the Disaster Recovery Plan be reviewed and updated regularly to ensure new information is documented and contingency measures are revised if required.

As a general rule, the plan should be reviewed for accuracy and completeness at least annually or whenever significant changes occur to any element of the plan. Certain elements will require more frequent reviews, such as contact lists. Based on the system type and criticality, it may be reasonable to evaluate plan contents and procedures more frequently. At a minimum, plan reviews should focus on the following elements:

- Operational requirements
- Security requirements
- Technical Procedures
- Hardware, software, and other equipment (types, specifications, and amount)
- Names and contact information of team members
- Names and contact information of vendors, including alternate and off-site POCs (Points of Contact)
- Alternate and off-site facility requirements
- Vital records (electronic and hardcopy)

Because the Disaster Recovery Plan contains potentially sensitive operational and personnel information, its distribution should be marked accordingly and controlled. Typically, copies of the plan are provided to recovery personnel for storage at home and office. A copy should also be stored at the alternate site and with the backup tapes. Storing a copy of the plan at the alternate site ensures its availability and good condition in the event local plan copies cannot be accessed as a result of the disaster.

The Disaster Recovery planning coordinator should maintain a record of copies of the plan and to whom they were distributed. Other information that should be stored with the plan include contracts with vendors (SLAs and other contracts), software licenses, system user manuals, security manuals, and Operating Procedures.

Changes made to the plan, strategies, and policies should be coordinated through the Disaster Recovery planning coordinator, who should communicate changes to the representatives of associated plans or programs, as necessary. The Disaster Recovery planning coordinator should record plan modifications using a Record of Changes, which lists the page number, change comment, and date of change.

The Disaster Recovery planning coordinator should coordinate frequently with associated internal and external organizations and system points-of-contact (POC) to ensure that impacts caused by changes within either organization will be reflected in the Disaster Recovery Plan. Strict version control should be maintained by requesting old plans or plan pages to be returned to the Disaster Recovery planning coordinator in exchange for the new plan or plan pages.

The Disaster Recovery planning coordinator also should evaluate supporting information to ensure that the information is current and continues to meet system requirements adequately. This information includes the following:

- Alternate site contract, including testing times
- Off-site storage contract
- Software licenses
- Memorandum of understanding or vendor SLAs
- Hardware and software requirements
- Security requirements
- Recovery strategy
- Contingency policies
- Training and awareness materials
- Testing scope

Although some changes may be quite visible, others will require additional analysis. The BIA should be reviewed periodically and updated with new information to identify new contingency requirements or priorities. As new technologies become available, preventive controls may be enhanced and recovery strategies may be modified.

This phase must be totally connected with the IT Change Management Team, which is in charge of informing the Disaster Recovery Team of eventual updates in the system, which cause modification in the Recovery Plans.

Disaster Recovery Plan Execution

During a disaster the following steps are performed:

- 1** Notification and activation
- 2** Recovery
- 3** Reconstitution

A discussion of these steps follows.

Notification and Activation Procedures

Here we talk about Notification/Activation Procedures, which are initial actions taken once a system disruption or emergency has been detected or appears to be imminent. These procedures include activities to notify recovery personnel, assess system damage, and implement the plan. At the completion of these procedures, recovery staff will be prepared to perform contingency measures to restore system functions on a temporary basis.

Notification Procedures

An event may occur with or without prior notice. For example, advanced notice is often given that a hurricane will affect an area or that a computer virus is expected on a certain date. However, there may be no notice of equipment failure or a criminal act. Notification procedures should be documented in the plan for either type of situation. The procedures should describe the methods used to notify recovery personnel during business and non-business hours.

Prompt notification is important for reducing the effects on the IT system; in some cases, it may provide enough time to allow system personnel to shut down the system gracefully to avoid a hard crash. Following the disaster event, notification should be sent to the team responsible for the damage assessment, so that it may determine the status of the situation and the appropriate next steps. When damage assessment is complete, the appropriate recovery and support teams should be notified.

Notifications can be accomplished through a variety of methods, including telephone, pager, work or personal electronic mail (e-mail), or cell phone. Notification tools that are effective during widespread disasters include radio and television announcements and Web sites. The notification strategy should define procedures to be followed in the event that certain personnel cannot be contacted. Notification procedures should be documented clearly in the Disaster Recovery Plan.

A common notification method is a call tree. This technique involves assigning notification duties to specific individuals, who in turn are responsible for notifying other recovery personnel. The call tree should account for primary and alternate contact methods and should discuss procedures to be followed if an individual cannot be contacted. The list should identify personnel by their team position, name, and contact information, including home, work, and pager numbers, e-mail addresses, and home addresses. Additionally the primary personnel may not be available due to the disaster, so you would need to have contracted for alternate personnel at the recovery site to execute the Disaster Recovery Plan.

NOTE Be aware, that with some incidents of a physical site disaster, telephone lines can be impacted as well, or the phone lines can be heavily overloaded for hours or even days! A second alternative plan for notification should be considered.

Notification should also be sent to POCs of external organizations or interconnected system partners that may be adversely affected if they are unaware of the situation. Dependent on the type of disruption, the POC may have recovery responsibilities. Therefore, for each system interconnection with an external organization, a POC should be identified to the extent that the organizations will assist each other and the terms under which the assistance will be provided. These POCs should also be listed in an appendix to the plan.

The type of information to be relayed to those being notified should be documented in the plan. The amount and detail of information relayed may depend on the specific team being notified. As necessary, the notification information may include:

- Nature of the incident that has occurred or is impending
- Loss of life or injuries
- Any known damage estimates
- Response and recovery details
- Where and when to convene for briefing or further response instructions
- Instructions to prepare for relocation for estimated time period
- Instructions to complete notifications using the call tree (if applicable)

Activation Plan

The Disaster Recovery plan should be activated only when the damage assessment indicates that some of the activation criteria for that system are met. If an activation criterion is met, the Disaster Recovery planning coordinator should activate the plan. Activation criteria for events are unique for each organization and should be stated in the Disaster Recovery Planning policy statement and may be based on:

- Safety of personnel and/or extent of damage to the facility
- Extent of damage to system (physical, operational, or cost)
- Criticality of the system to the organization's mission (such as, critical infrastructure protection asset)
- Anticipated duration of disruption

Once the system damage has been characterized, the Disaster Recovery planning coordinator may select the appropriate recovery strategy, and the associated recovery teams may be notified.

Recovery Procedures

Recovery operations begin after the Disaster Recovery Plan has been activated, damage assessment has been completed (if possible), personnel have been notified, and appropriate teams have been mobilized. Recovery activities focus on contingency measures to restore temporary IT processing capabilities, whereas activities executed during Reconstitution (see “Reconstitution” on page 41) are directed to repair damage to the original system and restore operational capabilities at the original or new facility. At the completion of the recovery, the system will be operational and performing the functions designated in the plan. Depending on the recovery strategies defined in the plan, these functions could include temporary manual processing, recovery and operation on an alternate system, or relocation and recovery at an alternate site. Teams with recovery responsibilities should understand and be able to perform these recovery strategies well enough that if the paper plan is unavailable during an event, they can still perform the necessary activities.

Sequence of Recovery Activities

When recovering a complex system, such as a WAN involving multiple independent components, Recovery procedures should reflect system priorities identified in the BIA. The sequence of activities should reflect the system's allowable outage time to avoid significant impacts to related systems and their application. Procedures should be written in a sequential format so system components may be restored in a logical manner. For example, if a LAN is being recovered after a disruption, the most critical

servers should be recovered before other, less critical devices, such as printers. Similarly, to recover an application server, procedures should first address operating system restoration and verification before the application and its data are recovered. The procedures should also include instructions to coordinate with other teams when certain situations occur, such as:

- An action is not completed within the expected time frame
- A key step has been completed
- Items must be procured
- Other system-specific concerns

If conditions require the system to be recovered at an alternate site, certain materials will need to be transferred or procured. These items may include shipment of data backup tapes from off-site storage, hardware, copies of the Recovery Plan, and software programs.

Procedures should designate the appropriate team or team members to coordinate shipment of equipment, data, and vital records. References to applicable appendices, such as equipment lists or vendor contact information, should be made in the plan where necessary. Procedures should clearly describe requirements to package, transport, and purchase materials required to recover the system.

Recovery Procedure Highlights

To facilitate Recovery operations, the Disaster Recovery Plan should provide detailed procedures to restore the system or system components. Given the extensive variety of system types, configurations, and applications, this planning guide does not provide specific Recovery procedures.

Procedures should be assigned to the appropriate recovery team and typically address the following actions:

- Notifying internal and external business partners associated with the system
- Obtaining necessary office supplies and work space
- Obtaining and installing necessary hardware components
- Obtaining and loading backup tapes or media
- Restoring critical operating system and application software
- Restoring system data
- Testing system functionality including security controls
- Connecting system to network or other external systems
- Operating alternate equipment successfully

Recovery procedures should be written in a straightforward, step-by-step style. To prevent difficulty or confusion in an emergency, no procedural steps should be assumed or omitted. A checklist format is useful for documenting the sequential Recovery Procedures. It also is useful for troubleshooting problems if the system cannot be recovered properly.

Reconstitution

In the Reconstitution step, recovery activities are terminated and normal operations are transferred back to the organization's facility. During the Recovery Phase, as the contingency activities are performed, reconstitution of the original site should be under way. If the original facility is unrecoverable, the activities in this phase can also be applied to preparing a new facility to support system processing requirements.

Once the original or new site is restored to the level that it can support the IT system and its normal processes, the system may be restored back to the original or to the new site. Until the primary system is restored and tested, the contingency system should continue to be operated.

The Reconstitution step should specify teams responsible for restoring or replacing both the site and the IT system. The following major activities occur in this phase:

- 1** Ensuring adequate infrastructure support, such as electric power, water, telecommunications, security, environmental controls, office equipment, and supplies.
- 2** Installing system hardware, software, and firmware. This activity should include detailed Restoration Procedures similar to those followed in the Recovery Procedures.
- 3** Establishing connectivity and interfaces with network components and external systems.
- 4** Testing system operations to ensure full functionality.
- 5** Backing up operational data on the contingency system and uploading to the restored system.
- 6** Shutting down the contingency system.
- 7** Terminating contingency operations.
- 8** Removing and/or relocating all sensitive materials at the contingency site.
- 9** Arranging for recovery personnel to return to the original facility.

These teams should understand and be able to perform their required functions without a paper plan in the event such documentation is unavailable.

Data Center Availability Planning

In this section we discuss planning strategies and metrics for designing fault tolerant infrastructure, including the following topics:

- Infrastructure, the first line of defense
- Data center availability and planning fundamentals
- Distributed systems considerations
- Storage architecture
- Network architecture

Infrastructure, the First Line of Defense

In a data processing environment, application services depend on every fundamental building block of the data center. A systemic view of a data center shows the relationships and dependencies of every basic component. Every system depends directly on a long list of services. Designing a world class data center from the ground up follows a design process focused on intrinsic properties for security, reliability, and availability of services. Services can range from power, to cooling, to security, to data storage, to networking.

In many cases, large IT infrastructures have grown at tremendous rates, with few, if any design standards for technology. The resulting environments sometimes contain 20+ platform types, 5 to 6 database platforms, and a heterogeneous mix of applications and development tools from different vendors. Usually development teams select application and database technology, while operations teams influence hardware platform decisions. Everyone has their personal technology preference, and through time and employee turnover this naturally tends to lead to increasing entropy in the data center environment.

Since storage management usually crosses so many departmental boundaries, storage infrastructures are often completely unplanned and enterprise data recovery systems are more often distributed than centralized. Beyond the physical infrastructure, enterprise data management policy is a rare occurrence. Storage and storage management continues to move into the limelight for data center operational challenges.

In the following sections, we discuss the importance of strategic planning and the importance of mapping design metrics to availability requirements. The role of IT architecture planning has become central to strategic planning for the enterprise. To create a manageable enterprise environment, organizations must incorporate strategic architecture planning into data center procedures. Enterprise hardware and software standards, architectural design metrics, and the establishment of business requirement driven policy all make a tremendous difference in creating a manageable enterprise environment.

We encourage enterprise deployments of storage management software tools. However designing functional infrastructure to support operational requirements for availability should come first.

Data Center Planning

During the late 1990's e-commerce boom, several companies specialized in providing data center outsourcing facilities for high-growth startup companies. These data centers were built around design fundamentals which incorporated redundancy, resiliency, and high security concepts. The selling point for this high tech real-estate was the fact that companies could trust their hardware operations to a world-class data center specialist. Any organization can adopt the fundamental elements of data center design which created this market for world class data center space.

Space and Growth Considerations

One of the most challenging elements of data center planning includes space planning. Many organizations fill up space at 2 or 3 times the planned rate of space consumption. Technology density (such as, servers and storage) continues to improve, but space planning should be the first element of design. Data center space should accommodate

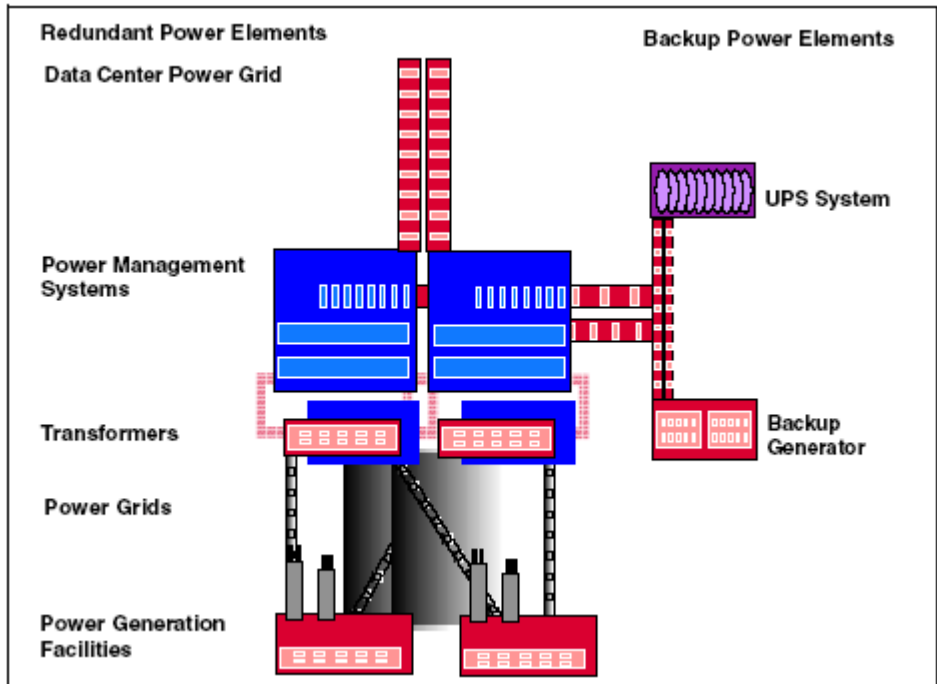
5-10 years of growth, depending on the business type and scale. The cost of building extra space typically pales in comparison to data center relocation costs and risks.

Fundamental Elements of Data Center Design

From a structural standpoint, the data center must be designed to withstand many kinds of disruptions, ranging from building evacuations to regional seismic events. The Risk Analysis phase of the Disaster Recovery Planning process, described in 3.2.2, "Disaster Recovery Planning" on page 44, helps organizations understand the credible risks to the environment. Location of the data center is extremely important too, since building access can be controlled by external entities. This could hinder or complement recovery and security procedures for the data center.

Design fundamentals for a data center components center on scalability, redundancy, and resiliency. As an example, the power infrastructure must provide redundancy and scalability without disruption. Every power management device (transformers, UPS, systems, and so on) must be built with redundancy in mind, just like high availability systems architecture. Some organizations take measures to source power from separate power grids and suppliers, so as to reduce points of infrastructure failure, back to multiple power generation sources. An example of infrastructure redundancy is shown in Figure 3.4 on page 43.

Figure 3.4 Example of power systems design for availability



Similar design methods can be applied to other infrastructure components, including cooling, HVAC, halon, fire prevention, networks, telecommunications, and fibre optics.

Support in the design or retro-fitting process can be obtained through professional services groups, including the data center planning specialists within IBM Global Services.

Preventative Controls

An additional element of data center design includes the use of preventative controls. Redundancy and protection come at a cost that must balance with risks identified in the BIA process. Frequently found preventative controls include:

- Uninterruptible power supplies (UPS) to provide short-term backup power to all systems components (including environmental and safety systems)
- Petroleum powered generators to provide long-term backup power
- Air conditioning systems with adequate excess capacity
- Fire suppression systems
- Fire and smoke detectors
- Water detectors
- Plastic tarpaulins to protect equipment from water damage

Preventative controls must be documented and integrated into the overall DRP. General awareness of how these measures are used is very important for personnel (for example, appropriate response if a fire alarm sounds), and can be instilled via drills, documentation provided to employees and so on. This will ensure they know what to do in the case of a real disaster.

Onsite Parts Maintenance Locker

A simple and effective way to fortify site redundancy is to initiate a program for onsite parts inventory. Components which frequently fail (network cables, connectors, disk drives, tapes, HBAs, and so on) can be stocked in the data center for fast access in the event of component level failures. Some organizations compile and analyze component failures for mean time between failure (MTBF) data, to help cost-effectively prepare for these basic kinds of service interruptions.

Data Center Facilities Security

Security controls for data center access are also extremely important. Creating access controls and procedures helps to fortify operations against malicious human behavior. Scenarios range from card-key access doors to armed and fully supervised one person entry/exit chambers, which include visual recognition, armed guards, and two-stage points of clearance. At a minimum, data center access should be controlled and monitored.

Site locations can also be kept virtually private from public awareness by the use of unmarked buildings and data center locations. Some government and energy installations, for instance, limit the number of people who have knowledge of data center locations and access procedures. However paranoid, these measures provide an excellent level of protection and security.

Disaster Recovery Planning and Infrastructure Assessment

We realize that the majority of organizations rarely have the opportunity to design a data center from the ground up. Best practices for infrastructure and data center design

are applied retroactively at best, and in most cases affect only new systems. The DRP process provides incredible value for mapping logical and physical environments and identifying contingencies within the infrastructure. We encourage readers to approach DRP with this frame of reference.

Distributed Systems Architecture Considerations

Every organization faces the challenge of standardizing platform architectures to help streamline operating efficiencies in an open systems environment. Analysts, such as Gartner Group, have suggested the identification of dual strategic platform vendors, to help control the proliferation of various operating systems and hardware devices in heterogeneous distributed systems environments while still encouraging competition and best pricing for acquisitions. The dual vendor strategy is worth consideration for long term strategic planning efforts.

Server Architecture

Servers should be designed to provide maximum redundancy and scalability. Standards for platform architecture should include redundant power supplies, redundant cooling devices, hot-swappable PCI devices, and hot-swappable internal disk devices. The internal system design should also eliminate most, if not all, single points of failure. Predictive failure analysis capabilities for CPU and memory should also be available from the operating system. For any organization, these architecture standards should be designed, documented, and followed for all production systems.

Server Operating System Clustering Technologies

The iSeries server allows multiple servers to be clustered together for high availability. Clustering software generally monitors systems resources and devices through protocol driven "heartbeat" messages. Depending on events and definitions, the absence of system resources triggers the failover of an application environment from one server to another.

Clustering software adds a layer of complexity to system administration and application administration. Changes made in a production cluster can accidentally trigger a failover event. We suggest high availability clustering for mission-critical environments that have development or testing platforms available to test significant systems modifications. A clustered server environment should be infrequently modified during production.

More information on OS/400 clusters can be found at "Clusters" on page 65.

Capacity Planning

Capacity planning and growth forecasting also affects metrics for systems architecture. Depending on the platform environment, several classes of machine type and size can be applied to specific performance and growth requirements.

System Software

System software, such as operating systems, patches, upgrades, security patches, and systems utilities should be standardized for all platforms in a production environment. This software should be documented and considered a functional component of systems level recovery. Copies of all systems software and licensing information should be cataloged and stored in a secure offsite location.

Configuration Management

An overall objective for infrastructure planning is configuration standardization across the enterprise. Using configuration and platform standards not only affects efficiencies for systems management, but it also directly impacts the ability of an organization to recover systems in the event of a disaster. Detailed records for system configurations should be integrated into the DRP, as an additional guide for internal staff and vendors.

Problem and Change Management

Problem and change management procedures should be established and documented as part of an enterprise server environment. For every production system, a problem and change management procedure and log should be defined and consistently used. Copies of these logs should also be stored offsite and possibly at the Disaster Recovery location.

Disk Storage Architecture Considerations

Like server architecture, disk storage devices should incorporate design capabilities to limit single points of failure. At a minimum, disk devices should include redundant power supplies, redundant cooling devices, hot-swappable adapters, and hot-swappable internal disk drives. The internal system design (that is, the system backplane) should include no single points of failure.

Advanced storage subsystems tend to be either cache-centric or controller-centric. Cache-centric disk subsystems provide generous amounts of memory-cache for I/O operations, generally improving I/O performance and dramatically increasing system cost. Controller-centric disk subsystems offer front-end processing capabilities, with less cache capabilities and generally lower costs. Disk sub-system cost can be evaluated in terms of price per MB for raw storage volumes. The analysis should then include equivalent functionality and overall management cost between each subsystem.

Today, disk subsystems provide some level of RAID (Redundant Array Of Independent Disk) protection. In general, RAID algorithms spread data across multiple disks, to lower the odds of disk failure losing data. Hardware and software level RAID protection is available from most open systems vendors, and should be considered a standard for any disk environment in production.

RAID subsystems typically use three redundancy techniques:

- **Mirroring:** When data is mirrored, the system writes data simultaneously to separate hard drives or drive arrays. If a disk drive fails, the system can still read data from the mirrored logical volume.
- **Parity:** Parity is a technique used to determine whether or not data has been lost or overwritten. In RAID implementations, disk drives can be dedicated or partially used for data parity. Essentially, data redundancy is shared across multiple volumes with parity.
- **Striping:** Striping distributes data across multiple physical drives in a RAID array. A logical group of data is split into block or byte sized increments, and written sequentially across multiple drives. Striping alone is known to improve disk I/O performance, because more drives can access a logical sequence of data at one time. However by itself, it does not provide redundancy.

Several varieties of RAID are available for data protection. The iSeries server supports RAID 5, the most commonly implemented RAID technique. RAID-5 combines parity

with striping across multiple drives in a disk array. This means that data and parity data is spread across multiple volumes. Hot-spare drives can also be built into a RAID-5 array, to provide an additional level of fault tolerance.

While performance sometimes degrades due to parity I/O operations, front end caching in modern disk subsystem helps speed the performance of large scale RAID-5 storage environments.

At a minimum, data should be stored in a RAID storage environment. This provides a basic level of protection to guard data from component level failures in a disk environment.

Advanced Storage Subsystem Functionality

Within and between disk subsystems, several additional capacities exist for data protection and data management. Within subsystems, data volumes can be instantly copied, using techniques such as IBM ESS FlashCopy. Data can also be mirrored synchronously or asynchronously between subsystems remotely and locally using IBM PPRC or replication software from a business partner. See Chapter 12, “Data Considerations for High Availability” on page 209 for more information on data management considerations.

Network Architecture Considerations

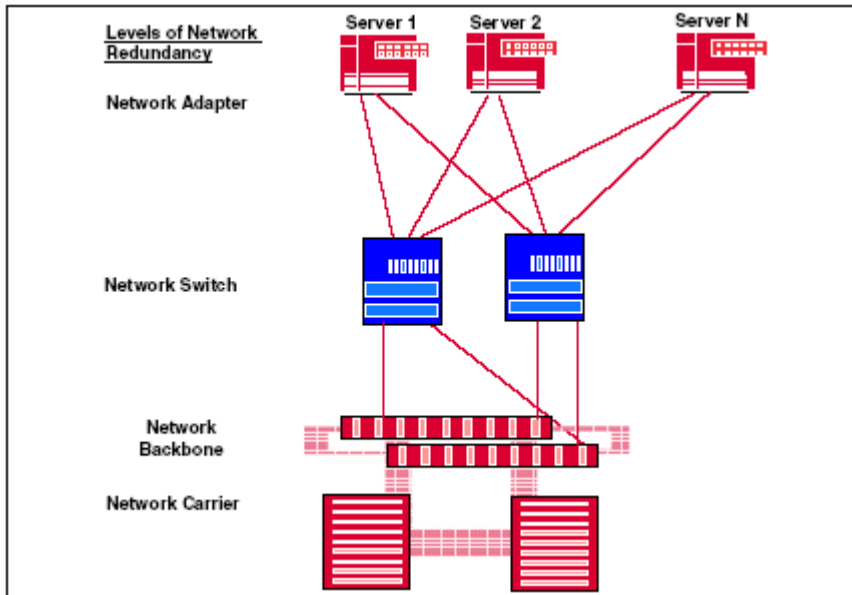
Enterprise design and contingency standards can be applied to LAN, SAN, and WAN environments alike.

Network Architecture

From an architectural view, single points of failure should be avoided for critical systems. Redundant backbones, switches, routers, cables, and adapters should be in place for critical systems identified in the BIA. For recovery procedures, network failover plans should focus on critical systems and TSM architecture.

Network architecture relates to both data and network service availability. Figure 3.5 on page 3-48 illustrates multiple levels of network redundancy to ensure availability for LAN and WAN type network services.

Figure 3.5 Redundant network architecture



A similar design methodology applies to Storage Area Networking (SAN) and data availability. The distance components in a SAN, such as IP extenders, multimode (longwave) fibre, etc. follow the same design metrics that has been discussed: as long as single points of failure are eliminated from any point to point connection, transaction integrity and data availability is best insured.

Network Documentation

The physical and logical network diagrams should accurately reflect the production environment. A physical diagram should display the physical layout of the facility and connected devices. The logical diagram should present the network devices and the connected nodes. Detailed information device names, addresses, DNS, and subnets (or zones) should be documented as well.

Network Management

Networks typically comprise a wide variety of devices, such as hubs, routers, bridges, switches, workstations, PCs, laptops, and printers. The more different components and protocols there are in a network, the more difficult it is to manage them.

Problem detection and response can be at a local or remote level. All network types should be monitored and managed from an enterprise view. Problem management and resolution in a large environment depends not only on skilled resources, but also intelligent monitoring tools for network topology and event management. Various network tools, such as Tivoli NetView, provide this functionality. Tools for SAN management with similar capabilities are emerging as well.

Tools can help you correct problems at the source before users are affected and minimize downtime and performance problems. Management tools provide a detailed view of the overall health and behavior of your network's individual elements. These

elements, such as routers, servers, end systems, data, event flow, and event traffic are useful to record. When network problems occur, use management tools to quickly identify and focus on the root cause of the error.

The focal point of control is where all network information is viewed to provide a complete picture of what is happening. This console or server provides monitoring and alert actions, as well as maintenance capabilities.

With proper network management, the time to respond to and resolve a network error is reduced.

Remote Access Considerations

Remote access should also be securely configured for all network components, so that systems can be administered from a remote location. In a DR scenario, remote access to systems might determine the ability of the DR team to work with normal production systems. Some threats to systems environments, such as biological or chemical terrorism events, might limit the ability for on site work, while systems could maintain functionality in a primary production facility.

WAN Considerations

WAN system configurations should also be well documented. Any vendor or service provider can be a single point of failure for an enterprise, especially for e-commerce or IP communications-dependent businesses. Now that voice and data services rely more on IP services, identifying and eliminating contingencies with network service providers is of ultimate importance. The overall WAN architecture can be designed to eliminate multiple layers of potential failure, ranging from switches to service providers.

Network Security Considerations

Security services for all networks must be an enterprise priority. Firewall device and software configurations should be well documented and secured off site. IP addresses should be treated as confidential information. All network security measures included in production network operations also must be included in the Disaster Recovery Plan. The plan should document the communications providers (and their SLAs) involved in all components of network security.

Network Components

Protocol, physical connection, the hardware and software needed, the backup options available, and network design are all factors to consider for a highly available network.

Components involved in a network include:

- Protocols:
 - Bisynchronous
 - Asynchronous
 - Systems Network Architecture (SNA)
 - Transmission Control Protocol/Internet Protocol (TCP/IP)
 - Internetwork Packet Exchange (IPX)
- Communication lines: Communication lines support the previously described protocols. Special interfaces may be required to connect to the iSeries system, remote

controllers, personal computers, or routers. The type and speed of the line depends on the requirement of performance and response times. Considerations for such facilities include:

- Leased: Private set of wires from the telephone company. These wires can be point-to-point (analog or digital); or multi-point, where the phone company provides the connection of multiple remote sites through the central telephone offices.
- Integrated Services Digital Network (ISDN): Basic or primary access.
- Frame relay: An internal standard. The re-routing of traffic is the responsibility of the frame relay provider. For a fail-safe network, all endpoints should have a dial backup or ISDN connection.
- Virtual Private Networks (VPN): Utilize a service provider that provides encryption and uses the Internet network for the transmission of data to the other site. VPN allows secure transfers over TCP/IP (IPSec) and multiprotocol (L2TP) networks.
- Multiprotocol Switched Services: ATM or LAN.
- Hardware: A factor that may limit the speed in which communication lines are connected. These factors include:
 - Controllers: Manage connections for legacy terminals and personal computers with twinax support cards.
 - Bridges
 - Routers: Routes traffic over a communication network.
 - Frame Relay Access Devices (FRADs): Used to buffer SNA devices from a frame relay network. It also channels SNA, BSC, Asynch, and multiprotocol LAN traffic onto a single frame relay permanent virtual circuit and eliminates the need to have separate WAN links for traditional and LAN traffic.
 - Terminal Adapters: Used in an ISDN network to buffer the device from the physical connection.
 - Modems: Used for analog lines.
 - DSU and CSU: Used for digital lines.
 - Switches: WAN switches for networks carrying high volumes of traffic at a high speed. These are networks that cope with both legacy and new emerging applications in a single network structure. The network mixes data, voice, image, and video information, and transports different traffic types over a single bandwidth channel to each point.
- Software: When the network consists of LANS, routers, and communication lines, a management tool should be used to help manage the network.

From routers to remote controllers, review all network components. If there are critical requirements for remote access, you must provide alternative network paths. This can be as simple as a dialup link or as complex as a multi-path private network.

Resources for Learning

Use the following links to find relevant supplemental information about various topics discussed in this chapter. The information resides on IBM and non-IBM Internet sites, whose sponsors control the technical accuracy of the information.

These links are provided for convenience. Often, the information is useful all or in part for understanding a topic discussed in this chapter. When possible, links are provided to technical papers and Redbooks that supplement the broad coverage of the release documentation with in-depth examinations of particular product areas.

Disaster recovery

- *A Disaster Recovery Solution Selection Methodology* (February 2004)

<http://www.redbooks.ibm.com/redpapers/pdfs/redp3847.pdf>

In this Redpaper, a suggested Disaster Recovery Solution Selection Methodology that is designed to provide assistance to this problem is provided. The intent of this methodology is to allow one to navigate the seemingly endless permutations of Disaster Recovery technology quickly and efficiently, and to identify initial preliminary, valid, cost-justified solutions.

- *IBM TotalStorage Solutions for Disaster Recovery* (January 2004)

<http://www.redbooks.ibm.com/redbooks/pdfs/sg246547.pdf>

This Redbook will help you design a Disaster Recovery solution and presents a Disaster Recovery Solution Selection Methodology to assist in this process. It also provides information on the IBM TotalStorage products which can be used in your solution.

IBM offerings and services

- IBM Business Continuity and Recovery Services

<http://www.ibm.com/services/continuity/recover1.nsf/documents/home>

IBM Business Continuity and Recovery Services can help you map your current infrastructure and design a customized recovery plan for a variety of disaster scenarios. IBM professionals also have the knowledge and experience to help you recover from an actual disaster situation.

AVAILABILITY AND THE iSERIES SERVER

iSeries servers earn some of the highest reliability and availability ratings of any server in the market place today. There are many availability features inherent within OS/400 and iSeries hardware. From its inception onward, the iSeries server is designed to run applications to support core business processes. The iSeries is built for business.

Because it is built to support business, features are designed to avoid unscheduled system downtime whenever possible and to quickly restore the system to an operational state should a failure occur.

This chapter describes the availability features built into the iSeries server.

Single Server Environment

The availability features of the iSeries server can be divided into hardware availability features and software availability features. The following sections discuss the various features.

Hardware Availability Features

This section provides a high-level summary about the availability features built into the iSeries server hardware.

Power Subsystem

Redundant power supplies, cooling fans, dual line cords

Redundant power supplies and cooling fans are options available for iSeries servers. Some models of the system can be ordered with dual line cords.

These features allow power to be supplied from more than one source, with one power source acting as a backup in the event of a disruption to the alternate power source.

Dedicated UPS interface

The iSeries server provides a program interface to monitor and manage the switch to an Uninterruptible Power Supply (UPS) source in the event of a power outage. The system sends a message (that can be monitored for) when it detects power loss. A power handling program can monitor for power-related messages and manage the switchover to a UPS.

Disk Subsystem

Device parity protection (RAID-5)

Device parity protection (RAID-5) is a hardware availability function that protects data from loss due to a disk unit failure or because of damage to a disk. The overall goal of device parity protection is to provide high availability and to protect data as inexpensively as possible.

To protect data, the disk controller or input/output processor (IOP) calculates and saves a parity value for each bit of data. Conceptually, the disk controller or IOP computes the parity value from the data at the same location on each of the other disk units in the device parity set. When a disk failure occurs, the parity value and values of the bits in the corresponding locations on the other disks are used to reconstruct the data. The system continues to run while the data is reconstructed.

Mirrored protection

Mirrored protection is an availability function that protects data from being lost due to failure or because of damage to a disk-related component. Data is protected because the system keeps two copies of data on two separate disk units. When a disk-related component fails, the system continues to operate without interruption. The mirrored copy of the data is used until the failed component is repaired.

Mirroring can be extended to include mirroring the disk IOPs and the busses that the disk units are attached to so the disk subsystem can continue to function even if a disk IOP or a bus fails.

Concurrent maintenance

The iSeries disk subsystem allows maintenance to be performed on a disk drive that is part of a mirrored pair or a RAID-5 set while the system remains operational. Disks can be added concurrently, meaning disk capacity can be increased without disruption to system operations. Because the system manages storage automatically, newly added drives are immediately available for use. There is no requirement to partition the drives or move data to them in order for the system to utilize the drives. The system manages all space as one virtual address. Other than configuring the disks as new hardware devices, special setup is not required to make a new disk operational.

I/O Subsystem

Hot pluggable PCI cards

Hot plugging is made possible by the existence of a power control to individual cards slots. PCI IOPs or IOAs can be added, removed, or replaced while the system remains active.

Dynamic hardware resource reallocation

Each hardware device on the iSeries server has a device description associated with it. The description contains the name of the specific hardware component that the hardware resource is associated with.

If a hardware device fails and there is a backup device for it installed in the system, the device description can be modified to point to the backup device. It can then be substituted for the failing device.

Redundant HSL loops

High Speed Link (HSL) is a new fiber bus structure introduced for iSeries servers. HSL is a 1 Gigabyte per second bus that includes a redundant path capability. If a system bus cable is broken or unplugged, I/O traffic is routed through an alternate path, therefore, avoiding a system outage.

IOP reset

The iSeries I/O architecture uses intelligent I/O processors (IOPs) to control hardware adapters. Should a failure occur in one of these IOPs, it can be reset (or “re-booted”) with the system VARY command. This avoids the need to IPL the system to recover from an I/O error.

Memory

iSeries memory represents “Chip Kill” technology. If a segment of memory fails, the iSeries simply makes unavailable the range of addresses, including the defective address or addresses. A message is sent to the system operator and the hardware error logs are updated with data related to the failure.

Therefore, the system can remain active should a part of main storage fail. Maintenance can be deferred, which allows the system to tolerate memory failures without bringing the system down.

The system also performs a background “scrub” of memory, to detect and correct single and double bit errors.

Hardware Service

Hardware failure notification

With iSeries Service Director, the system “phones home” to a service machine when it detects key hardware component failures. A customer can optionally choose to have a repair engineer dispatched automatically when a hardware failure is logged.

There are many cases recorded where a service engineer comes to a customer’s premises in response to a hardware problem detected by Service Director, and the customer is not even aware of the problem because the system was able to continue operations.

OS/400 and System Software Availability Features

This section provides a high-level summary about the availability features built into the iSeries server software.

Database

DB2 UDB for iSeries is the relational database management system that is built into OS/400. It is an open, extensible, high performance, and scalable RDBMS that adheres to industry standards while leveraging the OS/400 architecture to maintain the value proposition of lower total cost of ownership. The critical database features that is integral to any high available solution include:

- Journaling

Journal management enables you to recover the changes to an object that have occurred since the object was last saved. You can also use journal management to provide an audit trail or to help replicate an object. You use a journal to define what objects you want to protect with journal management. The system keeps a record of changes you make to objects that are journaled and of other events that occur on the system.

iSeries journaling was initially introduced to record changes made to database files. iSeries journaling has evolved over time, as has the style of computing that the system supports. Journaling support is enhanced to include byte stream files (Integrated File System files), data areas, and data queues.

When you use journal management you create an object called a *journal*. You use a journal to define which objects you want to protect. You can have more than one journal on your system. A journal can define protection for more than one object.

The system keeps a record of changes you make to objects that are journaled and of other events that occur on the system. These records are called *journal entries*. You can also write journal entries for events that you want to record, or for objects other than the object that you want to protect with journaling.

The system writes entries to an object called a *journal receiver*. The system sends entries for all the objects associated with a particular journal to the same journal receiver.

In a high availability environment, the following advanced journal features should be considered:

- Journal cache -- JRNCACHE(*YES)

A separately chargeable feature, journal caching provides significant performance improvement for batch applications which perform large numbers of add, update, or delete operations against journaled objects. Applications using commitment control will see less improvement (commitment control already performs some journal caching).

Journal caching modifies the behavior of traditional noncached journaling in batch. Without journal caching, a batch job waits for each new journal entry to be written to disk. Journal caching allows most operations to no longer be held up waiting for synchronous disk writes to the journal receiver.

It is not recommended to use journal caching if it is unacceptable to lose even one recent change in the event of a system failure where the contents of main memory are not preserved. This type of journaling is directed primarily toward batch jobs and may not be suitable for interactive applications where single system recovery is the primary reason for using journaling.

- Journal standby state -- JRNSTATE(*STANDBY)

Journal standby state is a separately purchased feature that prevents most journal entries from being entered into the journal. The advantage to journal standby state

over inactive state is that if there is an attempt to deposit a journal entry, there are no error messages indicating that the entry was not deposited.

- Maximize journal receiver size -- RCVSIZOPT(*MAXOPT2)

The system spreads journal receivers across multiple disk units to improve performance. The journal receiver can be placed on up to ten disk units in a disk pool. If you specify the *MAXOPT2 journal option, then the system can place the journal receiver on up to 100 disk units in an disk pool. Journal entries are written using a "round robin" technique with these arms. In addition, the option sets the maximum size of a journal receiver attached to your journal to approximately one terabyte and allows the system to deposit a journal entry as large as 4 000 000 000 bytes.

The use of journaling within a high availability environment should be coupled with database tuning mechanisms to ensure optimal performance, such as: the reuse of deleted record space, the enablement of concurrent writes, and symmetric multiprocessing for index updates.

- Remote Journaling

Remote journal management allows you to establish journals and journal receivers on a remote system or to establish journal and receivers on independent disk pools that are associated with specific journals and journal receivers on a local system. The remote journaling function can replicate journal entries from the local system to the journals and journal receivers that are located on the remote system or independent disk pools after they have been established.

Remote journaling can be setup to run in synchronous or asynchronous mode. When remote journaling is synchronous, a database update for the source system is not completed until the target system makes the journal entry in its receiver.

Remote journaling can be used in conjunction with database replication for high availability.

- Commitment Control

Commitment control is a function that allows you to define and process a group of changes to resources, such as database files or tables, as a transaction, where a transaction is a group of individual changes to objects on the system that should appear as a single atomic change to the user. Commitment control ensures that either the entire group of individual changes occur on all systems that participate or that none of the changes occur. For example, when you transfer funds from a savings to a checking account, more than one change occurs as a group. To you, this transfer seems like a single change. However, more than one change occurs to the database because both savings and checking accounts are updated. To keep both accounts accurate, either all the changes or none of the changes must occur to the checking and savings account.

Commitment control allows you to:

- Ensure that all changes within a transaction are completed for all resources that are affected.
- Ensure that all changes within a transaction are removed if processing is interrupted.
- Remove changes that are made during a transaction when the application determines that a transaction is in error.

The transaction boundary is defined by the application. In the event of a system failure, commitment control uses journal entries to “roll back” an entire transaction. Therefore, a partial update to database files is avoided.

In addition to the above features, DB2 UDB for iSeries has autonomic computing characteristics which can provide self-configuring, self-healing, self-tuning, and self-protecting capabilities.

Self configuring

The iSeries integration of the DB2 UDB engine eliminates various tasks that need to be performed with database products on other server platforms:

- **Installation**

Other server platforms require that the relational database software be installed on the server. Before the relational database software can be installed, the administrator must first verify that the database software version is compatible with the operating system version installed on the server. With iSeries, DB2 UDB is pre-loaded on the server eliminating the need for product installation. In addition, the DB2 UDB version loaded onto the iSeries always matches the OS/400.

- **Storage Allocation**

Another configuration step that is not needed on DB2 UDB for iSeries is the allocation of storage for database objects such as tables and indexes. For instance, the Create Table SQL statement is all that is needed to create and allocate space for a table on the iSeries. Other database products require storage (i.e., table space) to be allocated on a prior step before the Create Table statement can be executed. As data is added to the table and index objects, other databases need to be continually monitored to verify if additional storage needs to be manually added to the pre-allocated storage space for a database object. In contrast, DB2 UDB for iSeries automatically allocates additional storage for database objects as it's needed without user intervention. DB2 UDB for iSeries goes one step further by also spreading the storage for a single database object across multiple disk drives - this automatic data spreading eliminates disk hot spots by spreading the load across all disk drives on the systems. Database administrators on other systems have to manually configure the database to get an even distribution of the data across the disk units.

- **Database Accessibility**

The iSeries database communication servers that are needed for ODBC and JDBC middleware access are started automatically when TCP/IP is activated on the iSeries server - another database configuration step performed by the system itself.

- **National Language Considerations**

DB2 UDB for iSeries also interrogates the national language setting (English, French, etc.) of an iSeries server. Based on this language setting, DB2 UDB automatically adjusts to use the character set, date format, etc. that is associated with that language.

Self healing

As data in a database is changed and deleted the associated indexes are updated to reflect the data changes. Depending on the order that data is added and changed, the underlying data structure (typically based on a binary tree) can become unbalanced. On other systems, an administrator would manually re-balance the underlying data structure by reorganizing the system. On iSeries, the database engine is continually

monitoring the underlying index structure and rebalancing the index structure internally when necessary. This allows your applications to continue running without having to take the system down for index reorganization.

Database healing is also needed after an abnormal system termination. Although system failures are not a common occurrence with the iSeries, DB2 UDB for iSeries is fully equipped to address the situation. When a crashed iSeries server is restarted, DB2 UDB will automatically restore the database to a consistent state during the IPL of the system.

To speed database recovery after an abnormal system failure, the iSeries also provides *system-managed access path protection* (SMAPP). Access paths (i.e., indexes and keyed logical files) need additional protection because if indexes are in flux and the corresponding changes are not written out to disk when the system crashes, then the entire index will have to be rebuilt after the system is restarted. Recovery can be delayed substantially if there are just a couple indexes that need to be rebuilt over the largest tables (or physical files). Access paths can be journaled explicitly by an administrator to prevent the indexes from being rebuilt, but that doesn't fit the model of a self-managed system. With SMAPP activated, the database engine automatically will selectively start journaling only those access paths that it has determined to be at risk and that it determines are worth protecting. This risk determination occurs based on a user-specified value (set with EDTRCYAP CL command) which identifies an upper limit regarding how much time they would be willing to spend on access path recovery phase of a system restart (IPL). The value can be set at a system-wide level or at a more granular auxiliary storage pool (ASP) level. As part of the risk assessment, the database engine continually monitors change activity for all indexes on the system. Those indexes that have a high degree of activity that are not yet journaled and are of sufficient size to justify the extra run-time overhead are then automatically protected by starting journaling on their behalf. The system “automatically” protects at-risk access paths instead of relying on a human to intervene and start journaling them via the STRJRNAP command.

Journaling is the primary method that is used on the iSeries to ensure a safe and fast database recovery after a system crash. iSeries provides another self-healing feature, system-managed journal receivers, to avoid database processing delays and to reduce the administrative requirements of journaling. The journal receiver is the object that actually stores the logged database changes - over time the size of the journal receiver object will grow as more and more changes are made to the data. When the journal receiver approaches the maximum receiver size, then a new receiver object needs to be attached to the journal. If not, a hard error will stop the database processing on your system. The system-managed journal receiver option (MNGRCVR (*SYSTEM) on CRTJRN & CHGJRN commands) prevents this hard error from occurring by automatically attaching a new journal receiver once the existing journal receiver nears the maximum size limit. An administrator no longer has to intervene to check on the receiver size or to attach a new journal receiver, the system handles it.

Self tuning

The iSeries' cost-based query optimizer is a key component of the self-tuning aspects of the DB2 UDB engine. The integration of the iSeries query optimizer's costing algorithms allows it to automatically adjust to system resource changes. This allows new system resources such as additional processing power or additional memory to be fully utilized by the DB2 UDB for iSeries engine to achieve the fastest performance. For instance, if the iSeries capacity on demand capability has been used to enable additional processors on a server, the DB2 optimizer will immediately recognize the new processing power and attempt to fully utilize it.

To benefit from recent database changes (e.g., new index available) or system changes (e.g., more memory), other databases require a bind operation to be performed by the administrator to force the possible update of an access plan for an SQL statement. Bind operations are usually only required for programs with SQL embedded in a high-level language program. DB2 UDB for iSeries does not even include a bind utility or bind operation since it automatically recognizes these changes and updates the access plan when it's appropriate. This is sometimes referred to as late-binding or automatic binding which is performed on the iSeries for both embedded SQL and dynamic SQL (ODBC, JDBC, etc.). DB2 UDB does analyze the changes with some intelligence prior to rebuilding any access plans, so that DB2 UDB is not continuously rebuilding access plans. For example, a memory pool grows in size from 1.2 GB to 1.4 GB is a resource change that will probably not make a significant improvement in the performance of an SQL statement, so in this case DB2 UDB for iSeries decides not to update any existing access plans.

When an SQL statement is executed on the iSeries, it requires both an access plan and an open data path (ODP). The access plan contains information on how the SQL statement will be implemented (e.g., table scan) and the ODP is the pipe used to get the data in or out. The iSeries SQL engine automatically tries to improve performance of frequently executed SQL statements on the server by caching their access plans and open data paths. Instead of spending system resources to create an access or ODP, the cached objects are reused as much as possible. Open data paths are cached at the job (or connection) level. Access plans for dynamic SQL interfaces are also cached at the job (or connection) level as well as a system-wide level. The storage areas used for all this caching are automatically allocated and populated by DB2 UDB for iSeries. Most other databases require an administrator to carve out storage for the database to use for caching.

Statistics collection and maintenance is another task automated by DB2 UDB for iSeries to improve database performance. On other databases this task is not automatic and requires manual intervention to create and refresh statistics. Cost-based optimizers rely on index and table statistics to make educated decisions when trying to determine the fastest way to implement a query request. Cardinality (number of distinct values) is a statistic used by the query optimizer to better understand the data. For instance, if an index exists over the state column the optimizer can use the cardinality statistic to determine how many state values are stored in your database. If all 50 states are represented in your data then that could bias the optimizer a certain way to search the column. A smaller number of state values like 5 could bias the optimizer to search the column in a different way. DB2 UDB for iSeries is unique in that it automatically updates all of the table and index statistics concurrently as rows are being inserted, updated, and deleted. This automatic maintenance ensures that the query optimizer is working with the most current statistical values. Other databases rely on some type of statistics collection utility to refresh the statistic values. If an administrator forgets to perform statistics collection or the statistics collection is not performed often enough, then performance may suffer because the query optimizer will be making decisions based on old data. If the state column only contained 5 state values six months ago, but now contains 40 state values then the optimizer can easily be making the wrong implementation choice if it still thinks the state column only contains 5 distinct values. DB2 UDB for iSeries continual maintenance of statistics ensures that the query optimizer is operating with the latest statistics - without administrator assistance.

Earlier it was noted that DB2 UDB for iSeries is able to automatically react to changes in system resources such as an additional CPU to improve performance. DB2 UDB also has the ability to dynamically adjust how it is using an existing system resource (memory) in an effort to improve database performance. This self-tuning database feature is known as expert cache and activated by changing the paging option for a memory

pool from *FIXED to *CALC (CHGSHRPOOL... PAGING(*CALC)). Once the expert cache option has been activated, the DB2 UDB engine monitors and analyzes the access of database objects. If it detects that every row in a table that is being read sequentially, it will increase the internal blocking size to bring larger portions of the table into memory in an effort to reduce the total number of disk operations. The DB2 UDB engine can also detect if a range of rows in the table is being accessed frequently. If that type of data access is detected, then DB2 UDB will keep the memory pages associated with those rows in memory longer. This action once again reduces the number of disk operations which usually results in improved performance. The only administrative requirement is to activate the expert cache - DB2 UDB for iSeries analyzes and adjusts the database I/O access patterns all by itself to tune the performance of your iSeries server.

Self Protecting

DB2 UDB for iSeries is very well positioned in this area because of the object based structure of OS/400, which DB2 leverages. In addition to the high degree of security that is made possible by the object based kernel, other self protecting attributes that are built into the operating system and available to use include:

- Digital signing of objects to help prevent unauthorized access
- 128 bit data encryption and SSL
- US Government C2 security compliance including object auditing

Storage Management

Auxiliary storage pools (ASP)

iSeries single level storage treats all storage as one large virtual address space (this includes main store memory as well as disk). There is no concept of a disk volume or data set partition. However, the system provides the capability to separate this contiguous address space into smaller disk “pools” to make system backup and recovery faster and to provide Hierarchical Storage Management facilities. These pools are called auxiliary storage pools.

Conceptually, each ASP on the system is a separate pool of disk units for single-level storage. The system spreads data across the disk units within an ASP. If a disk failure occurs, you need to recover only the data in the ASP that contains the failed unit.

The use of ASPs can reduce system backup time. To do this, an ASP can be created to include individual applications and data. A single ASP can then be backed up without impacting business operations while other applications that operate from different ASPs stay online.

Independent ASPs

An independent ASP is a collection of disk units that can be brought online or taken offline independent of the rest of the storage on a system. An independent ASP can be either:

- switchable among multiple systems in a clustered environment, or
- privately connected to a single system.

The benefits, in both multi-system clustered environments and single-system environments, can be significant. For example, in a clustered environment, the use of

independent ASPs can provide disk storage that is switchable amongst servers in the cluster, providing high availability of resources. In a single-system environment, independent ASPs could be used to isolate infrequently used data that does not always need to be present when the system is operational.

Hierarchical storage management

The iSeries Backup Recovery Media Services (BRMS) licensed program offers the Hierarchical Storage Management (HSM) component. BRMS provides automated backup and recovery support for database and IFS files. It also provides automation for system recovery.

HSM moves data across a hierarchy of storage, allowing data that is not heavily used to move to less costly storage. Retrieval of the data is transparent to users and programs. When the object is referenced, BRMS retrieves it for the user or program.

HSM also helps reduce system back up time, as seldom used data is moved out of the system ASP and can be saved outside the backup window used for daily saves of critical business data.

Automated storage management

The iSeries server has long been known for its low cost of ownership. A contributing factor is that the iSeries server does not need a database administrator to track storage utilization and worry about moving data around to balance or enhance disk subsystem performance.

Automated storage management is also an availability feature in that the database does not need to be made unavailable to perform this type of maintenance. OS/400 storage management automatically spreads data across all available disk arms to balance disk arm utilization. It also automatically allocates additional storage as files, libraries, and other objects grow. There is no need to take the database or a file offline to extend its size.

Online disk balancing

If a large number of disk drives are added at once, run the Start ASP Balance (STRASPBAL) CL command to redistribute data across the disk arms and rebalance arm utilization. There is no need to partition data sets or to move data between volumes as required with other databases to balance performance.

Save/Restore

The OS/400 and program products provide functions to save and restore the entire system. A sample of these functions follows.

TIP Saving database files with access paths will take longer for the files to be saved and require more tape, but recovery will be easier.

Save-while-active

Save-while-active provides a means to save an object to tape while the system remains active. Any application using an object being saved while the system is active must temporarily stop processing before the save can occur. Save-while-active then establishes a checkpoint image of the object and saves the object while the application resumes execution. For applications with short commitment control cycles, the checkpoint processing and save process can often be done without ending the application.

Save changed objects

The save/restore history for objects on a system is maintained by OS/400. OS/400 save commands use the history to provide the ability to save only objects that have been changed since the last time the objects were saved. This reduces the amount of data saved and the time required to perform a system backup.

Saves and restores using multiple tape drives

If you have multiple tape drives, you can use multiple jobs to save or restore different libraries to different tape drives at the same time. You can also use multiple jobs to save or restore objects from the same library to different tape drives at the same time. For example, you could run a SAVLIB command with a large data base file omitted by specifying the OMITOBJ parameter from one job, then at the same time run a SAVOBJ to save the data base file only from another job.

If you have multiple tape drives and very large objects, you can also do a parallel save or restore. A parallel save is where one object is spread across several different tapes (pieces of it are written to or read from different tapes at the same time). The restore can be done in any order (which is how it is different from conventional “striping”).

Backup Recovery and Media Services (BRMS)

BRMS provides an automated means to manage tape libraries and to set up system save policies. Save policies can be setup for daily, weekly, and other schedules to ensure critical enterprise data is saved to tape media. BRMS tracks which system objects are saved and the date of the save, and reports objects that are not saved in the operation. BRMS creates a “recovery report”, which lists the steps required to restore a system in the event of an outage where the system must be recovered from backup media. BRMS uses the parallel save and restore support provided in OS/400.

NOTE In addition to BRMS, there are 3rd party software that can be used for backup and recovery.

TCP/IP

iSeries servers support a full function TCP/IP communications stack. The support is built into TCP/IP to facilitate high availability computing in a network environment. A description of these functions follows.

Virtual IP

iSeries support for virtual IP allows the system to assign an IP address without designating it to a physical hardware device. All IP traffic can be routed through this virtual address. Each virtual address can have more than one physical communications adapter and/or system behind it. This way, if a physical card adapter or system fails, traffic can be rerouted to maintain availability. A client can be transparently re-routed. There is no need to re-establish or reconfigure the link to the alternate system.

Virtual IP can also be used for load balancing and to direct sessions across communications adapters in a system. This helps to evenly distribute traffic for workload management.

Security

With the well-known instances today of viruses and server hacking, to have a secure server that is not vulnerable to attack is a key component of availability. OS/400 has no open interfaces to the system kernel, which means the iSeries is highly resistant to

hacking and viruses. The iSeries provides security auditing and uses system journaling support to log security entries. System security auditing can log activities with user profiles, objects on the system, and jobs.

System Software Maintenance

To achieve higher levels of availability when applying PTFs, the iSeries adopts a philosophy to apply PTFs immediately (if possible), and not require a system IPL for the PTF to take effect.

Logical Partitions (LPARs)

Logical partitions allow you to distribute resources within a single iSeries server to make it function as if it were two or more independent servers, as depicted in Figure 4.1.

Figure 4.1 LPAR Logical View - one server looks like multiple servers



Logical partitions fall into two categories, primary partitions or secondary partitions. Each logically partitioned system has one primary partition and one or more secondary partitions. Before secondary partitions are created, all system resources are assigned to the primary partition. Secondary partitions are independent of each other. While each secondary partition maintains a dependency on the primary, it otherwise operates as a stand-alone server.

The primary partition, essentially the partition manager for your server, must remain active for the secondary partitions to be active. It is important to plan carefully on how you operate the primary partition or the types of workload you run in the primary partition. For example, OS/400 commands such as Power Down System (PWRDWN SYS) and applying fixes (PTFs) that require a restart will affect all of the secondary partitions. It is recommended that the primary partition be restricted to simple partition management tasks only and minimal hardware resources.

Each logical partition represents a division of resources in your iSeries server. Each partition is logical because the division of resources is virtual, not along physical boundaries. The primary resources in your server are its processors, memory, buses, and IOPs.

One of the most exciting things about LPARs is the ability to dynamically move resources such as processor(s), memory or interactive performance, across partitions. For example, during workload peaks in a production partition, you can move entire processors or partial processors from one partition to another partition. Alternatively, one can schedule resource movement to happen at a specific time.

From a high availability perspective, logical partitions are very useful in a number of ways:

- **Hot backup:** When a secondary partition replicates another logical partition within the same system, to switch to the backup during partition failure would cause minimal inconvenience. This configuration also minimizes the effect of long save windows. You can take the backup partition off line and save, while the other logical partition continues to perform production work.
- **Integrated Cluster:** Using OptiConnect and high availability application software, your partitioned server can run as an integrated cluster. You can use an integrated cluster to protect your server from most unscheduled failures within a secondary partition. See “Clusters” on page 65 for more information on iSeries Cluster support.
- **Resource Isolation:** You can run different applications or workloads on different partitions in order to isolate one from the other. For example, you can run applications in one partition and have your database in another partition. This will ensure that your database system is not affected by application errors that may cause a database running on the same system not to function properly.

Multiple Server Environments

An iSeries server provides an impressive list of hardware and software availability features (see “Single Server Environment” on page 53). However, the limit of what can be achieved in a single system environment is reached between 99.9% and 99.99% availability. Achieving higher availability (99.999% and above) is only possible using a multiple system approach. Equally as important is the flexibility that a multiple system approach gives you to perform system and software upgrades without disrupting your business by allowing you to use one system in a production environment while updating the other system.

One of the key technologies that enable the iSeries to come close to providing continuous availability is clustering.

Clusters

An iSeries cluster is a collection or group of one or more servers or logical partitions that work together as a single server. Servers in a cluster, called cluster nodes, work cooperatively to provide a single computing solution. The perspective from the end user is that the cluster operates as though it were a single system. Work can be distributed across multiple systems in the cluster. Any single outage (planned or unplanned) in the cluster will not disrupt the services provided to the end user. End user services can be relocated from system to system within the cluster in a relatively transparent fashion.

How a Cluster Works

The cluster infrastructure provided as a part of OS/400, called cluster resource services, provides failover and switchover capabilities for your servers that are used as database

servers or application servers in a client-server environment. If a system outage or a site loss occurs, the functions that are provided on a clustered database server can be switched over to one or more designated backup systems that either:

- Contain a current copy, provided through replication, of your critical application data.
- Become the primary point of access for the resilient device containing that critical data.

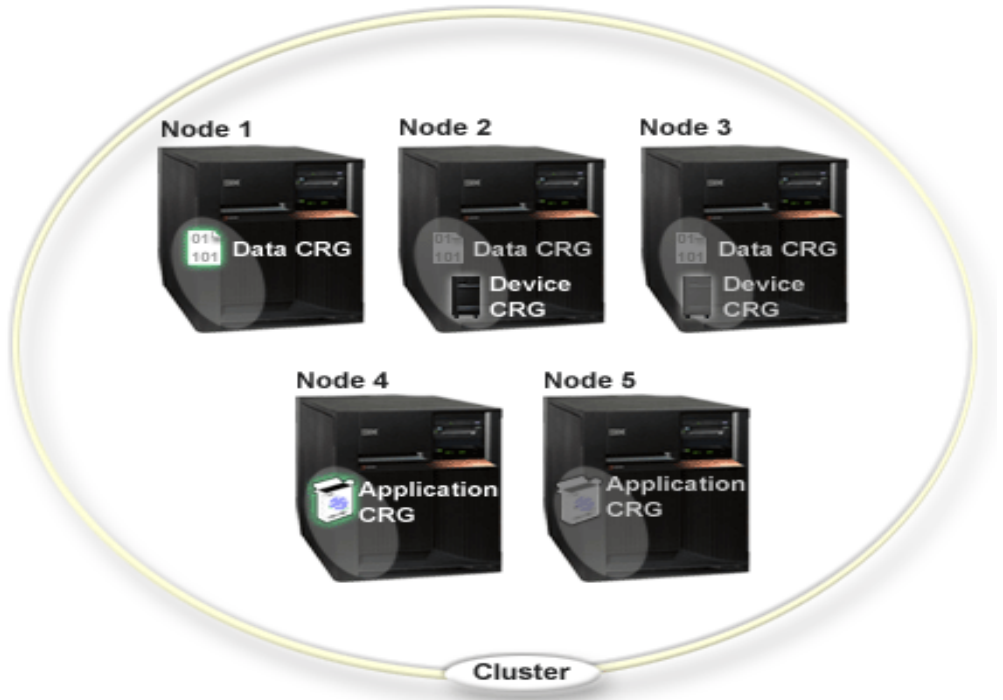
In either scenario, the data and applications remain available. The switching of the access point can be automatic if a system failure, or failover, should happen, or you can control how and when the transfer will take place by manually initiating a switchover.

The switchover and failover will not impact you as a system user or impact applications that you have running on an application server. You can automatically reroute data requests to the new primary node. You can easily maintain multiple replicates of the same data or store the data in a resilient device. If your clusters contain more than two nodes, you can group a system's resilient data (replicated data) together to allow different nodes to act as the backups for each group's resilient data. Multiple backup nodes can be defined. Once a node has been restarted after a failure, cluster resource services provides the means to reintroduce (rejoin) nodes to the cluster and restore their operational capabilities.

Cluster Basics

Figure 4.2 illustrates the basic constructs of a cluster: its *cluster nodes* and *cluster resource groups* (CRGs).

Figure 4.2 Cluster elements



In the cluster above, there are five cluster *nodes*. Nodes are the iSeries servers or logical partitions that are members of the cluster. When you create a cluster, you specify the servers that you want to include in the cluster as nodes.

There are three *cluster resource groups* (CRGs) present in this example. A cluster resource group serves as the control object for a collection of resilient resources. The CRG defines actions to be taken during a switchover or failover. Each CRG accomplishes this by defining the following:

- *Recovery domain* - specifies the role of each node in the CRG:
 - The primary node is the cluster node that is the primary point of access for the resilient cluster resource.
 - A backup node is a cluster node that will take over the role of primary access if the present primary node fails or a manual switchover is initiated.
 - A replicate node is a cluster node that has copies of cluster resources, but is unable to assume the role of primary or backup.
- *Exit program* - manages cluster-related events for that group; one such event would be moving an access point from one node to another node

When you create a CRG in a cluster, the CRG object is created on all nodes specified to be included in the recovery domain. However, a single system image of the CRG object, which you can access from any active node in the CRG's recovery domain, is provided.

That is, any changes made to the CRG will be made on all nodes in the recovery domain.

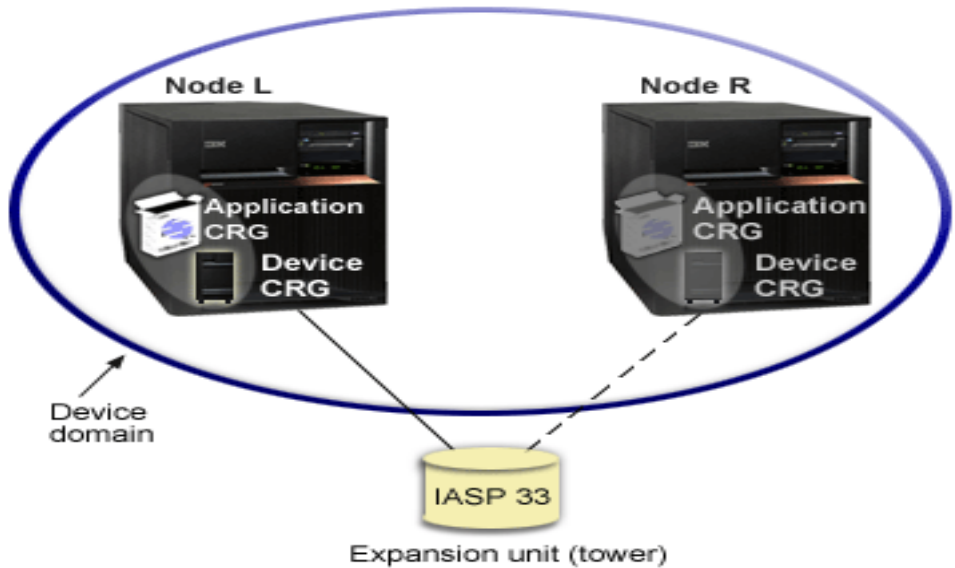
An iSeries cluster supports three types of CRGs: application, data and device. In the example above, one CRG of each type is present:

- *Data CRG*: The data CRG is present on Node 1, Node 2 and Node 3. This means that the recovery domain for the data CRG has specified a role for Node1 (primary), Node 2 (first backup) and Node 3 (second backup). In the example, Node 1 is currently serving as the primary point of access. Node 2 is defined as the first backup in the recovery domain. This means that Node 2 contains a copy of the resource which is kept current through replication. Should a failover or switchover occur, Node 2 would become the primary point of access.
- *Application CRG*: The application CRG is present on Node 4 and Node 5. This means that the recovery domain for the application CRG has specified Node 4 and Node 5. In the example, Node 4 is currently serving as the primary point of access. Should a failover or switchover occur, Node 5 would become the primary point of access for the application.
- *Device CRG*: The device CRG is present on Node 2 and Node 3. This means that the recovery domain for the device CRG has specified Node 2 and Node 3. In the example, Node 2 is currently serving as the primary point of access. This means that the resilient device owned by the device CRG can currently be accessed from Node 2. Should a failover or switchover occur, Node 3 would become the primary point of access for the device.

A device CRG requires a resilient device called an independent disk pool (also called an independent auxiliary storage pool or independent ASP) to be configured on an external device, an expansion unit (tower) or IOP in a logical partition.

The nodes in the recovery domain of a device CRG must also be members of the same device domain. Figure 4.3 on 69 illustrates a device CRG with Node L and Node R in its recovery domain. Both nodes are also members of the same device domain.

Figure 4.3 Device CRG



The following sections gives further details on the various elements that make up a cluster.

Cluster nodes

As has been described previously, a cluster node is any iSeries server or partition that is a member of a cluster. Cluster nodes must be interconnected on an IP network.

A cluster node name is an eight-character cluster node identifier. Each node identifier is associated with one or more Internet Protocol (IP) addresses that represent an iSeries server. Any name can be given to a node. However, for simplicity, make the node name the same as the system name.

Cluster communications that run over IP connections provide the communications path between cluster services on each node in the cluster. The set of cluster nodes that are configured as part of the cluster are referred to as the cluster membership list.

A cluster consists of a minimum of two nodes. The environment can be extended to a cluster with a maximum of 128 nodes.

A node of a cluster can fill one of three possible roles within a recovery domain:

- Primary node, which has the following characteristics: (1) point of access for resilient device; (2) contains principal copy of any replicated resource; (3) current owner of any device resource; and (4) all CRG objects fail over to a backup node.
- Backup node, which has the following characteristics: (1) can take over the role of primary access at failure of the current primary node; (2) contains copy of cluster resource; and (3) copies of data are kept current via replication.

- Replicate node, which has the following characteristics: (1) has copies of cluster resources; and (2) is unable to assume the role of primary or backup (typically used for functions such as data warehousing).

Cluster resource groups (CRGs)

A Cluster Resource Group is an OS/400 external system object that is a set or grouping of cluster resources. The Cluster Resource Group (and replication software) is a foundation for all types of resilience.

Resources that are available or known across multiple nodes within the cluster are called cluster resources. A cluster resource can conceptually be any physical or logical entity (database, file, application, device, and so forth). Examples of cluster resources include iSeries objects, IP addresses, applications, and physical resources. When a cluster resource persists across an outage, that is any single point of failure within the cluster, it is known to be a resilient resource. As such, the resource is resilient to outages and accessible within the cluster even if an outage occurs to the node currently “hosting” the resource.

Cluster nodes that are grouped together to provide availability for one or more cluster resources are called the recovery domain for that group of cluster resources. A recovery domain can be a subset of the nodes in a cluster, and each cluster node may actually participate in multiple recovery domains.

Resources that are grouped together for purposes of recovery action or accessibility across a recovery domain are known as a Cluster Resource Group. The Cluster Resource Group defines the recovery or accessibility characteristics and behavior for that group of resources.

A CRG describes a recovery domain and supplies the name of the Cluster Resource Group exit program that manages cluster-related events for that group. One such event is moving the users from one node to another node in case of a failure.

There are three Cluster Resource Group object types that are used with Cluster Services:

- **Data resilient:** A data resilient CRG enables data resiliency, so that multiple copies of data can be maintained on more than one node in a cluster.
- **Application resilient:** An application resilient CRG enables an application (program) to be restarted on either the same node or a different node in the cluster.
- **Device resilient:** A device resilient CRG enables a hardware resource to be switched between systems. The device CRG is represented by a (device) configuration object as a device type of independent ASP (IASP).

Each CRG definition object specifies the cluster exit program to be called. The exit program is responsible for handling the action codes passed to it by the Cluster Resource Group Manager. Action codes are managed in the APIs that interact with the applicable CRG. And the Cluster Resource Group exit program manages the movement of the access point of a resilient resource.

Exit programs are usually written or provided by high availability business partners and by cluster-aware application program business partners.

Recovery domains

A recovery domain is a subset of nodes in the cluster that are grouped together in a Cluster Resource Group for purposes such as performing a recovery action. Each Cluster Resource Group has a recovery domain that is a subset of the nodes in the cluster.

Here are some facts about recovery domains:

- The nodes within a recovery domain participate in any recovery actions for the resources of the domain.
- Different CRGs may have different recovery domains.
- As a cluster goes through operational changes (for example nodes end, nodes start, nodes fail), the current role of a node may change. Each node has a preferred role that is set when the CRG is created.
- A recovery domain can be a subset of the nodes in a cluster, and each cluster node may participate in multiple recovery domains.

Device domains

The construct known as a device domain is a subset of cluster nodes that share a set of resilient devices. A resilient device might be an independent ASP.

A function of a device domain is to prevent conflicts that would cause the failure of an attempt to switch a resilient device between systems.

Resources involved in a device domain include the structures used to identify and manage the content of the structures across the multiple systems involved in the domain.

Exit programs

The main purpose of exit programs is to “tell” each node in the cluster what to do in case of a failure on the primary system.

When a change occurs in the recovery domain, the exit program associated with the CRG is called on all the active nodes in the recovery domain. Changes range from a system failure to a planned switchover from one system to another, to the addition of a new node to the recovery domain.

The exit program is also called when other events happen, such as when the CRG is started or ended or when an exit program fails. When an exit program is initiated, OS/400 passes the program an action code indicating the event that caused the program call.

It is not mandatory for a device CRG to have an associated exit program. It is for application and data CRGs:

- **Exit programs with data CRGs**

An exit program associated with a data CRG must ensure that as much data as possible (for example, any final journal entries) is transferred to the backup system in the event of a switchover. On the backup system (the new primary system), all outstanding journal entries must be applied. Any other objects must be synchronized as well. When a new node is added to a recovery domain, the exit program may handle the initial data replication.

For example, when a switchover is initiated for a data CRG, the cluster middleware software for data replication writes any remaining journal entries to the backup system. (Remote journaling eliminates this step). Then, when the exit program is called on the backup system (changing its role to primary), the cluster middleware software applies any outstanding journal entries and synchronizes the non-database objects. The software establishes tasks to replicate data from the new primary system to the next backup node, if necessary. Cluster middleware providers are committed to enhancing their products to use the new clustering support (for example, to call the appropriate functions from the exit program), which lets them take advantage of system services such as heartbeat monitoring.

In most cases, a cluster middleware provider supplies data CRG exit programs that take care of all the functions mentioned previously. This means that when a customer wants OS/400 functions combined with data replication, the link between these is already provided.

- **Exit programs with application CRGs**

Exit programs associated with application CRGs are particularly critical, because the application must be restarted on the backup system. OS/400 supplies the exit program with information about the node status change, but all application-specific details, such as current users, record pointers, and even which programs are active, must be handled by the application developer.

Clustered hash table

The *clustered hash table* is the mechanism provided by iSeries cluster support to enable the sharing and replicating of data between cluster nodes.

To use the clustered hash table in your application, the clustered hash table server must be started (a cluster must be active on the node performing the start function). When starting the clustered hash table server, a user would define the domain of the clustered hash table. A CRG (same name as clustered hash table server) is created to manage the domain of the clustered hash table.

The storage for the clustered hash table is not persistent. Not persistent means the storage for the clustered hash table is only known to the server on the local node and only available until the clustered hash table server is ended. All requests to store entries are replicated to other nodes in the clustered hash table domain. When an entry is stored, a time to live value is specified. The entry can become expired, when the time to live value has expired. Expired entries will be removed when processing various functions. For example, when adding another cluster node to the domain of an existing clustered hash table server. The existing clustered hash table entries, if any, are replicated to the cluster hash table domain node being added. Expired entries are removed from the clustered hash table during this process.

The storage and retrieval of data stored in the cluster hash table is done by the use of APIs. The clustered hash table APIs have associated java classes.

The Importance of Having a Backup and Recovery Strategy

A sound data backup and recovery strategy is important in a high-availability environment. The actual backup implementation and restore procedures may likely change as a result of adding high-availability technologies, but the need for regular backups of such things as database files and journal receivers does not change.

So let us say it again. It is imperative that you have a proven strategy for backing up your server; the time and money you spend creating this strategy is more than recovered should you need to restore lost data or perform a recovery. Once you have created a strategy, you must ensure that it works by testing it, which involves performing a backup and recovery and then validating that your data was backed up and restored correctly. If you change anything on your server, you need to assess whether your backup and recovery strategy needs to change.

Every server and business environment is different, but, ideally, you should try to do a full backup of your server at least once a week. If you have a very dynamic environment, you will also have to back up changes to objects on your server since the last backup. Then, if you have an unexpected outage and need to recover those objects, you can recover the latest version of them.

The utilization of such previously-discussed save/restore features as save-while-active, parallel save and restore, and BRMS is critical in making backup and recovery as efficient and painless as possible.

Business Continuity and Recovery Services

IBM Business Continuity and Recovery Services (BCRS) offers an extensive portfolio of services to help you design, set up, and manage a comprehensive, enterprise-wide iSeries 400 or AS/400e business continuity and recovery strategy.

BCRS can help you minimize the effects of network and system outages and plan for their efficient recovery. BCRS consultants can help you do the following:

- Evaluate critical business functions and applications.
- Assess your system environment.
- Design a recovery plan to keep your business running in the event of an extended outage.

BCRS recovery centers are equipped with the latest iSeries and AS/400e server technologies. They are staffed with technical experts to assist you in testing your recovery plan and performing a recovery in the event of a disaster.

For more information about Business Continuity and Recovery Services, please call 1-800-599-9950 or visit the Web site at:

<http://www.ibm.com/services/continuity/recover1.nsf/documents/home>

Resources for Learning

Use the following links to find relevant supplemental information about various topics discussed in this chapter. The information resides on IBM and non-IBM Internet sites, whose sponsors control the technical accuracy of the information.

These links are provided for convenience. Often, the information is useful all or in part for understanding a topic discussed in this chapter. When possible, links are provided to technical papers and Redbooks that supplement the broad coverage of the release documentation with in-depth examinations of particular product areas.

DB2 UDB for iSeries

- DB2 UDB for iSeries Home Page
<http://www.ibm.com/servers/eserver/iseries/db2/>
The home page of DB2 UDB for iSeries.
- DB2 Universal Database for iSeries

<http://publib.boulder.ibm.com/series/v5r2/ic2924/index.htm?info/rzahf/rzahfms1.htm>

This iSeries Information Center topic contains information about DB2 UDB for iSeries.

- Journal Management

<http://publib.boulder.ibm.com/series/v5r2/ic2924/index.htm?info/rzaki/rzakikickoff.htm>

This iSeries Information Center topic contains information about strategies for effective journal management.

- *Striving for Optimal Journal Performance on DB2 Universal Database for iSeries* (May, 2002)

<http://www.redbooks.ibm.com/redbooks/pdfs/sg246286.pdf>

The aim of this IBM Redbook is to provide you with an understanding of the factors which influence journal performance on the iSeries and to help you identify which hardware options and software settings you can employ to minimize this impact. The Redbook also explains the remote journal function on the OS/400 and offers options for reliable and fast transfer of journal entries to a remote iSeries server.

- *AS/400 Remote Journal Function for High Availability and Data Replication* (February 1999)

<http://www.redbooks.ibm.com/redbooks/pdfs/sg245189.pdf>

This IBM Redbook provides suggestions, guidelines, and practical examples about when and how to efficiently use remote journal function provided by OS/400.

iSeries LPARs

- Logical Partitioning homepage

<http://www.ibm.com/servers/eserver/series/lpar/>

The home page for iSeries LPAR.

- Logical partitions

<http://publib.boulder.ibm.com/series/v5r2/ic2924/index.htm?info/rzaj9/rzaj9iclp.htm>

This iSeries Information Center topic contains information about logical partitions.

- *LPAR Configuration and Management - Working with iSeries Logical Partitions* (April 2002)

<http://www.redbooks.ibm.com/redbooks/pdfs/sg246251.pdf>

This IBM Redbook is intended to help you with LPAR planning and implementing considerations.

iSeries clusters

- Clusters

<http://publib.boulder.ibm.com/series/v5r2/ic2924/index.htm?info/rzaig/rzaigicclust.htm>

This iSeries Information Center topic contains information on iSeries clusters.

- *Clustering and IASPs for Higher Availability on the IBM eServer iSeries Server* (April 2002)

<http://www.redbooks.ibm.com/redbooks/pdfs/sg245194.pdf>

This IBM Redbook should be used to gain a broad understanding of the cluster architecture available with OS/400 and where clustering is viable. You should also use it to learn how to plan and implement clustering and independent ASPs.

iSeries server backup and recovery

- Backup and recovery

<http://publib.boulder.ibm.com/series/v5r2/ic2924/index.htm?info/rzak4/rzak4icbackup.htm>

This iSeries Information Center topic contains information on how to plan a backup and recovery strategy, how to back up your server, how to manage tape libraries, and how to set up disk protection for your data. It also includes information about the Backup, Recovery and Media Services, and answers to some frequently asked questions about backup and recovery.

- Backup Recovery and Media Services Home Page

<http://www.ibm.com/servers/eserver/series/service/brms/>

The home page of BRMS.

PROGRAMMING ARCHITECTURES

To have a complete picture of what components need to be scrutinized to ensure high availability, one needs to understand the components that make up the environment in which an application runs in. There are predominantly two programming environments that are currently in use today:

- 1 The programming environment based on the J2EE (Java 2 Enterprise Edition) standard, and
- 2 The Common Gateway Interface (CGI) programming environment, a standard that defines how information is exchanged between a Web server and an external program (CGI program).

The following sections will discuss each of these programming environments.

WebSphere Application Server Network Deployment V5 Architecture

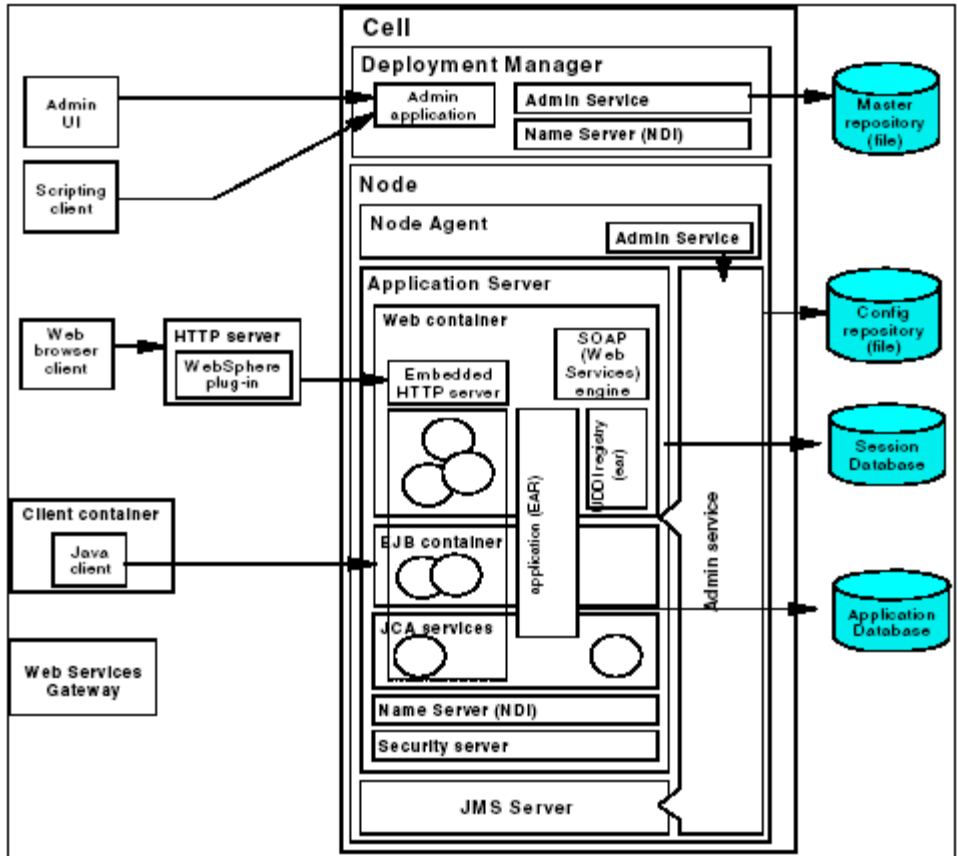
WebSphere Application Server V5 is IBM's implementation of the J2EE platform. WebSphere Application Server is available in several different configurations that are designed to meet a wide range of customer requirements. Each configuration is packaged with other components that provide a unique environment. At the heart of each configuration is a WebSphere Application Server that provides the runtime environment for enterprise applications. This discussion will center on WebSphere Application Server Network Deployment V5 for iSeries. We will refer to this as the *Network Deployment configuration*.

Network Deployment Configuration

The Network Deployment configuration offers central administration and workload management. A Network Deployment environment consists of one or more Base installations and a Deployment Manager installation. The Base application servers are added to the cell and managed by the Deployment Manager. The Network Deployment

package also includes the Web Services Private UDDI Registry and Web Services Gateway.

Figure 5.1 WebSphere Network Deployment configuration



Not shown in the figure above is the Edge Components of IBM WebSphere Application Server Network Deployment V5.0. The Edge Components include the caching proxy and load balancer components.

Cells, Nodes, and Servers

WebSphere Application Server is organized based on the concept of cells, nodes and servers.

Servers

Servers perform the actual code execution. There are several types of servers, depending on the configuration. Each server runs in its own JVM.

- **Application Servers**

The application server is the primary runtime component. This is where the application actually executes. A WebSphere Application Server configuration can have one or more application servers. In the Network Deployment configuration, multiple application servers are maintained from a central administration point. In addition, application servers can be clustered for workload distribution.

- **JMS servers:**

The Network Deployment configuration provides an embedded JMS server for messaging support which runs in a separate JVM. There is one JMS server per node.

Nodes

A node is a logical grouping of WebSphere-managed server processes (i.e. jobs) that share common configuration and operational control. A node is generally associated with one physical installation of WebSphere Application Server. Thus, a node can be a non-partitioned physical machine or a partition in a multi-partitioned physical system.

In the Network Deployment configuration, the concepts of configuring multiple nodes from one common administration server and workload distribution among nodes are introduced. In these centralized management configurations, each node has a node agent that works with a Deployment Manager to manage administration processes.

Cells

A cell is a grouping of nodes into a single administrative domain. In the Network Deployment configurations, a cell can consist of multiple nodes, all administered from a single point. The configuration and application files for all nodes in the cell are centralized into a cell master configuration repository. This centralized repository is managed by the Deployment Manager process and synchronized out to local copies held on each of the nodes.

Servers

WebSphere Application Server uses servers to provide the functions required to host applications. Application servers execute user applications. They may act as individual servers or a member of a cluster. JMS servers provide the embedded messaging support.

Application Server

Application servers provide the runtime environment for application code. They provide containers and services that specialize in enabling the execution of specific Java application components.

Each application server runs in its own Java virtual machine (JVM). Unless you are running Network Deployment, application servers are configured independently and have no workload distribution capabilities.

JMS Server

Messaging support is provided through the use of one of the following JMS providers:

- WebSphere JMS provider (embedded)
- WebSphere MQ JMS provider

- Generic JMS provider

If you use the WebSphere MQ JMS provider or a generic JMS provider, the JMS functions are provided by an external product.

The embedded WebSphere JMS provider, on the other hand, is included in the Network Deployment configuration. The integrated messaging functions provided by the WebSphere JMS provider include support for message-driven beans, point-to-point and publish/subscribe styles of messaging, and integration with the transaction management service.

The functions of the embedded WebSphere JMS provider are implemented by a JMS server. In the Network Deployment configuration, the JMS server is separated from the application server and runs in a separate dedicated JVM. In this case, the configuration for the JMS server is independent of application servers.

Clusters

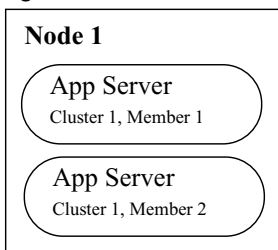
The Network Deployment configuration offers the option of using application server clustering to provide enhanced workload distribution. A *cluster* is a logical collection of application server processes, with the sole purpose of providing workload balancing. Application servers that belong to a cluster are “members” of that cluster and must all have identical application components deployed on them. When you install, update, or delete an application, the updates are automatically distributed to all members in the cluster. Other than the applications configured to run on them, cluster members do not have to share any other configuration data.

For example, one cluster member might be running on a large multi-processor server while another member of that same cluster might be running on a small laptop. The server configuration settings for each of these two cluster members are very different, except in the area of application components assigned to them. In that area of configuration, they are identical.

The members of a cluster can be located on a single node (vertical cluster), across multiple nodes (horizontal cluster) or on a combination of the two:

- In *vertical clustering*, shown in Figure 5.2 on page 80, multiple cluster members for an application server are defined and run in the same node, which may allow the processing power assigned to the node to be more efficiently utilized.

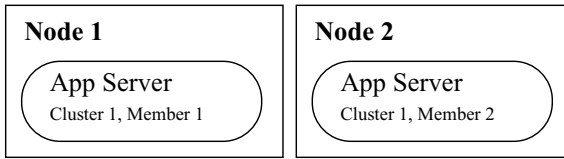
Figure 5.2 Vertical clustering



- In *horizontal clustering*, shown in Figure 5.3 on page 81, cluster members are created and run on multiple nodes, allowing a single WebSphere application to run on several nodes while still presenting a single system image, making the most effective use of the resources of a distributed computing environment. Horizontal scaling is especially effective in environments that contain many smaller, less

powerful machines (or partitions). Client requests that overwhelm a single node can be distributed over several nodes in the system. Failover is another benefit of horizontal scaling. If a node becomes unavailable, its workload can be routed to other nodes containing cluster members.

Figure 5.3 Horizontal clustering



Containers

The J2EE 1.3 specification defines the concept of containers to provide runtime support for applications. There are two containers in the WebSphere application server implementation:

- Web container - processes HTTP requests, servlets and JSPs.
- EJB container - processes enterprise beans (EJBs).

In addition, there is an application client container that can run on the client machine.

Web Container

The Web container processes servlets, JSP files and other types of server-side includes. Each application server runtime has one logical Web container, which can be modified, but not created or removed.

- **Servlet processing:** when handling servlets, the Web container creates a request object and a response object, then invokes the servlet service method. The Web container invokes the servlet's destroy method when appropriate and unloads the servlet, after which the JVM performs garbage collection.
- **Embedded HTTP server:** the Web container runs an embedded HTTP server for handling HTTP(S) requests from external Web server plug-ins or Web browsers. The embedded Web server is based on the IBM HTTP Server product.

Directing client requests to the embedded Web server is useful for testing or development purposes. The use of an external Web server and Web server plug-in as a front-end to the Web container is more appropriate for a production environment.

- **Session management:** support is provided for the `javax.servlet.http.HttpSession` interface described in the Servlet API specification.
- **Web services engine:** Web services are provided as a set of APIs in cooperation with the J2EE applications. Web services engines are provided to support SOAP.

Web server plug-ins

Although the Web container has an embedded HTTP server, a more likely scenario is that an external Web server will be used to receive client requests.

The Web server can serve requests that do not require any dynamic content, for example, HTML pages. However, when a request requires dynamic content (JSP/servlet processing), it must be forwarded to WebSphere Application Server for handling.

The mechanism to accomplish this is provided in the form of a Web server plug-in. The plug-in is included with the WebSphere Application Server packages for installation on a Web server. An XML configuration file, configured on the WebSphere Application Server, is copied to the Web server plug-in directory. The plug-in uses the configuration file to determine whether a request should be handled by the Web server or an application server. When a request for an application server is received, it is forwarded to the appropriate Web container in the application server. The plug-in can use HTTP or HTTPS to transmit the request.

The plug-in also provides workload management for clustered application servers that host Web containers. See “HTTP Plug-in Workload Management (WLM)” on page 90 for further information.

EJB Container

The EJB container provides all the runtime services needed to deploy and manage enterprise beans. It is a server process that handles requests for session, entity, and message-driven beans.

The enterprise beans (packaged in EJB modules) installed in an application server do not communicate directly with the server; instead, the EJB container provides an interface between the EJBs and the server. Together, the container and the server provide the bean runtime environment.

The container provides many low-level services, including threading and transaction support. From an administrative viewpoint, the container manages data storage and retrieval for the contained beans. A single container can host more than one EJB JAR file. The Network Deployment configuration also provides EJB workload management. See “EJB Workload Management (WLM)” on page 95 for further information.

EJB types

In the case of an EJB client interacting with one or more EJBs, the management of state information associated with a series of client requests is governed by the EJB specification and implemented by the WebSphere EJB container, and depends on the types of EJBs that are the targets of these requests.

Stateless session bean

By definition, when interacting with a stateless session bean, there is no client-visible state associated with the bean. Every client request directed to a stateless session bean is independent of any previous request that was directed to the same bean. The container will maintain a pool of instances of stateless session beans of each type, and will provide an arbitrary instance of the appropriate bean type whenever each client request is received. It does not matter if the same actual bean instance is used for consecutive requests, or even if two consecutive requests are serviced by bean instances in the same application server.

Stateful session bean

In contrast, a stateful session bean is used precisely to capture state information that must be shared across multiple consecutive client requests that are part of one logical sequence of operations. The client must take special care to ensure that it is always accessing the same instance of the stateful session bean, by obtaining and keeping an EJB object reference to that bean. At the present time, WebSphere supports the distribution of stateful session bean homes among multiple application servers, but not

the distribution of a specific instance. Each instance of a particular stateful session bean exists in only one application server, and can only be accessed by directing requests to that particular application server (of course, different instances of the same type of stateful session bean, used by different clients, may be spread among multiple servers). The various load-distribution techniques available in WebSphere make special provisions to support this characteristic of stateful session beans.

Entity bean

Most external clients access WebSphere services through session beans, but it is possible for an external client to access an entity bean directly. Furthermore, a session bean inside WebSphere is itself often acting as a client to one or more entity beans also inside WebSphere; if load-distribution features are used between that session bean and its target entity bean, then the same questions arise as with plain external clients.

Strictly speaking, the information contained in an entity bean is not usually associated with a “session” or with the handling of one client request or series of client requests. But it is common for one client to make a succession of requests targeted at the same entity bean instance. Furthermore, and unlike all the previous cases, it is possible for multiple independent clients to access the same entity bean instance more or less concurrently. Therefore, it is important that the state contained in that entity bean be kept consistent across the multiple client requests.

For entity beans, the notion of a session is more or less replaced by the notion of transaction. For the duration of one client transaction to which it participates, the entity bean is instantiated in one container (normally the container where the first operation within that transaction was initiated). All subsequent accesses to that same bean, within that same transaction, must be performed against that same instance in that same container.

In between transactions, the handling of the entity bean is specified by the EJB specification, in the form of a number of caching options:

- With option A caching, WebSphere assumes that the entity bean is used within a single container. Clients of that bean must direct their requests to the bean instance within that container. The entity bean has exclusive access to the underlying database, which means that the bean cannot be clustered or participate in workload management if option A caching is used.
- With option B caching, the bean instance remains active (so it is not guaranteed to be made passive at the end of each transaction), but it is always reloaded from the database at the start of each transaction. A client can attempt to access the bean and start a new transaction on any container that has been configured to host that bean.
- With option C caching (the default), the entity bean is always reloaded from the database at the start of each transaction and made passive at the end of each transaction. A client can attempt to access the bean and start a new transaction on any container that has been configured to host that bean. This is effectively similar to the session clustering facility described for HTTP sessions: the shared state is maintained in a shared database, and can be accessed from any server when required.

Entity beans can participate in workload management as long as the server reloads the data into the bean at the start of each transaction, assuming that transactional affinity is in place. Guaranteed passivation at the end of each transaction is not a requirement for a bean to participate in workload management. Hence, option B and option C caching are both compatible with workload management, but option A caching is not.

Message-driven beans

The MDB is a stateless component that is invoked by a J2EE container when a JMS message arrives at a particular JMS destination (either a queue or topic). Loosely, the MDB is triggered by the arrival of a message.

Messages are normally anonymous. If some degree of security is desired, the listener will assume the credentials of the application server process during the invocation of the MDB.

MDBs handle messages from a JMS provider within the scope of a transaction. If transaction handling is specified for a JMS destination, the listener will start a global transaction before reading incoming messages from that destination. JTA transaction control for commit or rollback is invoked when the MDB processing has finished.

Naming and Name Spaces

Naming is used by clients to obtain references to objects related to those applications, such as Enterprise JavaBeans (EJB) homes.

These objects are bound into a mostly hierarchical structure, referred to as a *name space*. Each application server, Node Agent, and the Deployment Manager has its own name space. However, the separate name spaces are federated (through context links between the name spaces) to form a single logical name space for the cell. See “Name Service” on page 86 for more information on Naming Service.

An InitialContext is used to access objects in the name space. To obtain an InitialContext a bootstrap server and port can be supplied. If one is not supplied, the default values of host=localhost and port=2809 will be used as the provider URL.

WebSphere Application Server name servers are an implementation of the CORBA CosNaming interface.

InitialContext requests participation in workload management when the provider URL is a clustered resource (cluster member) and they do not when they are not a clustered resource.

Client Application Container

The client application container is a separately installed component on the client's machine. It allows the client to run applications in an EJB-compatible J2EE environment.

There is a command-line executable (`launchClient`) that is used to launch the client application along with its client container runtime.

Application Server Services

The application server provides other services besides the containers.

JCA Services

Connection management for access to enterprise information systems (EIS) in WebSphere Application Server is based on the J2EE Connector Architecture (JCA) specification, also sometimes referred to as J2C. The connection between the enterprise application and the EIS is done through the use of EIS-provided resource adapters, which are plugged into the application server. The architecture specifies the connection management, transaction management, and security contracts that exist between the application server and EIS.

The Connection Manager in the application server pools and manages connections. It is capable of managing connections obtained through both resource adapters defined by the JCA specification and data sources defined by the JDBC 2.0 Extensions Specification.

Transaction Service

WebSphere applications can use transactions to coordinate multiple updates to resources as one unit of work such that all or none of the updates are made permanent. Transactions are started and ended by applications or the container in which the applications are deployed.

WebSphere Application Server is a transaction manager that supports the coordination of resource managers through their XAResource interface and participates in distributed global transactions with other OTS 1.2 compliant transaction managers (for example J2EE 1.3 application servers).

WebSphere applications can also be configured to interact with databases, JMS queues, and JCA connectors through their local transaction support when distributed transaction coordination is not required.

The way that applications use transactions depends on the type of application component, as follows:

- A session bean can either use container-managed transactions (where the bean delegates management of transactions to the container) or bean-managed transactions (where the bean manages transactions itself).
- Entity beans use container-managed transactions.
- Web components (servlets) use bean-managed transactions.

In WebSphere Application Server, transactions are handled by three main components:

- A transaction manager that supports the enlistment of recoverable XAResources and ensures that each such resource is driven to a consistent outcome either at the end of a transaction or after a failure and restart of the application server.
- A container in which the J2EE application runs. The container manages the enlistment of XA Resources on behalf of the application when the application performs updates to transactional resource managers (for example, databases). Optionally, the container can control the demarcation of transactions for enterprise beans configured for container-managed transactions.
- An application programming interface (`UserTransaction`) that is available to bean-managed enterprise beans and servlets. This allows such application components to control the demarcation of their own transactions.

Object Request Broker (ORB) Service

An Object Request Broker (ORB) manages the interaction between clients and servers, using IIOP. It enables clients to make requests and receive responses from servers in a network-distributed environment.

The ORB provides a framework for clients to locate objects in the network and call operations on those objects as if the remote objects were located in the same running process as the client, providing location transparency. The client calls an operation on a local object, known as a stub. Then the stub forwards the request to the desired remote object, where the operation is run and the results are returned to the client.

The client-side ORB is responsible for creating an IIOP request that contains the operation and any required parameters, and for sending the request on the network. The server-side ORB receives the IIOP request, locates the target object, invokes the requested operation, and returns the results to the client. The client-side ORB demarshals the returned results and passes the result to the stub, which, in turn, returns to the client application, as if the operation had been run locally.

WebSphere Application Server uses an ORB to manage communication between client applications and server applications as well as communication among product components.

Name Service

Each application server hosts a name service that provides a Java Naming and Directory Interface (JNDI) name space. The service is used to register resources hosted by the application server. The JNDI implementation in WebSphere Application Server V5 is built on top of a Common Object Request Broker Architecture (CORBA) naming service (CosNaming)

JNDI provides the client-side access to naming and presents the programming model used by application developers. CosNaming provides the server-side implementation and is where the name space is actually stored. JNDI essentially provides a client-side wrapper of the name space stored in CosNaming, and interacts with the CosNaming server on behalf of the client.

The naming architecture is used by clients of WebSphere applications to obtain references to objects related to those applications. These objects are bound into a mostly hierarchical structure, referred to as a *name space*. The name space structure consists of a set of name bindings, each consisting of a name relative to a specific context and the object bound with that name. The name space can be accessed and manipulated through a name server.

The following are features of a WebSphere Application Server V5 name space:

- The name space is distributed

For additional scalability, the name space for a cell is distributed among the various servers. The Deployment Manager, node agent and application server processes all host a name server. In previous releases, there was only a single name server for an entire administrative domain.

In WebSphere Application Server V5, the default initial context for a server is its server root. System artifacts, such as EJB homes and resources, are bound to the server root of the server with which they are associated.

- Transient and persistent partitions

The name space is partitioned into transient areas and persistent areas. Server roots are transient. System-bound artifacts such as EJB homes and resources are bound under server roots. There is a cell persistent root, which can be used for cell-scoped persistent bindings, and a node persistent root, which can be used to bind objects with a node scope.

- Federated name space structure

A name space is a collection of all names bound to a particular name server. A name space can contain naming context bindings to contexts located in other servers. If this is the case, the name space is said to be a *federated name space*, because it is a collection of name spaces from multiple servers.

The name spaces link together to cooperatively form a single logical name space.

In a federated name space, the real location of each context is transparent to client applications. Clients have no knowledge that multiple name servers are handling resolution requests for a particular requested object.

In the Network Deployment configuration, the name space for the cell is federated among the Deployment Manager, node agents, and application servers of the cell. Each such server hosts a name server. All name servers provide the same logical view of the cell name space, with the various server roots and persistent partitions of the name space being interconnected by means of the single logical name space.

- Configured bindings

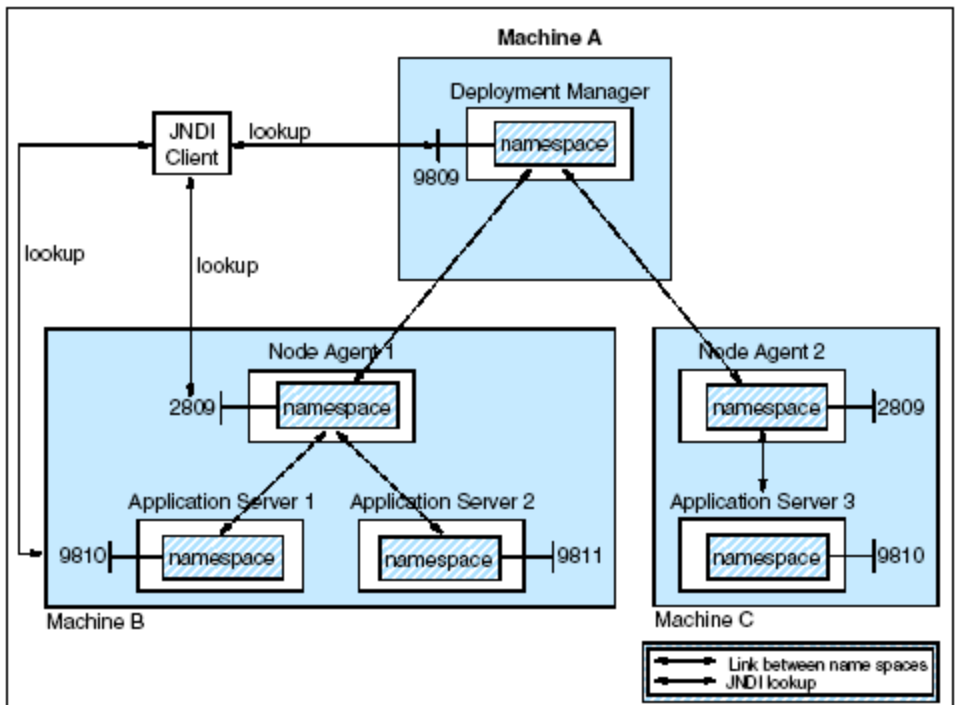
The configuration graphical interface and script interfaces can be used to configure bindings in various root contexts within the name space. These bindings are read-only and are bound by the system at server startup.

- Support for CORBA Interoperable Naming Service (INS) object URLs

WebSphere Application Server V5 contains support for CORBA object URLs (corbaloc and corbaname) as JNDI provider URLs and lookup names, as required by the J2EE 1.3 specification.

Figure 5.4 summarizes the naming architecture and its components.

Figure 5.4 WebSphere JNDI naming architecture



Security Service

Each application server JVM hosts a security service that uses the security settings held in the configuration repository to provide authentication and authorization functionality.

Dynamic Caching

The dynamic cache service improves performance by caching the output of servlets, commands and JSP files. The dynamic cache works within an application server, intercepting calls to cacheable objects, for example, through a servlet's `service()` method or a command's `execute()` method, and either stores the object's output to or serves the object's content from the dynamic cache.

Because J2EE applications have high read-write ratios and can tolerate small degrees of latency in the currency of their data, the dynamic cache can create an opportunity for significant gains in server response time, throughput, and scalability.

The following caching features are available in WebSphere Application Server.

- **Cache replication:**

Cache replication among cluster members takes place using the WebSphere internal replication service. Data is generated one time and copied or replicated to other servers in the cluster, thus saving execution time and resources.
- **Cache disk offload:**

By default, when the number of cache entries reaches the configured limit for a given WebSphere server, eviction of cache entries occurs, allowing new entries to enter the cache service. The dynamic cache includes an alternative feature named disk offload, which copies the evicted cache entries to disk for potential future access.
- **Edge Side Include caching:**

The Web server plug-in contains a built-in ESI processor. The ESI processor has the ability to cache whole pages, as well as fragments, providing a higher cache hit ratio. The cache implemented by the ESI processor is an in-memory cache, not a disk cache, therefore, the cache entries are not saved when the Web server is restarted.
- **External caching:**

The dynamic cache has the ability to control caches outside of the application server, such as IBM Edge Server, and a non- IBM HTTP Server's FRCA cache. When external cache groups are defined, the dynamic cache matches externally cacheable cache entries with those groups, and pushes cache entries and invalidations out to those groups. This allows WebSphere to manage dynamic content beyond the application server. The content can then be served from the external cache, instead of the application server, improving savings in performance.

Message Listener Service

The message listener service is an extension to the JMS functions of the JMS provider. It provides a listener manager that controls and monitors one or more JMS listeners. Each listener monitors a JMS destination on behalf of a deployed message-driven bean.

Admin Service

The admin service runs within each server JVM. In the Network Deployment configuration, each of the following hosts an admin service:

- Deployment Manager
- Node agent
- Application server
- JMS server

The admin service provides the necessary functions to manipulate configuration data for the server and its components. The configuration is stored in a repository in the server's file system.

PMI Service

WebSphere Application Server collects data on runtime and applications through the Performance Monitoring Infrastructure (PMI). This infrastructure is compatible with and extends the JSR-077 specification.

PMI uses a client-server architecture. The server collects performance data from various WebSphere Application Server components and stores it in memory. This data consists of counters such as servlet response time and data connection pool usage. The data can then be retrieved using a Web client, Java client or JMX client. The Network Deployment configuration contains Tivoli Performance Viewer, a Java client which displays and monitors performance data.

WebSphere Application Server also collects data by timing requests as they travel through the product components. PMI request metrics log time spent in major components, such as Web containers, EJB containers, and databases. These data points are recorded in logs and can be written to Application Response Time (ARM) agents used by Tivoli monitoring tools.

Virtual Hosts

A virtual host is a configuration enabling a single host machine, where a host machine can be a non-partitioned physical machine or a partition in a multi-partitioned physical machine, to resemble multiple hosts machines. It allows a single physical machine or partition to support several independently configured and administered applications. It is not associated with a particular node. It is a configuration, rather than a “live object”, which is why it can be created, but not started or stopped.

Each virtual host has a logical name and a list of one or more DNS aliases by which it is known. A DNS alias is the TCP/IP host name and port number used to request the servlet, for example `yourHostName:80`. When a servlet request is made, the server name and port number entered into the browser are compared to a list of all known aliases in an effort to locate the correct virtual host and serve the servlet. If no match is found, an HTTP 404 error is returned to the browser.

IBM WebSphere Application Server provides two default virtual hosts:

- `default_host`
Used for accessing most applications.
- `admin_host`

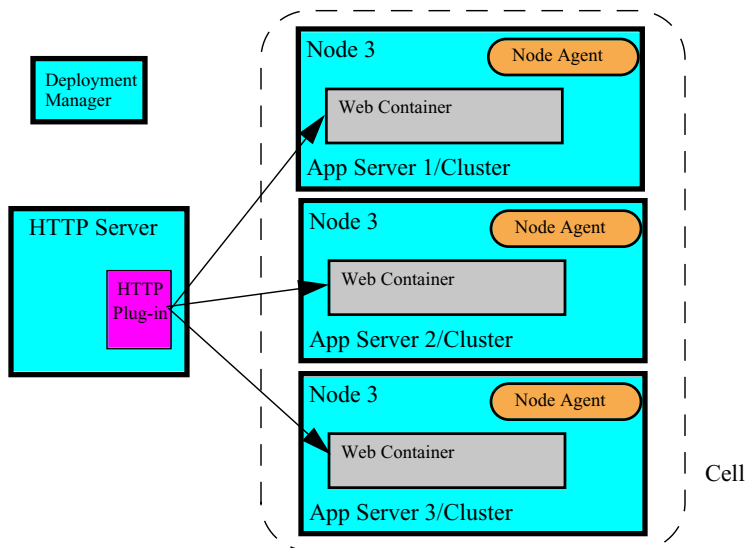
Specifically configured for accessing the WebSphere Application Server V5 administrative console. Other applications are not accessible through this virtual host.

HTTP Plug-in Workload Management (WLM)

Clustering application servers that host Web containers automatically enables plug-in workload management for the application servers and the servlets they host. Routing of servlet requests occurs between the Web server plug-in and the clustered application servers using HTTP or HTTPS.

Figure 5.5 on page 90 shows how HTTP plug-in workload management is handled by WebSphere.

Figure 5.5 WebSphere HTTP plug-in spraying requests to application servers



This routing is based on weights associated with the cluster members. If all cluster members have identical weights, the plug-in will send equal requests to all members of the cluster assuming no strong affinity configurations. If the weights are scaled in the range from zero to twenty, the plug-in will route requests to those cluster members with the higher weight value more often.

The Web server plug-in will temporarily route around unavailable cluster members.

Request Processing

Once a cluster and its members have been set up, the plug-in can start directing requests from the Web server to the correct application server. When processing a request, the plug-in goes through the following steps (following assumes users asks for the page `http://web1:80/snoop` from their browser):

- 1 The Web server immediately passes the request to the plug-in. All requests go to the plug-in first.
- 2 The plug-in then starts by looking at all the defined Routes in the plugin-cfg.xml. For each Route, it searches through its configuration to find if there is any match to the virtual host web1:80 and URI /snoop. It will find the first match and then decide that WebSphere should handle the request. If there isn't any match, WebSphere will not handle the request.
- 3 The plug-in takes the request and separates the host name and port pair and URI. The plug-in now starts by looking at all the defined Routes. It gets a Route and then searches for a virtual host match to web1 port 80 in that Route. It matches that host name and port to web1:80 in the VirtualHost block.
- 4 The plug-in then tries to match the URI /snoop in the current Route. It searches its configuration for a URI mapping that matches the requested URI in the UriGroup block. It matches /snoop in the UriGroup.
- 5 Once the VirtualHostGroup and UriGroup are found, it sets a score depending on the number of characters in the URI and virtual host.
- 6 The plug-in continues to the next Route and searches through virtual hosts and URI's setting scores for matches with any URI and virtual host match. The Route that has the highest score is chosen and the ServerCluster is set.
- 7 The plug-in now checks the request to see if any session identification has been passed to the server. Our request does not contain any session information.
- 8 The plug-in chooses a cluster member to manage the request. A server is chosen to handle the request. If there is a session identifier and a CloneID associated with it, the plug-in will choose a server based on the CloneID.
- 9 This cluster member has two transports associated to it; HTTP and HTTPS are defined. As this request is HTTP, the cluster member uses the HTTP transport definition.
- 10 The request is sent to the cluster member and successfully processed. The plug-in passes the response on to the Web server, which in turn passes it back to the user's browser. Note that since HTTP 1.1 is used for connections between plug-in and Web container, it is possible to maintain a connection (stream) over a number of requests. If a stream is already established, the plug-in uses the existing stream.

The plug-in Configuration File

The previous section showed the plug-in process flow, now let us look at the configuration file used by the plug-in and some of the fields that affect request processing.

The configuration file, by default, is placed in the following directory:

<WAS_HOME>/config/cells

The example below shows how the configuration file is stored. It has been manually edited to show some of the non-default options that are available.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Config ASDisableNagle="false" IISDisableNagle="false"
  IgnoreDNSFailures="false" RefreshInterval="60" ResponseChunkSize="64">
<Log LogLevel="Error" Name="/QIBM/UserData/WebAS5/ND/ND839/logs/http_plugin.log"/>
<Property Name="ESIEnable" Value="true"/>
```

```

<Property Name="ESIMaxCacheSize" Value="1024"/>
<Property Name="ESIInvalidationMonitor" Value="false"/>
<VirtualHostGroup Name="default_host">
  <VirtualHost Name="*:9080"/>
  <VirtualHost Name="*:80"/>
</VirtualHostGroup>
<ServerCluster CloneSeparatorChange="false" LoadBalance="Round Robin" Name="cluster839"
  PostSizeLimit="1000000" RemoveSpecialHeaders="true" RetryInterval="60">
  <Server CloneID="v60aig0j" ConnectTimeout="0" ExtendedHandshake="false"
    LoadBalanceWeight="2" MaxConnections="0" Name="rchas839_BASE839_BASE839"
    WaitForContinue="false">
    <Transport Hostname="rchas839.RCHLAND.IBM.COM" Port="30009" Protocol="http"/>
  </Server>
  <Server CloneID="v60aiiv5" ConnectTimeout="0" ExtendedHandshake="false"
    LoadBalanceWeight="2" MaxConnections="0" Name="rchas840_CLONE839_CLONE839"
    WaitForContinue="false">
    <Transport Hostname="rchas840.RCHLAND.IBM.COM" Port="30010" Protocol="http"/>
  </Server>
  <PrimaryServers>
    <Server Name="rchas839_BASE839_BASE839"/>
    <Server Name="rchas840_CLONE839_CLONE839"/>
  </PrimaryServers>
</ServerCluster>
<ServerCluster CloneSeparatorChange="false"
  LoadBalance="Round Robin" Name="ND839_ND839Manager_Cluster"
  PostSizeLimit="1000000" RemoveSpecialHeaders="true" RetryInterval="60">
  <Server ConnectTimeout="0" ExtendedHandshake="false"
    MaxConnections="0" Name="ND839Manager_ND839" WaitForContinue="false"/>
  <PrimaryServers>
    <Server Name="ND839Manager_ND839"/>
  </PrimaryServers>
</ServerCluster>
<UriGroup Name="default_host_cluster839_URIs">
  <Uri AffinityCookie="JSESSIONID"
    AffinityURLIdentifier="jsessionid" Name="/Darin/*"/>
  <Uri AffinityCookie="JSESSIONID"
    AffinityURLIdentifier="jsessionid" Name="/Steve/*"/>
  <Uri AffinityCookie="JSESSIONID" Name="/snoop/*"/>
</UriGroup>
<Route ServerCluster="cluster839"
  UriGroup="default_host_cluster839_URIs" VirtualHostGroup="default_host"/>
<RequestMetrics armEnabled="false" newBehavior="false"
  rmEnabled="false" traceLevel="HOPS">
  <filters enable="false" type="URI">
    <filterValues enable="false" value="/servlet/snoop"/>
    <filterValues enable="false" value="/webapp/examples/HitCount"/>
  </filters>
  <filters enable="false" type="SOURCE_IP">
    <filterValues enable="false" value="255.255.255.255"/>
    <filterValues enable="false" value="254.254.254.254"/>
  </filters>
</RequestMetrics>
</Config>

```


The tags within the file can be associated with the processing steps to help show what each is defining:

Table 5.1 Plug-in configuration XML tag descriptions

XML Tag	Description
VirtualHostGroup VirtualHost	A group of virtual host names and ports that will be specified in the HTTP Host header when the user tries to retrieve a page. Enables you to group virtual host definitions together that are configured to handle similar types of requests. The requested host and port number will be matched to a VirtualHost tag in a VirtualHostGroup.
UriGroup Uri	A group of URIs that will be specified on the HTTP request line. The incoming client URI will be compared with all the Uri tags in the UriGroup to see if there is match to determine if the application server will handle the request for the Route in conjunction with a virtual host match.
Route	<p>The Route definition is the central element of the plug-in configuration. It specifies how the plug-in will handle requests based on certain characteristics of the request. The Route definition contains the other main elements: a required ServerCluster, and either a VirtualHostGroup, UriGroup, or both.</p> <p>Using the information that is defined in the VirtualHostGroup and the UriGroup for the Route, the plug-in determines if the incoming request to the Web server should be sent on to the ServerCluster defined in this Route.</p> <p>The plug-in sets scores for Routes if there is a VirtualHost and Uri match for an incoming request. Once the plug-in processes all Routes, the Route chosen is the one with the highest score.</p>
ServerCluster Server	<p>The located ServerCluster from the Route tag contains a list of Server tags that in turn contain the requested object. The Server tag is used to check session affinity.</p> <p>The ServerCluster located by finding the correct Route can optionally specify the WLM algorithm. This will then be used to select one Server from within the ServerGroup.</p>
Transport	Once a Server has been located, its Transport tags describe how to connect to it.

TIP A detailed description of each XML tag and its relationships can be found in the WebSphere InfoCenter. To find the appropriate section, open the InfoCenter, select the Network Deployment package (drop-down box in the upper-right corner). Then search for "plugin-cfg.xml file" and select the document called "plugin-cfg.xml file". Alternatively, in the left pane of the InfoCenter select **Environment => Web servers => Configuring Web server plug-ins => Manually editing the plug-in configuration => plugin-cfg.xml file**.

HTTP Plug-in Workload Management Policies

The plug-in has two options for the load distributing algorithm:

- Round robin with weighting

- Random

The default value is Round Robin. It can be changed by editing the plugin-cfg.xml file and amend each ServerCluster tag to:

```
<ServerCluster Name="PluginCluster" LoadBalance="Random">
```

or:

```
<ServerCluster Name="PluginCluster" LoadBalance="Round Robin">
```

When the plugin-cfg.xml file is initially generated, the ServerCluster tag will not have a LoadBalance attribute defined.

Weighted round robin

When using this algorithm, the plug-in selects a cluster member at random from which to start. The first successful browser request will be routed to this cluster member and then its weight is decremented by 1. New browser requests are then sent round robin to the other application servers and subsequently the weight for each application server is decremented by 1. The spreading of the load is equal between application servers until one application server reaches a weight of 0. From then on, only application servers without a weight of 0 will be routed requests to. The only exceptions to this pattern are if the chosen cluster member cannot process the request or the browser request is associated by a session to a different cluster member.

NOTE When starting the HTTP Server, the application server weight is reduced to the lowest common denominator. For example: PluginMember1's weight is 8 and PluginMember2's weight is 6. When you start the HTTP Server, the weight of PluginMember1 will be set to 4 and the weight of PluginMember2 will be set to 3.

Using Table 5.2 on page 95 and the following explanations will show you how weighted round robin is performed. To begin with, we have a weight of 5 for PluginMember1 and a weight of 2 for PluginMember2:

- 1 When the first request comes in, PluginMember1 is randomly picked. PluginMember's weight is decremented by 1 to 4.
- 2 Second request is sent to PluginMember2. PluginMember2's weight is decremented by 1 to 1.
- 3 The third request has a cookie that specifies a server affinity to PluginMember2. The request is sent to PluginMember2 and PluginMember2's weight is decremented by 1 to 0.
- 4 The fourth request is sent to PluginMember1. PluginMember1's weight is decremented by 1 to 3.
- 5 The fifth request has a cookie that specifies a server affinity to PluginMember1. The request is sent to PluginMember1 and PluginMember1's weight is decremented by 1 to 2.
- 6 The sixth request is sent to PluginMember1. PluginMember1's weight is decremented by 1 to 1.
- 7 The seventh request is sent to PluginMember1. PluginMember1's weight is decremented by 1 to 0.
- 8 On the eighth request, the weights get reset to 5 for PluginMember1 and 2 for PluginMember2.

- 9 The next request is sent to PluginMember1. After resetting the weights, the sequence gets repeated with the same starting point (no random server selection this time).

Table 5.2 Request handling using weighted round robin server selection

Number of Request	PluginMember1 Weight	PluginMember2 Weight
0	5	2
1	4	2
2	4	1
3 (request with session affinity to PluginMember2)	4	0
4	3	0
5 (request with session affinity to PluginMember1)	2	0
6	1	0
7	0	0
8 (reset)	5	2

NOTE When there are no servers marked up having positive weights, the plug-in will change the weights of the servers by checking how many times the maximum weight should be added to make the current weight positive. A multiple of the lowest common denominator of the servers' maximum weight is added back to the current weights to make all weights positive again. For example, let us assume that the current weight of PluginMember1 is -5 because many session-based requests have been served, and PluginMember2 has a weight of 0. If the starting weights for PluginMember1 and PluginMember2 was 4 and 3, respectively, then adding 4 (the lowest common denominator) would not set PluginMember1's weight to a positive weight. Thus the plug-in decides to add the starting weights * 2, which is 8 for PluginMember1 and 6 for PluginMember2. So the new current weights are 3 for PluginMember1 (-5 + 2 * 4) and 6 for PluginMember2 (0 + 2 * 3).

Random

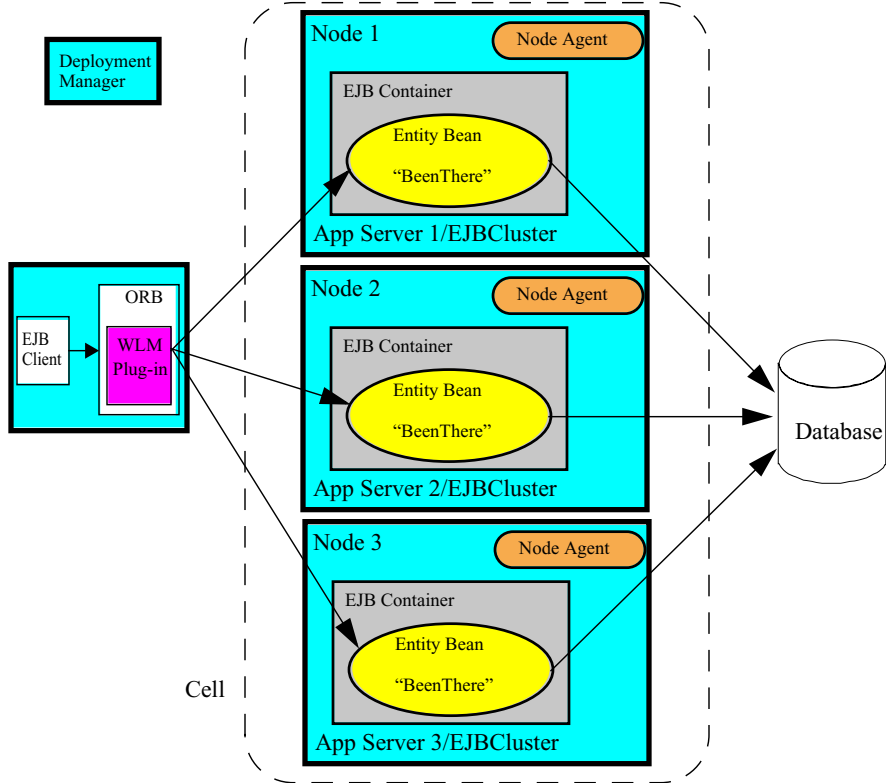
Requests are passed to cluster members randomly. Weights are not taken into account as per round robin. The only time the application servers are not chosen randomly is when there are requests with sessions associated with them. When the random setting is used, cluster member selection does not take into account where the last request was handled. This means that a new request could be handled by the same cluster member as the last request.

EJB Workload Management (WLM)

In the Network Deployment configuration, workload management for EJBs is enabled automatically when a cluster is created within a cell. There is no need for a special configuration to enable it. Workload management uses a WLM plug-in in the ORB (Object Request Broker) to dispatch the load among the application servers (cluster members) of the cluster.

Figure 5.6 on page 96 shows how EJB workload management is handled by WebSphere.

Figure 5.6 EJB workload management



The Network Deployment configuration uses the concept of cell, cluster, and cluster members to identify which application servers participate in workload management. Requests from clients are distributed among the cluster members' EJB containers within the cluster. The diagram above illustrates a clustered EJB server with three cluster members. The cluster members are App Server 1, App Server 2 and App Server 3, where each are on separate physical systems or LPAR partitions. The Java client initially bootstraps into one of the clustered application servers name spaces. During the bootstrap, the client ORB WLM plug-in is updated with the clustered EJB server information. With this information, subsequent client requests to the EJB container can participate in load balancing and failover. The WLM plug-in within the ORB is the ringmaster and provides failover amongst clustered EJB application servers.

Looking Up an EJB Home

Most applications that use JNDI run in a container. Some do not. The name used to look up an object depends on whether or not the application is running in a container.

JNDI lookup from an application running in a container

Applications that run in a container can use `java:` lookup names. Lookup names of this form provide a level of indirection such that the name used to look up an object is not dependent on the object's name since it is bound in the server's name space. The deployment descriptors for the application provide the mapping between the `java:` name and the name registered in the server's name space.

The container sets up the `java:` name space based on the deployment descriptor information so that the `java:` name is correctly mapped to the corresponding object. You can use the administrative console to change the JNDI mapping in the deployment descriptor during or after application deployment. To change a JNDI, follow these steps:

- 1 Start the administrative console.
- 2 In the topology tree, expand **Applications** and click **Enterprise Applications**.
- 3 Click the name of the application for which you want to edit enterprise bean references.
- 4 Click **Map EJB references to beans**.
- 5 In the **JNDI Name** field, specify the JNDI name of your enterprise bean. For example, to map the `BeenThere` bean, specify `Cell/clusters/EJBcluster/BeenThere`. In this example the cluster name is `EJBcluster`. On a lookup, if one of the application servers is down, then the request will be transparently redirected via the WLM plug-in to one of the other clustered application servers.
- 6 Click **Apply**.
- 7 Save the configuration.
- 8 Stop and restart the application server that contains the application client.

Example below illustrates a lookup of an EJB home:

```
// Get the initial context
import java.util.Hashtable;
import javax.naming.Context;
import javax.naming.InitialContext;
Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY,
        "com.ibm.websphere.naming.WsnInitialContextFactory");
env.put(Context.PROVIDER_URL, "corbaloc::localhost:2809");
Context initialContext = new InitialContext(env);

// Look up the home interface using the JNDI name
try {
    java.lang.Object ejbHome =
        initialContext.lookup("java:comp/env/BeenThere");
    BeenThereHome = (BeenThereHome) javax.rmi.PortableRemoteObject.narrow(
        (org.omg.CORBA.Object) ejbHome, BeenThereHome.class);
}
catch (NamingException e) { // Error getting the home interface
    ...
}
```

JNDI lookup from an application not running in a container

Applications that do not run in a container cannot use java: lookup names because it is the container that configures the java: name space for the application. Instead, an application of this type must look up the object directly from the name server. Remember, each application server contains a name server.

The example below shows the lookup of an EJB home that is running in the cluster, EJBcluster. The name can be resolved if any of the cluster members are running:

```
// Get the initial context as shown in a previous example
// Using the form of lookup name below, it doesn't matter which
// server in the cell is used to obtain the initial context.
...
// Look up the home interface using the JNDI name
try {
java.lang.Object ejbHome =
    initialContext.lookup("cell/clusters/EJBcluster/BeenThere");
beenThereHome = (BeenThereHome) javax.rmi.PortableRemoteObject.narrow(
    (org.omg.CORBA.Object) ejbHome, BeenThereHome.class);
}
catch (NamingException e) { // Error getting the home interface
...
}
```

How EJBs Participate in Workload Management

In this section, we examine how EJBs participate in workload management through the following stages:

- Initial request
- Subsequent requests
- Configuration changes

Initial Request

EJB clients of all types use EJBs as follows:

- 1 First, the client has to retrieve the initial context, as shown below:

```
import java.util.Hashtable;
import javax.naming.Context;
import javax.naming.InitialContext;
Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY,
    "com.ibm.websphere.naming.WsnInitialContextFactory");
// All servers in the provider URL below are members of the same
// cluster.
env.put(Context.PROVIDER_URL,
    "corbaloc::applw.itso.ibm.com:2809,:app2w.itso.ibm.com:2809");
Context initialContext = new InitialContext(env);
```

A J2EE application client can retrieve its initial context in a similar way using the provider bootstrap host and port configured for the client container. For example:

```
InitialContext initContext = null;
initContext = new InitialContext();
```

- 2 Next, the client needs to look up the EJB home based on the JNDI name. For example:

```
Object home = initContext.lookup("java:comp/env/BeenThere");
BeenThereHome beentherehome =
    (BeenThereHome) narrow(home, BeenThereHome.class);
```

Note that this example uses an EJB reference, `java:comp/env/BeenThere`. This EJB reference must be bound to the fully qualified JNDI name of the deployed EJB, for example: `cell/clusters/EJBcluster/BeenThere`.

- 3 The client needs to create or retrieve an EJB object, using the create method or finders of the home interface. For example:

```
BeenThere beenThere = beentherehome.create();
```

- 4 Once the EJB object has been created, you can invoke methods from the remote interface. For example:

```
Hashtable envInfo = beenThere.getRuntimeEnvInfo();
```

We will call these four steps the initial request from the EJB client. Let us see in detail what is happening from a workload management point of view, using Figure 5.6 on page 96:

- 1 The new InitialContext request goes through the ORB (Object Request Broker). This returns a JNDI context object.
- 2 The lookup on the context returns a home object of the BeenThere bean. This is an indirect IOR (Interoperable Object Reference), that is, it points to one of the Location Service Daemons (LSD), port (9090) on the Node Agent.
- 3 The first request goes to the LSD and the LSD selects one of the cluster members by using the WLM plug-in in the LSD.
- 4 The request is forwarded to the cluster member that the LSD selected.
- 5 Upon successful completion of the request, the response contains the cluster configuration information.
- 6 WLM plug-in stores the cluster configuration information and uses it for subsequent requests.

Following Requests

Now let us look at what happens with subsequent EJB requests:

- 1 Subsequent requests from the client to a remote method goes through the ORB as well.
- 2 The ORB asks the WLM plug-in for the IOR of the server in order to process the request.
- 3 Based on the workload management policy, process affinity, and transaction affinity (see “EJB Workload Management Server Selection Policy” on page 100), the WLM plug-in returns the IOR of the next target.
- 4 The ORB invokes the remote method on the selected server.

Configuration Changes

To conclude this section, we examine how the WLM plug-in is informed of changes to the cluster configuration:

- 1 Changes are made to the cluster configuration, such as adding or starting a third cluster member, in our example.
- 2 The Deployment Manager pushes the changes to all cluster members in the cluster.
- 3 Meanwhile, the EJB client is still performing requests on the cluster.
- 4 With each request for a remote component, information about the cluster is returned in the response from the cluster member.

With each client request for a remote component, information about the cluster is returned in the response from the cluster member, if the cluster has been modified since the last request from this client. The WLM plug-in will update its cluster data using the new data returned in the response.
- 5 The EJB client makes another method call to the remote interface.
- 6 The ORB that handles the request asks the WLM plug-in for the IOR of the server to contact.
- 7 The WLM plug-in returns the IOR of the next target, based on the workload management policy, process affinity, and transaction affinity (see “EJB Workload Management Server Selection Policy” on page 100), and the request can be processed by the ORB.

It is important to note that a change in the selection policy does not cause the cluster information to be sent to a client in response to a request. The WLM plug-in will continue using the selection policy defined at the first request. If the cluster topology or the number of cluster members started is changed, the WLM plug-in will get the new selection policy as part of the new cluster configuration in the response.

EJB Workload Management Server Selection Policy

An EJB server selection policy defines how clients (such as servlets, stand-alone Java clients, or other EJBs) choose among EJB cluster members (instances). EJB workload management offers the following selection policies:

Server weighted round robin routing

The server weighted round robin routing will select the next currently available cluster member. The policy will ensure a distribution based on the set of server weights that have been assigned to the members of a cluster. For example, if all servers in the cluster have the same weight, the expected distribution for the cluster would be that all servers receive the same number of requests. If the weights for the servers are not equal, the distribution mechanism will send more requests to the higher weight value servers than the lower weight value servers. The policy ensures the desired distribution, based on the weights assigned to the cluster members.

The server weight value defaults to 2 for each member of the cluster and is maintained in the cluster routing table.

When setting the server weight values for your cluster members, you should utilize low values to avoid load variations. For example, you would be better off by setting server weights to 2 and 5 versus 8 and 20 so the refresh will occur more often and thus the server with the weight of 8 won't have to sit idle while 12 requests go to the server with

a weight of 20. It would only sit idle for three requests instead of 12 requests. Valid values for the weights range from 0 to 20.

If a particular EJB server instance is stopped or otherwise unavailable, that instance is skipped (no attempt is made to select it) until it can be verified as being back in service.

The ORB in the EJB client has a routing table for each cluster. The routing table is recalculated for every new request that comes in. There are no additional requests to an application server once its outstanding request ratio has reached its server weight value, but there are exceptions to this rule:

- Transaction affinity

When several requests to EJB methods occur in the scope of the same transaction (a user-managed transaction or a container-managed transaction), all requests will go to the same cluster member, if possible. As soon as the WLM plug-in notices that a transaction is started, it will stop dispatching the requests to different cluster members. All requests within the scope of this transaction are sent to the same cluster member.

- Process affinity

Whatever the workload management server selection policy, if an EJB is available in the same cluster member as the client, all the requests coming from the client will be directed to this EJB. This is called process affinity, because all the requests are in-process requests. The advantage of process affinity is that there is no need for serialization for method calls. Parameters can be passed by value without any serialization costs, since all method calls are performed within the same Java virtual machine, in the same memory space.

To take advantage of the process affinity, the client can only be a servlet or an EJB (stand-alone Java clients' EJB requests come from outside of the application server process, and thus are not affected by process affinity). In the case of a servlet, process affinity is only possible if the Web container running the servlet is in the same application server as the EJB container. In the case of an EJB (e.g. a stateless session bean acting as a facade), process affinity occurs when the called EJB is in the same EJB container as the calling EJB.

Prefer local (selected by default)

Along with the server weighted round robin routing, there is also a Prefer local policy.

Once the Prefer local policy is turned on, it is used for every cluster member in your cluster. Similarly, when you turn it off, it is off for every cluster member in your cluster.

With the Prefer local policy, the selection made by the WLM plug-in not only depends on the running cluster members, but also on the node (i.e. machine) where the request comes from. The WLM plug-in will only select cluster members on the same node as the client, unless all local cluster members are unavailable.

The advantage of the Prefer local policy is that there are no network communications between the client and the EJB, so depending on the chosen topology, this policy can provide improved performance.

The client is the Java virtual machine (JVM) in which the code calling the EJB is running. This client might be a WebSphere process such as an application server running a Web container, or an application server running an EJB container.

When a servlet calls an EJB, the client is the application server that contains the servlet, and if the Prefer local policy is selected, the requests will go to the EJB container

running on the same system. If EJB1 is calling EJB2, the client is the application that is running the EJB container containing EJB1. With the Prefer local policy, the requests will go to the same EJB container if EJB2 can be found (see 6.6.3, "Process affinity" on page 227), or to another EJB container running on the same system.

This client might also be a Java virtual machine not running in WebSphere, such as a J2EE client application, or a stand-alone Java program accessing EJBs. For a J2EE client application, the Prefer local policy has no influence on the request distribution because the client runs on a remote workstation.

In the case of a Java program running on the same machine as WebSphere and using the WebSphere JRE and its ORB, the Prefer local policies will dispatch requests among EJB containers running on the same machine. EJB WLM requires the WebSphere ORB and its WLM plug-in. If non-WebSphere ORBs are used, then the Java client will not be able to participate in EJB WLM.

IMPORTANT Whatever the server selection policy, process affinity and transaction affinity will always override the selection policy.

If there is no cluster member available on the local system (because of a failure, or because of the topology), the request will be dispatched to available cluster members following the server weighted round robin routing policy.

The Prefer local option is based on topology and pre-production test results.

Naturally, the local host call will be quicker, and if you can put your clients (usually Web containers) on the same machines as your servers, the Prefer local option is a good choice. If you have clients only on a subset of your machines, then you will need to analyze the load distribution, since you will have client requests coming from remote machines as well as from the local machine.

Session Management

One of the other functions that the plug-in provides, in addition to failover and workload management, is the ability to manage session affinity. *Session affinity* is when the load-distribution facility recognizes the existence of a session and attempts to direct all requests within that session to the same server.

HTTP sessions allow applications to maintain state information across multiple user visits. The Java servlet specification provides a mechanism for applications to maintain all session information at the server and exchange minimal information with a user's browser.

The methods used to manage the sessions are very important in a workload-managed environment. There are different methods of identifying, maintaining affinity, and persisting the sessions.

Session Affinity

In the Servlet 2.3 specification, as implemented by WebSphere Application Server V5.0, only a single cluster member may control/access a given session at a time. After a session has been created, all requests need to go to the application server that created the session. This session affinity is provided by the plug-in.

In a clustered environment, there will be more than one application server that can serve the client request. The plug-in needs to read a request and be able to identify which cluster member should handle it. Session identifiers are used to do this.

The session identifier basically provides two vital pieces of information which allow the plug-in to route to the proper cluster member. The following is an example of a session identifier:

```
JSESSIONID=0000U0ES24K2PYG153E14CN1W5I:u87u15ed
```

The `JSESSIONID` cookie is divided into four parts:

Table 5.3 Parts of a `JSESSIONID` cookie

Content	Value used in example above
Cache ID	0000
Session ID	U0ES24K2PYG153E14CN1W5I
Separator	:
Clone ID (cluster member)	u87u15ed

The clone ID identifies the application server where the session object is cached. The plug-in file generated by the Network Deployment Manager would also have the same clone ID.

```
<ServerCluster Name="PluginCluster" LoadBalance="Round Robin">  
<Server CloneID="u87u15ed" LoadBalanceWeight="8" Name="PluginMember1">  
<Transport Hostname="app1.itso.ibm.com" Port="9084" Protocol="http"/>  
<Transport Hostname="app1.itso.ibm.com" Port="9443" Protocol="https">
```

With this information the plug-in can route this request back to the proper cluster member.

Session Persistence

Many Web applications use the simplest form of session management, the in-memory local session cache. The local session cache keeps session information in memory and local to the machine and WebSphere Application Server where the session information was first created. Local session management does not share user session information with other clustered machines. Users only obtain their session information if they return to the machine and WebSphere Application Server holds their session information on subsequent accesses to the Web site.

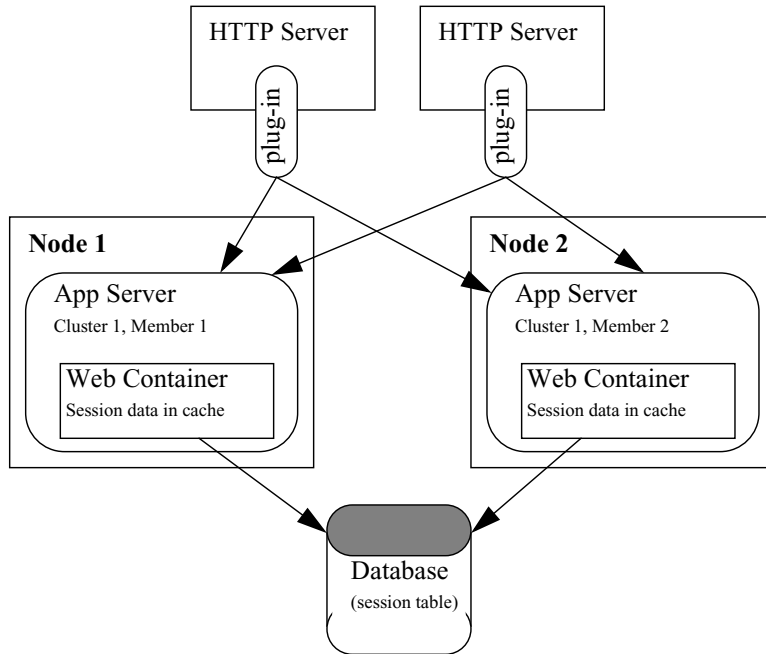
Most importantly, local session management lacks a persistent store for the sessions it manages. A server failure takes down not only the WebSphere instances running on the server, but also destroys any sessions managed by those instances.

By default, WebSphere places session objects in memory. However, with the Network Deployment configuration, the administrator has the option of enabling persistent session management. This instructs WebSphere to place session objects in a persistent store. Using a persistent store allows the user session data to be recovered by another cluster member after a cluster member in a cluster fails or is shut down. Two options for session persistence are available:

- Database

In a multi-server environment, session information can be stored in a central session database for session persistence. The multiple application servers hosting a particular application need to share this database information in order to maintain session states for the stateful components.

Figure 5.7 Database session persistence - session affinity assured by plug-in



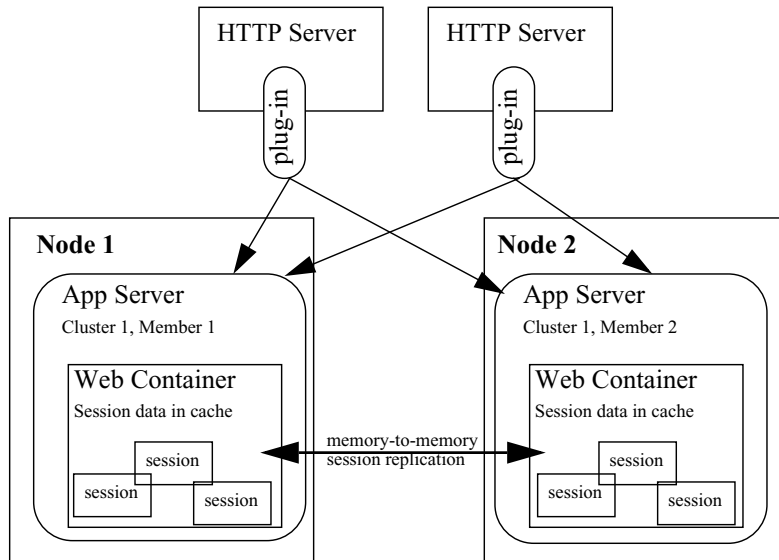
- Memory-to-memory using WebSphere internal messaging

WebSphere internal messaging enables sessions to be shared among application servers without using a database. Using this method, sessions are stored in the memory of an application server, providing the same functionality as a database for session persistence. Two topology options are offered: peer-to-peer and client/server.

In the peer-to-peer topology, each application server stores sessions in its own memory. It also stores sessions to and retrieves sessions from other application servers. In other words, each application server acts as a client by retrieving sessions from other application servers, and each application server acts as a server by providing sessions to other application servers.

In the client/server topology, application servers act as either a replication client or a server. Those that act as replication servers store sessions in their own memory and provide session information to clients. They are dedicated replication servers that just store sessions but do not respond to the users' requests. Client application servers send session information to the replication servers and retrieve sessions from the servers. They respond to user requests and store only the sessions of the users with whom they interact.

Figure 5.8 In-memory session persistence - session affinity assured by plug-in



Session Identifiers

There are three methods of identifying a user's session to the application server. They are:

- SSL ID
- Cookies
- URL rewriting

SSL session ID

To use the SSL Session ID as the session modifier, clients need to be using an SSL connection to the Web server. This connection does not need to use client certificate authentication, but simply a normal server authentication connection. This can be enabled by turning on SSL support in the Web server.

The session ID is generated from the SSL session information. This is passed to the Web server and then passed on to the plug-in. If more than one Web server is being used, then affinity must be maintained to the correct Web server, since the session ID is defined by the connection between browser and Web server.

Connecting to another Web server will reset the SSL connection and a new session ID will be generated.

It is possible to maintain the SSL session affinity using the Load Balancer from IBM WebSphere Edge Components. See “SSL Session ID Affinity” on page 141 for details.

SSL session ID cannot be used on its own in a clustered environment. It is not possible to add the cluster member ID to the end of the SSL session information, so another mechanism must be used. Either cookies or URL rewriting needs to be enabled to provide this function. The cookie or rewritten URL then contains session affinity information that enables the Web server to properly route requests back to the same

server once the HTTP session has been created on a server. If cookies or URL rewriting are not enabled, then a session will be created but there will be no mechanism to return the user to the correct cluster member at their next request.

The format of the cookie or URL rewrite is shown below:

```
SSLJSESSION=0000SESSIONMANAGEMENTAFFINI:u87u15ed
```

With SSL, the session timeout is not controlled by the application server. The session timeout delay is governed by the Web server and the Web browser. The lifetime of an SSL session ID can be controlled by configuration options in the Web server.

When to use SSL session IDs

When using a clustered environment, SSL ID requires a lot of overhead to set up the infrastructure. There is also a single point of failure for each session: the Web server. If the Web server goes down, the user will lose the session ID and therefore access to session information.

SSL ID will also be slower than other mechanisms, since the browser has to communicate using HTTPS and not HTTP. The inconsistency of browsers and their SSL handling could also affect how the application performs. However, SSL provides a more secure mechanism of user identification for session information. The session identifier is difficult to copy.

If your Web site requires the highest security, then use SSL ID, but be aware that it comes with the overheads and deficiencies mentioned above. Consider using standard cookies over an SSL session instead.

Cookies

Cookies are more straightforward than the SSL session ID. When this option is selected, the plug-in will use the JSESSIONID to manage requests. This name is required by the Servlet 2.3 specification. The session management tracking mechanism can be performed at the application server, Web container, or the Web module level.

To use the cookie JSESSIONID, users must have their browsers set up to allow cookies.

None of the workload management issues we discussed with SSL IDs applies to cookies. The browser can be connected to any Web server and there will be no effect on the session. Cookie session identifiers will survive a Web server crash and, provided persistent sessions are enabled, will also survive unavailability of the application server.

The session lifetime is governed by the cookie lifespan. By default, WebSphere defines its cookies to last until the browser is closed. It is also possible to define the maximum age of the cookie in the cookies configuration.

When to use cookies

Cookies are the most common method of tracking the session. They work well in a workload-managed environment, with ease-of-use advantages over SSL IDs.

They do not require additional application coding and can be used from static HTML links, unlike URL rewriting.

The fact that any user can turn off cookie support in his/her browser could be more important to your application. If this is the case, then one of the other options must be considered.

If security is important, then it is possible to use cookies in an SSL environment and not have the overhead of setting up SSL session ID management.

URL rewriting

URL rewriting (or URL encoding) is a useful mechanism that does not require users to enable cookies in their browsers, and yet still allows WebSphere to manage sessions.

The process of setting up URL rewriting is not transparent to the Web application. It requires a developer to include specific commands to append the session information to the end of any HTTP link that will be used from the Web browser. The plug-in will search for any URL encoded session information on incoming requests and route them accordingly.

Rewriting the URLs can only be done on dynamic HTML that is generated within WebSphere, for example the output from JSPs or servlets. Session information will be lost if static HTML links are accessed, restricting the flow of site pages to dynamic pages only. From the first page, the user receives a session ID, and the Web site must continue using dynamic pages until the completion of the session.

There are no specific issues with using URL encoding in a workload-managed environment.

When to use URL rewriting

The only situation in which URL encoding excels over the other options is when users have not enabled cookies in their browsers.

Because it is possible to select more than one mechanism to pass session IDs, it is also possible to compensate for users not using cookies. URL encoding could be enabled and then used as a fallback mechanism if the users are not accepting cookies.

Web Services

Web services are self-contained, modular applications that can be described, published, located, and invoked over a network. WebSphere Application Server can act as both a Web service provider and as a requester. As a provider, it hosts Web services that are published for use by clients. As a requester, it hosts applications that invoke Web services from other locations.

WebSphere Application Server supports SOAP-based Web service hosting and invocation.

The Web services support includes the following:

- Web Services Description Language (WSDL), an XML-based description language that provides a way to catalog and describe services. It describes the interface of Web services (parameters and result), the binding (SOAP, EJB), and the implementation location.
- Universal Discovery Description and Integration (UDDI), a global, platform-independent, open framework to enable businesses to discover each other, define their interaction, and share information in a global registry.
- Simple Object Access Protocol (SOAP), a lightweight protocol for exchange of information in a decentralized, distributed environment.
- eXtensible Markup Language (XML), which provides a common language for exchanging information.
- Web Services Invocation Framework (WSIF), a runtime architecture for service-oriented applications that provides a binding-independent mechanism for Web service invocation. WSIF enables easy encapsulation of back-end systems connected through the J2EE Connector Architecture (JCA) as Web services.

- JAX-RPC (JSR 101) provides the core programming model and bindings for developing and deploying Web services on the Java platform. It is a Java API for XML-based RPC and supports only JavaBeans as Web Service provider.
- Enterprise Web services (JSR 109): this adds EJBs and XML deployment descriptors to JSR 101.
- WS-Security: this specification covers a standard set of SOAP extensions that can be used when building secure Web services to provide integrity and confidentiality. It is designed to be open to other security models including PKI, Kerberos, and SSL. WS-Security provides support for multiple security tokens, multiple signature formats, multiple trust domains, and multiple encryption technologies. It includes security token propagation, message integrity, and message confidentiality. The specification is proposed by IBM, Microsoft®, and VeriSign for review and evaluation. In the future, it will replace existing Web services security specifications from IBM and Microsoft including SOAP Security Extensions (SOAP-SEC), Microsoft's WS-Security and WS-License, as well as IBM's security token and encryption documents.

Web Service Client (requester)

Applications that invoke Web services are known as Web service clients or Web service requestors. An application that acts as a Web service client is deployed to WebSphere Application Server like any other enterprise application. No additional configuration or software is needed for the Web services client to function. Web services clients can also be stand-alone applications.

A Web service client will bind to a Web service server to invoke the Web service. The binding is done using a service proxy (or *stub*), which is generated based on the WSDL document for the Web service. This service proxy contains all the needed information to invoke the Web service and is used locally by the clients to access the business service. The binding can also be done dynamically using WSIF.

Web Service Provider

An application that acts as a Web service is deployed to WebSphere Application Server like any other enterprise application. The Web services are contained in Web modules or EJB modules.

Publishing the Web service to a UDDI registry makes it available to anyone searching for it. Web services can be published to a UDDI registry using the Web Services Explorer provided with WebSphere Studio Application Developer, Integration Edition, and Enterprise Developer configurations.

When using WebSphere Studio V5 to package the application for deployment, no additional configuration or software is needed for the Web services client to function. The SOAP servlets are automatically added and a SOAP admin tool is included in a Web module.

If not, you will need to use `soapearenabler.bat`, found in the WebSphere `bin` directory to enable the SOAP services within the EAR file and add the SOAP admin tool.

IBM WebSphere UDDI Registry

WebSphere Application Server V5 provides a private UDDI registry that implements V2.0 of the UDDI specification. This enables the enterprise to run its own Web services broker within the company or provide brokering services to the outside world.

The UDDI registry is installed as a Web application by the administrator on one application server (or cluster). A DB2 database is used to store registry data.

There are three interfaces to the registry for inquiry and publish:

- Through the UDDI SOAP API.
- Through the UDDI EJB client interface.
- Through the UDDI user console. This Web-based GUI interface can be used to publish and to inquire about UDDI entities but only provides a subset of the UDDI API functions.

Security for the UDDI registry is handled using WebSphere security. To support the use of secure access with the IBM WebSphere UDDI Registry, you need to configure WebSphere to use HTTPS and SSL.

Web Service Gateway

The Web Services Gateway bridges the gap between Internet and intranet environments during Web service invocations. The gateway builds upon the Web services Definition Language (WSDL) and the Web Services Invocation Framework (WSIF) for deployment and invocation.

The primary function of the Web Services Gateway is to map an existing WSDL-defined Web service to a new service that is offered by the gateway to others. The gateway thus acts as a proxy. External services are imported into the gateway and made available to the enterprise as internal proxy services. Likewise, internal services are imported into the gateway and made available as external proxy services. These services are also published to the relevant UDDI registries where required.

The Web Services Gateway is installed as Web application by the administrator.

To expose an internal service for outside consumption, the gateway takes the WSDL file for the existing service and generates a new WSDL file that can be shared with outside requestors. The interface described in the WSDL is exactly the same, but the service endpoint is changed to the gateway, which is now the official endpoint for the service client.

The gateway facilitates working with UDDI registries. As you map a service for external consumption using the gateway, you can publish the exported WSDL in the UDDI registry. When the services in the gateway are modified, the UDDI registry is updated with the latest changes.

Resource Providers

Resource providers define resources needed by running J2EE applications. The aspects of the WebSphere Application Server resource provider support that we are interested in are the JCA resource adapters, JDBC resources, and JMS providers.

JCA Resources Adapters

The J2EE Connector Architecture (JCA) defines a standard architecture for connecting the J2EE platform to heterogeneous Enterprise Information Systems (EIS) such as ERP, database systems, and legacy applications not written in the Java programming language.

The JCA resource adapter is a system-level software driver supplied by EIS vendors or other third-party vendors. It provides the connectivity between J2EE components (an application server or an application client) and an EIS.

To use a resource adapter, you need to install the resource adapter code and create connection factories that use the adapter.

One resource adapter, the WebSphere Relational Resource Adapter, is predefined for handling data access to relational databases. This resource adapter provides data access through JDBC calls to access databases dynamically. It provides connection pooling, local transaction, and security support. The WebSphere persistence manager uses this adapter to access data for container-managed persistence (CMP) beans.

JDBC Resources

A data source represents a real-world data source, such as a relational database. When a data source object has been registered with a JNDI naming service, an application can retrieve it from the naming service and use it to make a connection to the data source it represents.

Information about the data source and how to locate it, such as its name, the server on which it resides, its port number, and so on, is stored in the form of properties on the DataSource object. This makes an application more portable because it does not need to hard code a driver name, which often includes the name of a particular vendor. It also makes maintaining the code easier because if, for example, the data source is moved to a different server, all that needs to be done is to update the relevant property in the data source. None of the code using that data source needs to be touched.

Once a data source has been registered with an application server's JNDI name space, application programmers can use it to make a connection to the data source it represents.

The connection will usually be a pooled connection, that is, once the application closes the connection, the connection is returned to a connection pool, rather than being destroyed.

Data source classes and JDBC drivers are implemented by the data source vendor. By configuring a JDBC provider, we are providing information about the set of classes used to implement the data source and the database driver, that is, it provides the environment settings for the DataSource object.

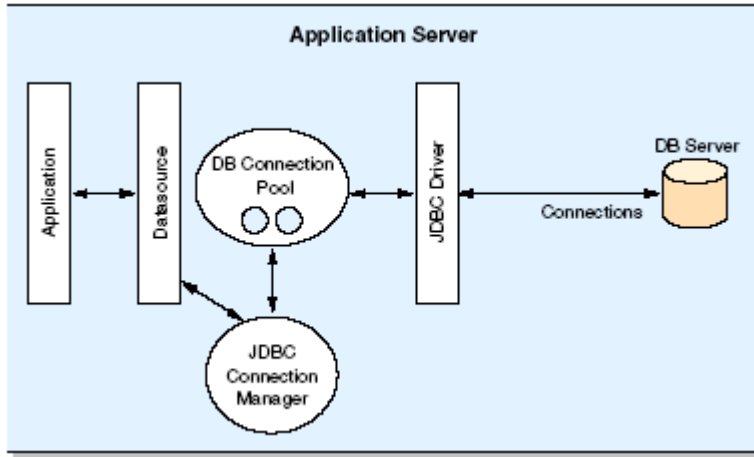
WebSphere provides two types of data sources, each differentiated by how the connections are handled:

- WebSphere Version 4 data source
- WebSphere Version 5 data source

Version 4 data source

WebSphere Version 4.0 provides its own JDBC connection manager to handle connection pooling and JDBC access. This support is included with WebSphere Application Server V5 to provide support for J2EE 1.2 applications. If an application chooses to use a Version 4 data source, the application will have the same connection behavior as in WebSphere Version 4.

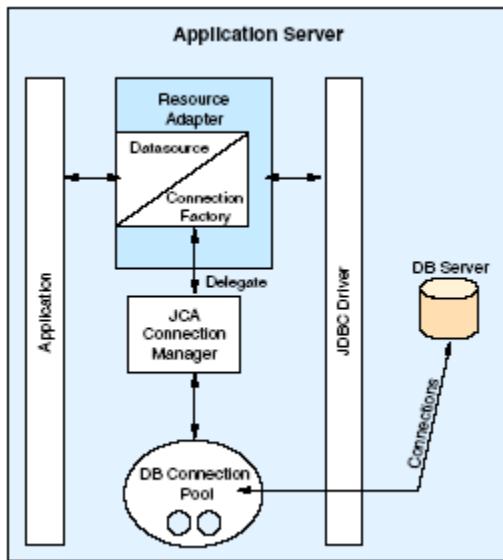
Figure 5.9 WebSphere V4 data source architecture



WebSphere 5 data source

In WebSphere Application Server V5, connection pooling is provided by two parts, a JCA Connection Manager and a relational resource adapter.

Figure 5.10 WebSphere V5 data source architecture



The JCA Connection Manager provides the connection pooling, local transaction, and security supports. The relational resource adapter provides both JDBC wrappers and JCA CCI implementation that allows BMP, JDBC applications and CMP beans to access the database.

JMS Providers

The JMS functionality provided by WebSphere includes support for three types of JMS provider:

- WebSphere JMS provider (embedded messaging)
- WebSphere MQ JMS provider
- Generic JMS providers

There can be more than one JMS provider per node. That is, a node can be configured to concurrently make use of any combination (or all) of the WebSphere JMS provider, WebSphere MQ JMS provider and a generic JMS provider.

There can only be one JMS server process (embedded WebSphere JMS provider) per node.

WebSphere JMS provider (embedded messaging)

The embedded messaging support is provided as a WebSphere Application Server installation option. The embedded support provides both point-to-point and publish/subscribe functions.

The JMS server hosts the broker and manages the external WebSphere MQ processes, including creation, management and deletion of the WebSphere MQ queues and topics.

WebSphere MQ JMS provider

WebSphere Application Server V5 supports the use of full WebSphere MQ as the JMS provider. The product is tightly integrated with the WebSphere installation, with WebSphere providing the JMS client classes and administration interface, while WebSphere MQ provides the queue-based messaging system.

WebSphere MQ JMS provider only supports the point-to-point function.

Generic JMS providers

WebSphere Application Server V5 supports the use of generic JMS providers, as long as they implement the ASF component of the JMS 1.0.2 specification. JMS resources for generic JMS providers are not configurable using WebSphere administration.

Security

IBM WebSphere Application Server security sits on top of the operating system security and the security features provided by other components, including the Java language.

- Operating system security protects sensitive WebSphere configuration files and authenticates users when the operating system user registry is used for authentication.
- Standard Java security is provided through the Java Virtual Machine (JVM) used by WebSphere and the Java security classes.
- The Java 2 Security API provides a means to enforce access control, based on the location of the code and who signed it. Java 2 Security guards access to system resources such as file I/O, sockets, and properties. WebSphere global security settings allow you to enable or disable Java 2 security and provide a default set of policies. Java 2 security can be activated or inactivated independently from

WebSphere global security; however, when global security is enabled, by default Java 2 Security is also enabled.

The current principal of the thread of execution is not considered in the Java 2 Security authorization. There are instances where it is useful for the authorization to be based on the principal, rather the code base and the signer. The Java Authentication and Authorization Services (JAAS) is a standard Java API that allows the Java 2 Security authorization to be extended to the code base on the principal as well as the code base and signers.

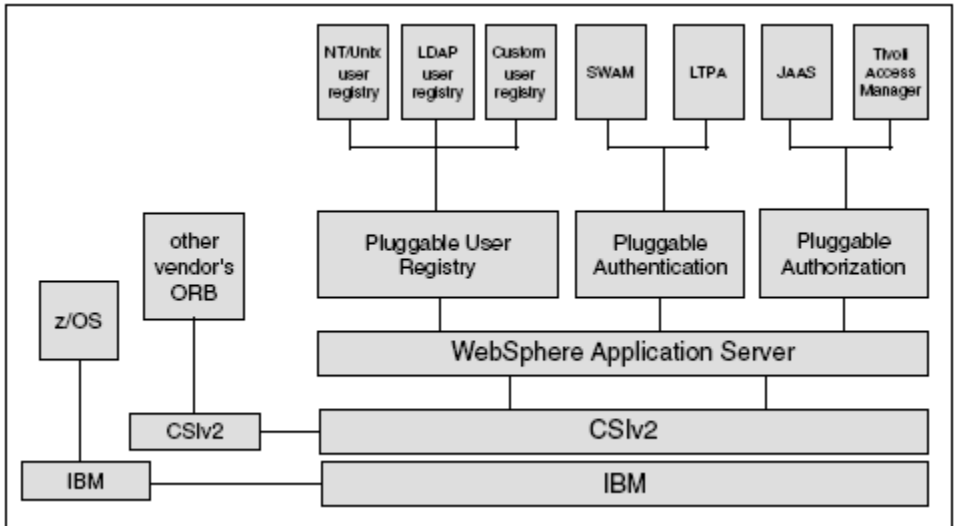
JAAS is a standard extension to the Java 2 SDK v1.3 and is part of the Java 2 SDK V1.4. The JAAS programming model allows the developer to design application authentication in a pluggable fashion, which makes the application independent from the underlying authentication technology. JAAS does not require Java 2 security to be enabled.

- The Common Secure Interoperability protocol adds additional security features that enable interoperable authentication, delegation and privileges in a CORBA environment. It supports interoperability with the EJB 2.0 specification and can be used with SSL.
- J2EE security uses the security collaborator to enforce J2EE-based security policies and support J2EE security APIs. APIs are accessed from WebSphere applications in order to access security mechanisms and implement security policies. J2EE security guards access to Web resources such as servlets/JSPs and EJB methods based on roles defined by the application developer. Users and groups are assigned to these roles during application deployment.
- IBM Java Secure Socket Extension (JSEE) is the Secure Sockets Layer (SSL) implementation used by WebSphere Application Server. It is a set of Java packages that enable secure Internet communications. It implements a Java version of SSL and Transport Layer Security (TLS) protocols and includes functionality for data encryption, server authentication, message integrity, and client authentication.

WebSphere Application Server V5 security relies on and enhances all the above mentioned layers. It implements security policy in a unified manner for both Web and EJB resources.

Figure 5.11 on 114 presents a general view of the logical layered security architecture model of WebSphere Application Server V5.0. The flexibility of that architecture model lies in pluggable modules that can be configured according to the requirements and existing IT resources.

Figure 5.11 WebSphere V5 security architecture model



User Registry

The pluggable user registry allows you to configure different databases to store user IDs and passwords that are used for authentication and authorization. There are three options:

- Local operating system user registry

When configured, WebSphere uses the operating system's users and groups for authentication.

- LDAP user registry

In many solutions, LDAP user registry is recommended as the best solution for large scale Web implementations. Most of the LDAP servers available on the market are well equipped with security mechanisms that can be used to securely communicate with WebSphere Application Server. The flexibility of search parameters that an administrator can set up to adapt WebSphere to different LDAP schemas is considerable.

- Custom user registry

This leaves an open door for any custom implementation of a user registry database. WebSphere API provides the UserRegistry Java interface that you should use to write the custom registry. This interface may be used to access virtually any relational database, flat files and so on.

Only one single registry can be active at a time.

Authentication

Authentication is the process of establishing whether a client is valid in a particular context. A client can be an end user, a machine, or an application. The pluggable

authentication module allows you to choose whether WebSphere will authenticate the user or will accept the credentials from external authentication mechanisms.

An authentication mechanism in WebSphere typically collaborates closely with a user registry when performing authentication. The authentication mechanism is responsible for creating a credential which is a WebSphere internal representation of a successfully authenticated client user. Not all credentials are created equal. The abilities of the credential are determined by the configured authentication mechanism.

Although WebSphere provides several authentication mechanisms, only a single “active” authentication mechanism can be configured at once. The active authentication mechanism is selected when configuring WebSphere global security.

WebSphere provides two authentication mechanisms: Simple WebSphere Authentication Mechanism (SWAM) and Lightweight Third Party Authentication (LTPA). These two authentication mechanisms differ primarily in the distributed security features each supports.

- SWAM (Simple WebSphere Authentication Mechanism)

The SWAM authentication mechanism is intended for simple, non-distributed, single application server type runtime environments. The single application server restriction is due to the fact that SWAM does not support forwardable credentials. What this means is that if a servlet or EJB in application server process 1 invokes a remote method on an EJB living in another application server process 2, the identity of the caller identity in process 1 is not transmitted to server process 2. What is transmitted is an unauthenticated credential, which, depending on the security permissions configured on the EJB methods, may cause authorization failures.

Since SWAM is intended for a single application server process, single sign-on (SSO) is not supported.

The SWAM authentication mechanism is suitable for simple environments, software development environments, or other environments that do not require a distributed security solution.

SWAM relies on the session ID; it is not as secure as LTPA, therefore using SSL with SWAM is strongly recommended.

- LTPA (Light Weight Third Party Authentication)

Lightweight Third Party Authentication (LTPA) is intended for distributed, multiple application servers and machine environments. It supports forwardable credentials and SSO. LTPA is able to support security in a distributed environment through the use of cryptography. This allows LTPA to encrypt, digitally sign and securely transmit authentication related data and later decrypt and verify the signature.

LTPA requires that the configured user registry be a central shared repository such as an LDAP registry.

Authorization

Pluggable authorization interfaces will allow the use of different authorization mechanisms for WebSphere applications. In the current version, JAAS is supported and Tivoli Access Manager is an external authorization system.

WebSphere Application Server standard authorization mechanisms are based on the J2EE security specification and Java Authentication and Authorization Services (JAAS).

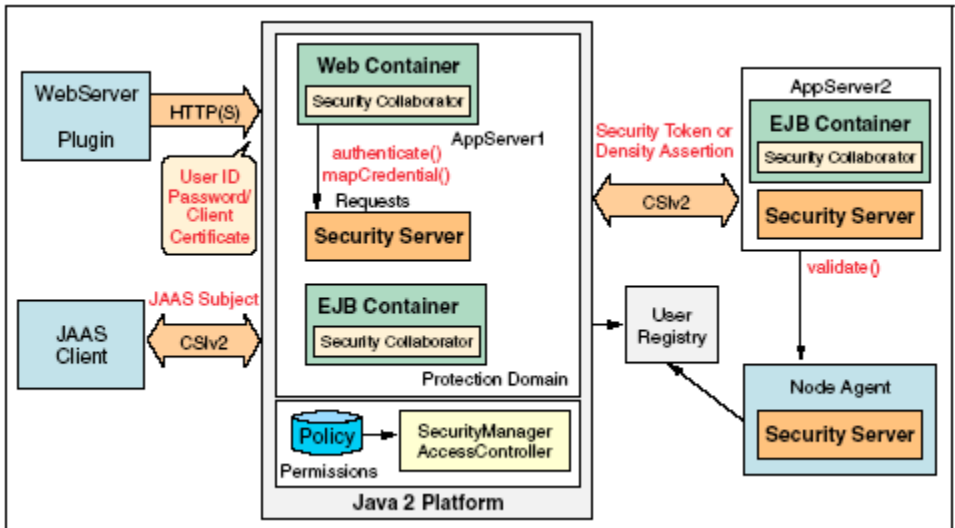
- Java 2 security architecture uses a security policy to specify who is allowed to execute code in the application. Code characteristics, like a code signature, signer ID, or source server, determine whether or not the code will be granted access to be executed.
- J2EE extends this approach with role-based access control. Permission to execute code is granted not only based on the code characteristics (Java 2 security) but also on the user, who is running it. J2EE programming models allow the developer to design application authentication in a pluggable fashion, which makes the application independent from the underlying authentication technology.

For each authenticated user, a `Subject` class is created and a set of `Principals` is included in the subject in order to identify that user. Security policies are granted based on possessed principals.

Security Components

Figure 5.12 shows an overview of the security components that come into play in WebSphere Application Security.

Figure 5.12 WebSphere V5 security components



Security server

The security server is a component of WebSphere Application Server that runs in each application server process. If multiple application server instances are executed on a single node, then multiple security servers exist on that node.

The security server component is responsible for managing authentication and for collaborating with the authorization engine and the user registry.

Security collaborators

Security collaborators are application server processes responsible for enforcing security constraints specified in deployment descriptors. They communicate with the security server every time that authentication and authorization actions are required. The following security collaborators are identified.

- Web security collaborator

The Web security collaborator resides in the Web container and provides the following services to the application:

- Checks authentication
- Performs authorization according to the constraint specified in the deployment descriptor
- Logs security tracing information

- EJB security collaborator

The EJB security collaborator resides in the EJB container. It uses CSIv2 and SAS to authenticate Java client requests to enterprise beans. It works with the security server to perform the following functions:

- Check authorizations according to the specified security constraint
- Support communication with local user registry
- Log security tracing information
- Communicate external ORB using CSIv2 when a request for a remote bean is issued

Security Flows

The following sections outline the general security flow.

Web browser communication

The following steps describe the interaction of the components from a security point of view when a Web browser sends a request to a WebSphere application.

- 1 The Web user requests a Web resource protected by WebSphere application server.
- 2 The Web server receives the request and recognizes that the requested resource is on the application server, and, using the WebSphere plug-in, redirects the request.
- 3 The WebSphere plug-in passes the user credentials to the Web security collaborator, which performs user authentication.
- 4 After successful authentication, the Web request reaches the Web container. The Web security collaborator passes the user's credentials and the security information contained in the deployment descriptor to the security server for authorization.
- 5 Upon subsequent requests, authorization checks are performed either by the Web collaborator or the EJB collaborator, depending on what the user is requesting. User credentials are extracted from the established security context.

Java client communication

The steps below describe how a Java client interacts with a WebSphere application.

- 1 A Java client generates a request that reaches the server side ORB.

- 2 The CSIv2 or IBM SAS interceptor performs authentication on the server side on behalf of the ORB, and sets the security context.
- 3 The server side ORB passes the request to the EJB container.
- 4 After submitting a request to access the protected EJB method, the EJB container passes the request to the EJB collaborator.
- 5 The EJB collaborator reads the deployment descriptor from the .ear file and user credential from the security context.
- 6 Credentials and security information are passed to the security server which validates user access rights and passes this information back to the collaborator.
- 7 After receiving a response from the security server, the EJB collaborator authorizes or denies access to the user to the requested resource.

Administration

IBM WebSphere Application Server V5 provides a new administration model based on the JMX framework. JMX allows you to wrap hardware and software resources in Java and expose them in a distributed environment. JMX also provides a mapping framework for integrating existing management protocols, such as SNMP, into JMX's own management structures.

Each application server has an administration service that provides the necessary functions to manipulate configuration data for the server and its components. The configuration is stored in a repository. The repository is actually a set of XML files stored in the server's file system.

Administrative Tools

The administrative tools that are provided by the Network Deployment configuration include the administrative console, commands, and wsadmin.

Administrative console

The administrative console is a Web-browser based interface that provides configuration and operation capability. It is implemented with an enterprise application (`adminconsole.ear`) installed on an application server. The administrator connects to the application using a Web browser client.

Users assigned to different administration roles can manage the application server and certain components and services using this interface. Because the administrative console is an application, the server and application must both be started before you can use it.

In the Network Deployment configuration, the administrative console application is installed and runs on the Deployment Manager. When a node is added to a cell, the `adminconsole` application is deleted from the node and the configuration files are integrated into the master cell repository to be maintained by the Deployment Manager.

Commands

WebSphere Application Server provides a set of commands in the `<server_install>/bin` directory that allow you to perform a subset of administrative functions.

For example, the `startServer` command is provided to start an application server.

Scripting client - wsadmin

The wsadmin scripting client provides extra flexibility over the Web-based administration application, allowing administration using the command-line interface. Using the scripting client not only makes administration quicker, but helps automate the administration of multiple application servers and nodes using scripts.

The scripting client uses the Bean Scripting Framework (BSF), which allows a variety of scripting languages to be used for configuration and control.

The wsadmin scripting interface is included in all WebSphere Application Server configurations but is targeted toward advanced users. The use of wsadmin requires in-depth familiarity with application server architecture and a scripting language.

Configuration Repository

The configuration repository holds copies of the individual component configuration documents. Unlike previous versions of WebSphere Application Server, which used a relational database to hold configuration information, in V5 all configuration information is stored in XML files. The application server's Admin service takes care of the configuration and makes sure it is consistent during the runtime.

Centralized Administration

The Network Deployment configuration allows multiple servers and nodes to be administered from a central location. This is facilitated through the use of a central Deployment Manager that handles the administration process and distributes the updated configuration to the node agent for each node. The node agent, in turn, is responsible for maintaining the configuration for the servers in the node.

Managed processes

All operating system processes that are components of the WebSphere product are called managed servers or managed processes. JMX support is embedded in all managed processes and these processes are available to receive administration commands and to output administration information about the state of the managed resources within the processes.

WebSphere provides the following managed servers/processes:

- Deployment Manager

The Deployment Manager process provides a single, central point of administrative control for all elements in the cell. It hosts the Web-based administrative console application. Administrative tools that need to access any managed resource in a cell usually connect to the Deployment Manager as the central point of control.

The Deployment Manager is responsible for the content of the repositories (configuration and application binaries) on each of the nodes. It manages this through communication with the node agent process resident on each node of the cell.

Using the Deployment Manager, horizontal scaling, vertical scaling and distributed applications are all easy to administer and manage, since application servers are managed by nodes, and one or more nodes is/are managed by a cell.

- Node agent

The node agent is an administrative process and is not involved in application serving functions. It hosts important administrative functions such as:

- File transfer services
- Configuration synchronization
- Performance monitoring

The node agent aggregates and controls all the managed processes on its node by communicating with:

- The Deployment Manager to coordinate configuration synchronization and to perform management operations on behalf of the Deployment Manager.
- Application servers and JMS servers to manage (start/stop) each server and to update its configuration and application binaries as required.

Only one node agent is defined (and run) on each node.

- Application server

Managed server that hosts J2EE applications.

- JMS server

Managed server that hosts the embedded messaging service for a node. There is one JMS server per node. In a base configuration, the JMS server functions are provided within the application server process.

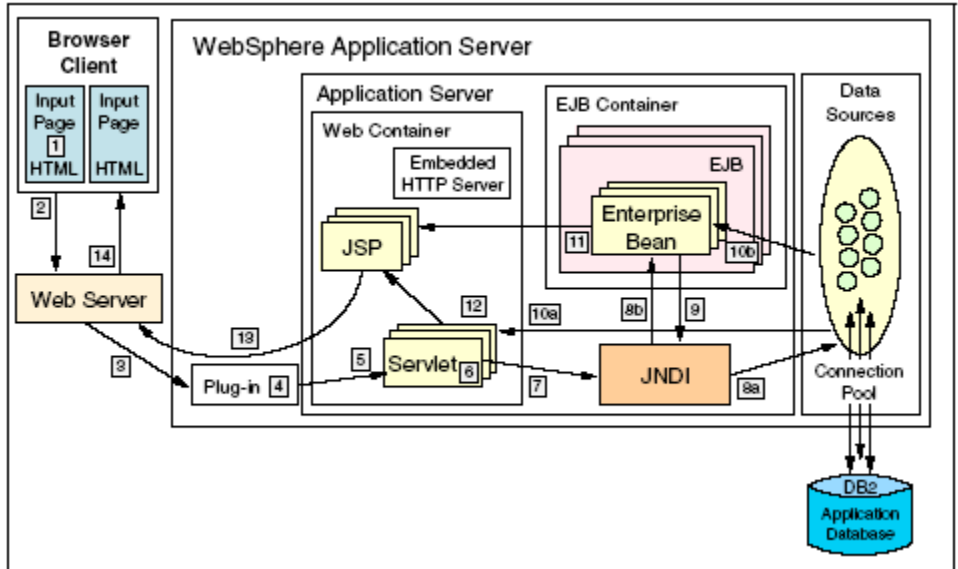
Master configuration repository

With the Network Deployment configuration, a master configuration repository that contains all of the cell's configuration data is maintained by the Deployment Manager. The configuration repository at each node is a synchronized subset of the master repository. The node repositories are read-only for application server access, since only the Deployment Manager can initiate their update, pushing configuration changes out from the cell master configuration repository.

The Flow of an Application

Figure 5.13 on page 121 shows the typical application flow for Web browser clients using either JDBC (from a servlet) or EJB to access application databases.

Figure 5.13 JDBC and EJB application flow



- 1 A Web client requests a URL in the browser (input page).
- 2 The request is routed to the Web server over the Internet.
- 3 The Web server immediately passes the request to the WebSphere plug-in. All requests go to the WebSphere plug-in first.
- 4 The WebSphere plug-in examines the URL, verifies the list of hostname aliases from which it will accept traffic based on the virtual host information, and chooses a server to handle the request.
- 5 A stream is created. A stream is a connection to the Web container. It is possible to maintain a connection (stream) over a number of requests. The Web container receives the request and, based on the URL, dispatches it to the proper servlet.
- 6 If the servlet class is not loaded, the dynamic class loader loads the servlet (servlet `init()`, then `doGet()` or `doPost()`).
- 7 JNDI is now used for lookup of either datasources or EJBs required by the servlet.
- 8 Depending upon whether a datasource is specified or an EJB is requested, the JNDI will direct the servlet:
 - a To the corresponding database, and get a connection from its connection pool in the case of a datasource
 - b To the corresponding EJB container, which then instantiates the EJB when an EJB is requested
- 9 If the EJB requested involves an SQL transaction, it will go back to the JNDI to look up the datasource.
- 10 The SQL statement will be executed and the data retrieved will be sent back:
 - a To the servlet

b To the EJB

11 Data beans are created and handed off to JSPs in the case of EJBs.

12 The servlet sends data to JSPs.

13 The JSP generates output (e.g. HTML) that is sent back through the WebSphere plug-in to the Web server.

14 The Web server sends the output page to the browser.

Common Gateway Interface (CGI) Architecture

This section will give you an overview of the Common Gateway Interface (CGI) which allows a Web server to execute external programs.

CGI Programming Model

The purpose of CGI is to extend the capability of an HTTP server by providing framework in which the HTTP server can interface with a program that is specified on a URL. The format of the URL allows parameters to be passed to the CGI program. On the server side, the interface describes how the program is started by the HTTP server and how parameters for the program are passed using a combination of standard-input and environment variables. It also describes how output information (e.g. HTML elements) are passed back to the HTTP server using standard output. Thus, in its simplest form, a CGI program can be defined as a program that:

- 1** Can be called as an executable and run as a child process of the HTTP server (this does not preclude the use of interpretive languages such as REXX and Net.Data. In this case, their respective interpreters run as child processes of the HTTP server).
- 2** Is able to read from the standard input.
- 3** Is able to access environment variables.
- 4** Is able to write to the standard output.
- 5** Is able to access "command line" arguments passed to `main()`

Given this definition of the CGI interface, the following characteristics are for CGI programs:

- They can be written in compiled programming languages or interpreted scripts.
- They can be simple five or six-line programs that serve a specialized function or generalized business applications that do complex calculations and database transactions (for example, order processing applications).
- Since they run as a child process of the HTTP server, they can benefit from the services provided by the server:
 - Communication over the HTTP protocol
 - Security
 - Firewall transparency. HTTP is usually allowed to pass through firewalls (through socks or application gateways).
 - Resource mapping and name indirection provided by HTTP server directives (program can be moved without affecting client URLs).

- Error logging and audit logging provided by HTTP server
- They provide process isolation. Since each CGI program runs in its own child process, it has the security and integrity advantages that separate processes offer.
- They run in a connectionless environment. The HTTP server starts the CGI program, it does its thing, and terminates. The output is sent to the browser and the connection is broken. Subsequent requests for that same program result in a new connection and a new instance. No state information is maintained unless the CGI program had stored previous state information in fields in the requesting HTML form or parameters in the requesting URL, or cookies, or in a persistent store such as a database or integrated file system (IFS) stream file.
- Process start time often gates performance. The price to pay for process isolation is to suffer the costs of process start time. In a connectionless environment, this becomes an even bigger problem as the process must get started for each request. The IBM HTTP Server for iSeries minimizes this penalty by providing a pool of pre-started server jobs that minimize a majority of the start-up costs. Add to that the ability to exploit named activation groups that also eliminate program initialization costs, and the iSeries CGI model starts to look better than what is provided on other platforms.
- They use short-running transactions. This is a consequence of the connectionless model. The length of a transaction is limited to a single URL request since that is the length of a process. This problem brings about the requirement for the persistent CGI model described later.
- The logic is almost entirely on the server. The CGI model is basically a distributed presentation model where the logic is on the server and the client is responsible for window presentation.

The functions and tasks that CGI programs can perform range from the simple to the very advanced and come in two different types, GGI script and gateway program. CGI scripts do not require to be compiled, while gateway programs require to be compiled. In general, both types are referred to as CGI programs.

Regardless of the type of CGI program, both are designed to handle dynamic information. Dynamic, in this context, refers to temporary information that is created for a one-time use and not stored as a static Web page. This information may be a document, an e-mail message, or the results of a conversion program. You must to compile gateway programs before using them. Compiled programs typically run faster than programs that are written in scripting languages. On the other hand, those programs that are written in scripting languages tend to be easier to write, maintain, and debug.

The administrator controls which CGI programs the system can run by using the server directives. Depending on the server directives, the server calls that program on behalf of the client browser. The IBM HTTP Server for iSeries supports CGI programs that are written in C++, REXX, Java, Net.Data, ILE C, ILE RPG, and ILE COBOL.

Persistent CGI

Persistent CGI is an IBM HTTP Server for iSeries extension to the CGI interface that allows a CGI program to remain active across multiple browser requests and maintain a session with that browser client. This allows files to be left open, the state to be maintained, and long running database transactions to be committed or rolled-back based on end-user input. The CGI program must be written using named activation groups which allows the program to remain active after returning to the server. The CGI program notifies the server it wants to remain persistent using the “Accept-HTSession” CGI header as the first header it returns. This header defines the session ID associated

with this instance of the CGI program and is not returned to the browser. Subsequent URL requests to this program must contain the session ID as the first parameter after the program name (in the URL). The server uses this ID to route the request to that specific instance of the CGI program. The CGI program should regenerate this session ID for each request.

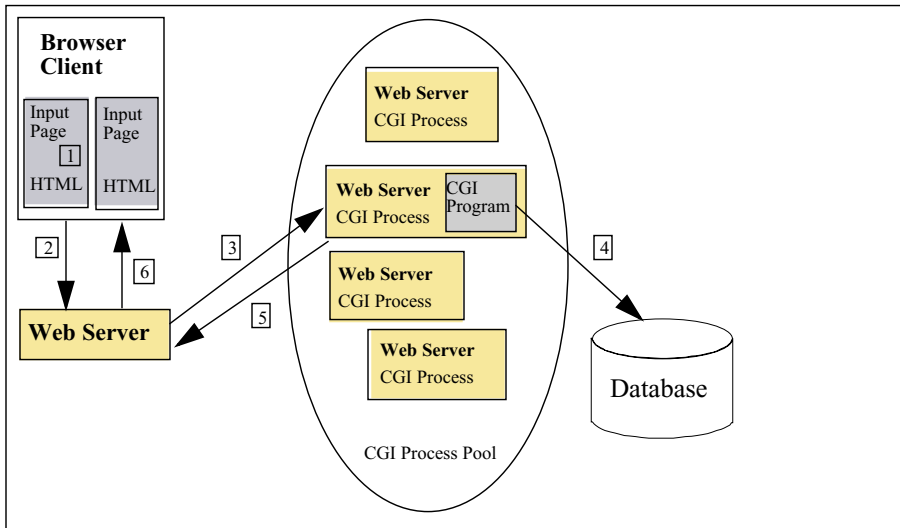
In addition to the “Accept-HTTPSession” CGI header, there is an “HTTimeout” CGI header that allows a CGI program to define the amount of time, in minutes, that this CGI program wants to wait for a subsequent request. This allows individual CGI programs to give users more time to respond to lengthy forms or explanations. However, it still gives the server ultimate control over the maximum time to wait. When the timeout value is reached, the CGI process is ended by the HTTP server.

NOTE Net.Data persistence is built on the persistent CGI support provided by the IBM HTTP Server.

The Flow of an Application

Figure 5.14 on page 124 shows the typical application flow for Web browser clients invoking a CGI application via the IBM HTTP Server for iSeries.

Figure 5.14 CGI program flow



- 1 A Web client requests a URL in the browser (input page).
- 2 The request is routed to the Web server over the Internet.
- 3 The Web server recognizes that the requested URL is a CGI program and immediately dispatches the request to an available CGI process (there is usually a pool of processes available to handle CGI requests). Within the CGI process, parameters are passed to the application program by one of two ways depending on the form's method of either post or get.

- 4 The server is executing the CGI-program. While running, the CGI-program is calling other programs or accessing databases.
- 5 The CGI program passes the output (e.g. HTML) from the other programs or databases plus any additional information back to the server. This is done by writing to the standard output of the application (stdout).
- 6 The Web server sends the output page to the browser.

Resources for Learning

Use the following links to find relevant supplemental information about various topics discussed in this chapter. The information resides on IBM and non-IBM Internet sites, whose sponsors control the technical accuracy of the information.

These links are provided for convenience. Often, the information is useful all or in part for understanding a topic discussed in this chapter. When possible, links are provided to technical papers and Redbooks that supplement the broad coverage of the release documentation with in-depth examinations of particular product areas.

WebSphere Application Server and J2EE

- *WebSphere Application Server V5 Architecture* (January 2004)
<http://www.redbooks.ibm.com/redpapers/pdfs/redp3721.pdf>
This IBM Redpaper covers the architecture of the WebSphere Application Server for the various versions of WebSphere.
- The J2EE Architecture
<http://www.sun.com/j2ee>
Sun's definition of the J2EE architecture.
- WebSphere Application Server for iSeries Home Page
<http://www.ibm.com/servers/eserver/series/software/websphere/wsappserver/>
The home page of WebSphere Application Server for iSeries.
- *WebSphere Development Studio Client for iSeries V5.0* (August 2003)
<http://www.redbooks.ibm.com/redbooks/pdfs/sg246961.pdf>
This IBM Redbook discusses WebSphere Development Studio Client for iSeries V5.0 and the IBM WebFacing Tool.
- *WebSphere Version 5 Application Development Handbook* (December 2003)
<http://www.redbooks.ibm.com/redbooks/pdfs/sg246993.pdf>
This IBM Redbook provides detailed information on how to develop J2EE-compliant Web applications for WebSphere Application Server Version 5 using WebSphere Studio Application Developer Version 5.1 as an integrated development environment. Also covered are a variety of application development tools from IBM and Open Source.
- *Patterns: Self-Service Application Solutions Using WebSphere V5.0 for iSeries* (October 2003)
<http://www.redbooks.ibm.com/redpapers/pdfs/redp3670.pdf>

This Redpaper focuses on the Self-Service::Stand-Alone Single Channel application pattern for facilitating user access to business sites and the Self-Service::Directly Integrated Single Channel application pattern for including one or more point-to-point connections with back-end applications on the IBM iSeries platform.

- *WebSphere Application Server V5 for iSeries: Installation, Configuration, and Administration* (June 2003)

<http://www.redbooks.ibm.com/redbooks/pdfs/sg246588.pdf>

This IBM Redbook will help you to gain proficiency in installing, configuring, and administering the WAS environment on iSeries.

CGI

- General Information

<http://hoohoo.ncsa.uiuc.edu/cgi/>

Link to Web page containing general information on CGI programming.

- *HTTP Server for iSeries Web Programming Guide*

<http://publib.boulder.ibm.com/series/v5r2/ic2924/info/rzaie/rzag3mst.pdf>

Book contains information on writing CGI programs on OS/400.

- CGI Programs for HTTP Server

<http://publib.boulder.ibm.com/series/v5r2/ic2924/index.htm>

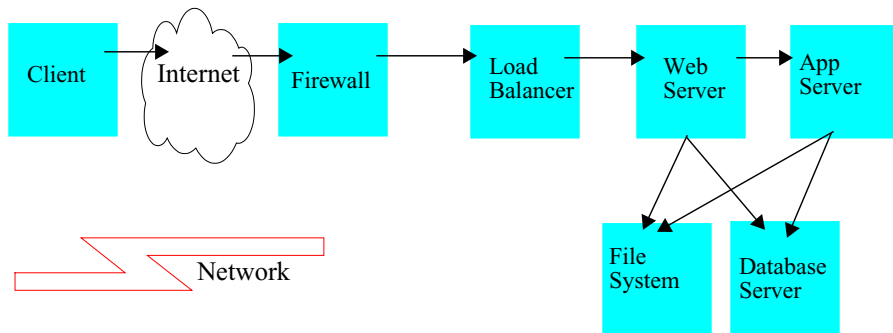
This iSeries Information Center topic contains information on CGI.

RUNTIME COMPONENT CONSIDERATIONS FOR HIGH AVAILABILITY

Identifying Runtime Components in an e-Business Environment

To apply the principles discussed in “High Availability Principles” on page 10, you must identify all the components in your e-business environment. A typical topology would include the components shown in Figure 6.1 on page 127.

Figure 6.1 Components in an e-business environment



Client (User) Component

The user component is most frequently a personal computing device (PC, etc.) supporting a commercial browser, for example, Netscape Navigator or Internet Explorer. The level of the browser is expected to support SSL and some level of DHTML. Increasingly, designers should also consider that this component may be a pervasive computing device, such as a Personal Digital Assistant.

Firewall Component

Firewalls provide services that can be used to control access from a less trusted network to a more trusted network. Traditional implementations of firewall services include:

- Screening routers (the protocol firewall)
- Application gateways (the domain firewall)

The two firewall components provide increasing levels of protection at the expense of increasing computing resource requirements. The protocol firewall is typically implemented as an IP router, while the domain firewall is a dedicated server component.

An alternative to having two firewalls is just having one firewall separating the external world from the internal network. This means that applications behind the firewall should be analyzed and protection mechanisms should be built into those applications that are deemed sensitive to the business.

Having protection built into sensitive business applications is a good idea independent of the firewall topology used.

Firewalls schemes must also support redundancy in order to avoid being a single point of failure.

Load Balancer Component

Establishing a highly available infrastructure means installing redundant components. Load balancing is the mechanism to distribute the incoming work between the redundant components.

A high availability load balancer component introduces a backup load balancer machine that remains ready at all times to take over load balancing tasks should the primary load balancer machine fail. A variation of this is mutual high availability which allows the two load balancer machines to both be active and standby for each other.

Web Server Component

The Web server component includes an HTTP server for serving static documents and any CGI programs and HTTP server plug-ins (e.g. WebSphere Application Server plug-in) that extends the HTTP server so that other technologies can be used to server client requests. The technologies can host both presentation and business logic. The WebSphere Application Server plug-in acts like a load balancer for clustered and redundant WebSphere Application Servers.

Application Server Component

The Application Server component is a functional extension of the informational (publishing-based) Web server component. It provides the technology platform and contains the components to support access to both public and user-specific information by users employing Web browser technology. For the latter, the component provides robust services to allow users to communicate with shared applications and databases. In this way, it acts as an interface to business functions, such as banking, lending, and human resource systems. The application can be clustered using WebSphere clustering capabilities or another option involves deploying the same application to non-clustered application servers and use a load balancer to distribute the requests.

Database Server Component

The database server component provides a persistent data storage and retrieval service in support of transactional interactions. The data stored is relevant to the specific business interaction, for example, bank balance, insurance information, current purchase by the user, etc.

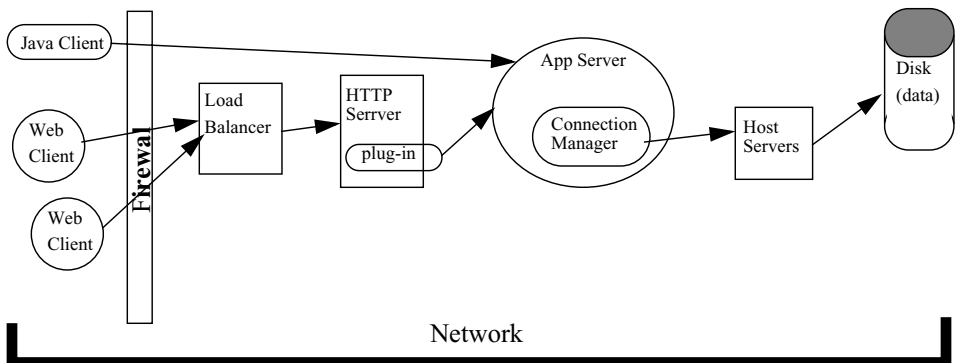
Shared File System Component

The timely synchronization of several Web server components is achieved by using a shared file system as the content storage. On the iSeries system, shared file systems can be done by using such technologies as the Network File System (NFS) or the QFileSvr.400 file system.

Runtime Components and High Availability

Figure 6.2 shows the basic logical topology (logical topologies are shown in graphical form using standard node types to identify the function represented by that node.). Clients (e.g. browser clients) send requests to the HTTP server, which in turn attempts to complete the request by sending the request to the application server.

Figure 6.2 Basic logical topology (not highly available)

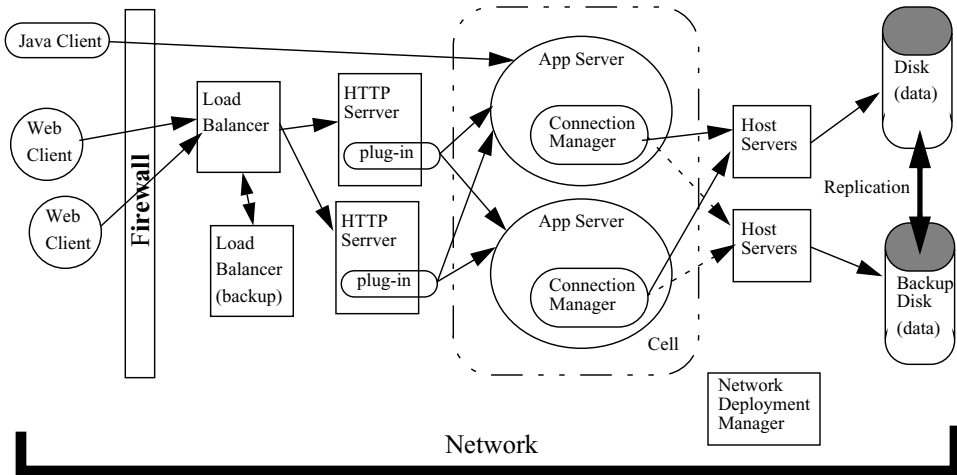


The http server, application server and data are all located within the internal network.

If you take a look at the basic logical topology, you will see that the topology does not meet the high availability principals discussed in “High Availability Principles” on page 10. Any failure in a component would result in clients being unable to access the web site.

Figure 6.3 on 130 illustrates a basic high availability topology.

Figure 6.3 A highly available logical topology.



This topology is composed of the following:

- One Network deployment administrative cell with two nodes federated into the cell. Each of the federated nodes contains one base instance which can contain one or more application servers. The Network Deployment Manager offers central administration of the federated nodes and the application servers running within the nodes. The Network Deployment cell and nodes can all be located on separate systems and/or LPAR partitions.
- Two Apache Web servers configured as a cluster under the control of WebSphere Load Balancer from the IBM WebSphere Application Server Network Deployment V5.0 Edge Components.
- The load balancer sprays client requests to the Apache Web servers. The load balancer is sometimes referred to as a network dispatcher. Both HTTP load balancing and failover are provided with this technique. The load balancer uses a HTTP advisor to perform HTTP health checks. This advisor sends HTTP head requests and expects a response in a configurable time frame. If that response is not received then no requests will be routed down that leg.
- Horizontally clustered WebSphere application servers. This function is part of the WebSphere workload management support. Each application server (sometimes referred to as a cluster member) contains both a Web and EJB container. The same application runs within each cluster member. The clustered application servers are located on the federated nodes.
- Once configured, the Network Deployment manager can be used to generate a HTTP plug-in to be used by the two HTTP web servers. This plug-in is an xml file. Both HTTP server instances shall be directed to the plug-in file. The HTTP servers contain WebSphere code which shall use this plug-in information to perform load balancing and failover amongst the horizontally clustered application servers.
- The database is accessed remotely using the Java Toolbox JDBC driver. The remote DB tier database availability is assured using OS/400 clustering and HABP (High Availability Business Partner) software. All database requests from the application

servers are directed to an IP address where the host servers are located. The database changes are replicated by HAP software from the primary database node to the backup node via HAP data replication. Transaction state is assured. OS/400 clustering is used to perform failover to the backup database tier should something catastrophic happens to the primary database tier. Also planned switchover of the database is supported.

The following chapters will discuss what things one has to consider on a per-component basis when implementing a highly available e-business infrastructure.

CLIENT CONSIDERATIONS FOR HIGH AVAILABILITY

Traditionally two client models are supported in a client server model.

- Web-based clients (servlets, JSPs, CGI)
- Standalone Java application clients

Web-based Clients

Web-based clients utilize a browser to access server-side resources:

- For the J2EE programming environment, the resources are typically accessed through a servlet or JSP. Servlets and JSPs run in a web container and typically in the same Java Virtual Machine (JVM) as the EJB container. The plug-in along with WLM clustered application servers provide load balancing and failover for application servers. See “Web Container Clustering and Failover” on page 167 for more information.
- For the CGI programming environment, the HTTP server in addition to a load balancer can provide failover support.

Stand-alone Java Application Clients

The standalone Java client model allows you to create a Java application which will communicate with enterprise beans deployed in an EJB container. There are two types of standalone Java clients:

- **J2EE application client**

A J2EE application client program operates similarly to a standard Java program in that it runs its own Java virtual machine (JVM) code and is invoked at its main method. The J2EE application client runs in a J2EE client container and uses the Java

Naming and Directory Interface (JNDI) name space to access resources. Sometimes J2EE application client are referred to as a fat client.

- **Thin client**

The thin application clients provide a lightweight Java client programming model and does not run in a J2EE client container.

High availability considerations for stand-alone Java application clients is discussed in “EJB Container Clustering and Failover” on page 177.

FIREWALL CONSIDERATIONS FOR HIGH AVAILABILITY

A number of telecommunications companies offer high-availability firewall solutions. Our advice is to choose a proven high-availability firewall solution that can support the performance, security, and network configuration options that you require for your business. The firewall component is a critical piece that must be very reliable since it is the gatekeeper which allows (or denies) external requests for data residing on your corporate servers. If it is not configured properly, your enterprise data could be compromised. Also, if the particular firewall chosen is not adequate to handle your peak load intervals or lacks failover capabilities, legitimate end users may not be able to access any of your applications, which results in zero availability.

Resources for Learning

Use the following links to find relevant supplemental information about various topics discussed in this chapter. The information resides on IBM and non-IBM Internet sites, whose sponsors control the technical accuracy of the information.

These links are provided for convenience. Often, the information is useful all or in part for understanding a topic discussed in this chapter. When possible, links are provided to technical papers and Redbooks that supplement the broad coverage of the release documentation with in-depth examinations of particular product areas.

High-Availability Firewall

- *Setting up a High Availability Firewall on Linux for iSeries* (October 2002)
<http://publib.boulder.ibm.com/series/v5r2/ic2924/info/experience/hafwl.pdf>
Experience report on setting a high-availability firewall on Linux for iSeries.

LOAD BALANCER CONSIDERATIONS FOR HIGH AVAILABILITY

Functioning in conjunction with content hosts, such as WebSphere Application Server, a load balancer enables you to enhance your network's availability and scalability.

A load balancer is used by enterprise networks and is installed between the Internet and the enterprise's back-end servers. The load balancer acts as the enterprise's single point-of-presence on the Internet, even if the enterprise uses multiple back-end servers because of high demand or a large amount of content.

Availability is achieved through load balancing and failover support.

In this chapter we focus on using a load balancer to cluster Web servers. The information on load balancing is based on the load balancer that is included in the Edge Components of IBM WebSphere Application Server Network Deployment V5.0. Even though we use the load balancer supplied by the Edge Components, the concepts and ideas regarding load-balancing in a high-availability environment should be applicable to any load-balancing solution you may choose.

- NOTE** The Edge Components include a very rich set of function. The material covered in this chapter is only a portion of the whole. We do not cover caching proxy, but we highly recommend you visit existing Edge Component documentation on this subject. A well thought-out caching methodology is critical to a well performing Web site. See "Resources for Learning" on page 154 for links to more information on the Edge Components.
- NOTE** On iSeries the load balancer does not run natively in OS/400, but can be run in a Linux partition or any of the other supported platforms. The fact that it does not run natively in OS/400 does not diminish it's usefulness in an OS/400 e-business configuration.

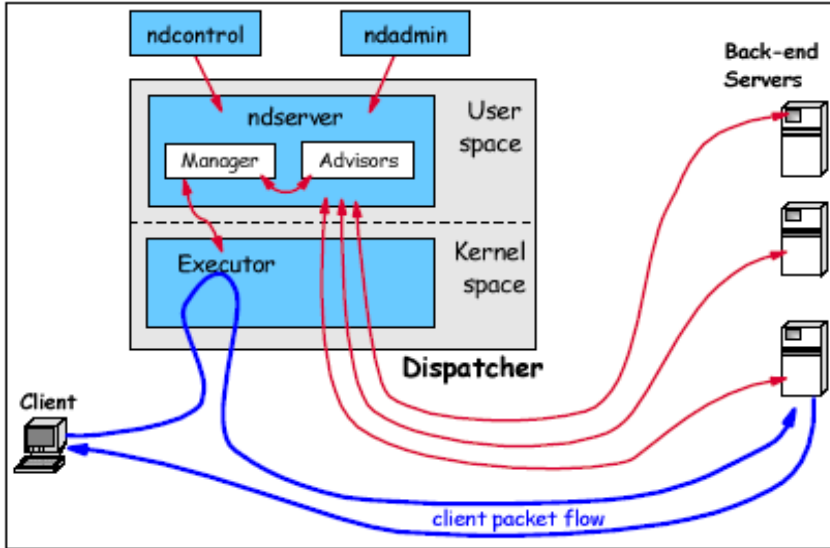
WebSphere Edge Server Load Balancer

The Load Balancer consists of the multiple components which can be used separately or together. For our discussion, we are interested in the Dispatcher component, which is the load balancing component that is able to dynamically monitor and balance Web

servers and corresponding applications in real time. It improves a Web site's availability, scalability and performance by transparently clustering edge, Web and application servers. It enables multiple web servers to be dynamically linked in a single entity that appears to the network as a single logical server.

The following figure is used to illustrate the main Dispatcher components.

Figure 9.1 Dispatcher components - MAC forwarding example



The main Dispatcher components are as follows:

- **Executor** - The core function of the Dispatcher is the Executor. It is a kernel-level function (or device driver) which examines only the header of each packet and decides whether the packet belongs to an existing connection or represents a new connection request. The Executor uses a simple connection table stored in memory to achieve this. Note that the connection is never actually set up on the Dispatcher machine. The connection is between the client and the server, just as it would be if the Dispatcher were not installed. But the connection table records its existence and the address of the server to which the connection was sent. The Executor selects which back-end server to forward a new connection packet to based on information provided to it by the Manager.
- **Manager** - The Manager periodically sets the weights used by the Executor to determine the right back-end server to forward a new connection request to. The Manager can use four metrics to set the weights for back-end servers:
 - The count of active connections for each back-end server on each port. This count is maintained in the Executor.
 - The count of new connections for each server on each port since the last manager interval started. This count is maintained in the Executor.
 - A check that each back-end server has “displaceable capacity” which means that it can realistically process the work.
 - Feedback from Advisors that provide an application-level check that each server is up and running.

- **Advisors** - An Advisor is a lightweight client that runs as part of Dispatcher, but actually passes real commands to each back-end server. It only executes a trivial command. If this request is successful, the server is deemed to be up. If it fails, the Advisor informs the Manager, and the Manager sets the weight of that back-end server to zero until the server responds again.
- **ndserver** - A Java application that contains the user space threads in a Dispatcher configuration.
- **cbrserver** - A Java application that contains the user space threads in a CBR configuration.
- **ndcontrol** - A Java application to support command line configuration of the Load Balancer.
- **cbrcontrol** - A Java application to support command line configuration of the Content Based Routing component of the Load Balancer.
- **ndadmin** - A Java application to support a graphical user interface (GUI) configuration of the Load Balancer.

The Dispatcher can balance load on servers within a local area network or wide area network using server weights that can be dynamically set by the Dispatcher or a user written custom advisor programs. Advisors are shipped (custom advisors can be created) with the product and can be used to perform health checks on the clustered Web servers. These advisors work hand in hand with the Dispatcher to assure client requests are routed to servers which will service the request.

In addition, the Dispatcher's content based routing (sometimes referred as kernel-based cbr or Dispatcher component's cbr), when used with passive cookie affinity, provides a performance efficient method to assure server affinity between the Dispatcher and a particular Web server.

The topics and recommendations we shall focus on have proven to compliment a highly scalable and available OS/400 e-business solution.

Dispatcher and Server Affinity

Server affinity is a technique that enables the Dispatcher to remember which server was chosen for a certain client at his initial request. Subsequent requests are then directed to the same server again.

If the affinity feature is disabled when a new TCP/IP connection is received from a client, Dispatcher picks the right server at that moment and forwards the packet to it. If a subsequent connection comes in from the same client, Dispatcher treats it as an unrelated connection, and again picks the most appropriate server at that moment.

With the affinity feature enabled, if a subsequent request is received from the same client, the request is directed to the same Web server. An affinity record is kept for the client at the Dispatcher level. Over time, the client will finish sending transactions, and the affinity record will go away. Hence the meaning of "stickytime". Each affinity record lives for stickytime seconds. When subsequent connections are received within the stickytime, the affinity record is still valid and the request will go to the same server. If a subsequent connection is not received within sticky time, the record is purged; a connection that is received after that time may have a new Web server selected for it.

Why is affinity important? Server affinity allows load balancing for those applications that need to preserve state across distinct connections from a client. Maintaining state is

a requirement of many application encountered on the Internet today, including “shopping carts”, home banking, and so on. If the shopping cart is only kept in memory of the application in which it was created, then subsequent client requests must get routed back to the same Web server and eventually application. Even if the shopping cart is persisted to shared storage, there are still performance advantages to be routed to the same server. Another important reason for server affinity is SSL session ID. Here client HTTPS requests are directed to the same Web server in order to avoid additional and expensive SSL handshake processing.

There are three type of affinities we have found to be useful:

- “Stickyness” to source IP address
- Passive Cookie
- SSL session ID

Each is discussed in more detail below.

“Stickyness” to Source IP Address Affinity

A group of clustered Web application servers all listen for client requests on the same set of ports. Within the Dispatcher the hierarchical definition of a cluster is as follows:

- 1** Cluster IP address (provides single point of presence for client requests).
- 2** Clustered Port (port upon which clustered Web servers are listening on).
- 3** List of clustered Web servers which client requests will be load balanced to for scalability and failover purposes.

Given this hierarchical definition of a cluster, the “Stickyness to source IP address” is enabled by configuring the clustered port to be sticky. Configuring a cluster’s port to be sticky allows subsequent client requests to be directed to the same Web server. The clustered port stickytime is set to a positive number with the unit of measure being seconds. The sticky feature can be disabled by setting the port stickytime to zero.

The affinity record kept by the Dispatcher includes the client IP address, Web server last visited, and time since last visited.

NOTE This affinity strategy has some drawbacks. Some ISPs use proxies which collapse many client connections into a small number of source IP addresses. A large number of users who are not part of the session will be connected to the same server. Other proxies use a pool of user IP addresses chosen at random, even for connections from the same user, invalidating the affinity.

Passive Cookie Affinity

Passive cookie affinity is rules based, which simply means that the affinity depends on the content of the client request. In essence, passive cookie affinity is based on the content of cookies (name/value) generated by the content provider (e.g. WebSphere application server, HTTP server, etc.). You must specify a cookie name to be monitored by Dispatcher in order to distinguish which server the request is to be sent to.

If the cookie name is not found in the client request or the cookie value does not match any of the servers’ cookie values kept in the Dispatcher affinity record table, the server will be chosen using the weighted round-robin technique.

The affinity record kept by the Dispatcher includes the cookie and the corresponding server.

Note that there is no sticky time. This is due to the fact that the cookie or HTTP session is managed by the content provider. For example, the WebSphere application server Session Management component supports a session time-out value which governs how long the session can be idle before it becomes invalid. When it does become invalid the cookie within the client data stream will denote the invalid cookie.

SSL Session ID Affinity

During establishment of an SSL encrypted session, a handshake protocol is used to negotiate a session ID. This handshaking phase consumes a good deal of CPU power, so directing HTTPS requests to the same server, using the already established SSL session, saves processing time and increases the overall performance of the HTTP server.

Dispatcher watches the packets during the handshake phase and holds information about the session ID if SSL session negotiation is detected.

The forwarding method used to configure SSL session ID affinity is the kernel-based content-based routing function called *cbr forwarding* (discussed in the following section).

The Dispatcher affinity record includes the SSL session ID, Web server IP address and time since last accessed.

A sticky time can be configured on a port basis.

Dispatcher Forwarding Methods

The Dispatcher component performs intelligent load balancing by using server availability, capability, workload, and other criteria (that you can define) to determine where to send a request. When routing a client request, the Dispatcher can use one of four routing methods, all based on port, on which the clustered Web servers are listening on. All four routing methods have different purposes:

- Media access control (MAC) forwarding
- Network Address Translation (NAT) forwarding
- Network Address Port Translation (NAPT) forwarding
- Dispatcher content-based routing (cbr)

Each is discussed in more detail below.

Media Access Control (MAC) Forwarding

MAC forwarding permits load balancing of multiple back-end server machine running on the same subnet as the Dispatcher machine.

How it works

MAC forwarding load balances client requests across multiple back-end servers based on load calculations and input from advisors. Clients see one IP address for the entire site; this address is called the *cluster* address. The cluster address is aliased to the Dispatcher network interface card. Clients send requests to the cluster address. The firewall or router knows how to route to the Dispatcher server. In addition, the cluster address must be added to the loopback adapter on all the back-end servers that are part of the cluster. This must be done so that the back-end servers will accept unmodified packets forwarded to them from the Dispatcher machine.

The Dispatcher selects a back-end server to satisfy a client request, changes the destination MAC address, and forwards the *unmodified* data packet to the selected server. The back-end servers respond directly to the client without going back through the Dispatcher server. The Dispatcher keeps a record of each active session between clients and back-end servers so that additional data flows on the session will be sent to the same back-end server. When the session is ended, the record is removed from the Dispatcher machine.

Figure 9.2 on page 142 illustrates what the topology may look like when using MAC forwarding.

Figure 9.2 Dispatcher MAC forwarding

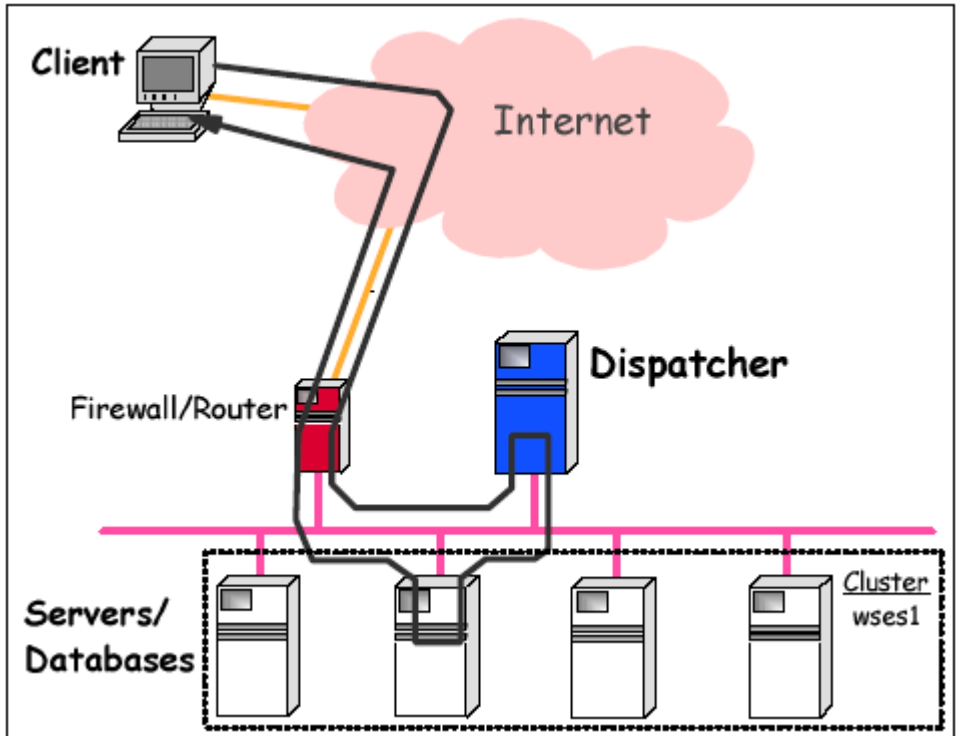
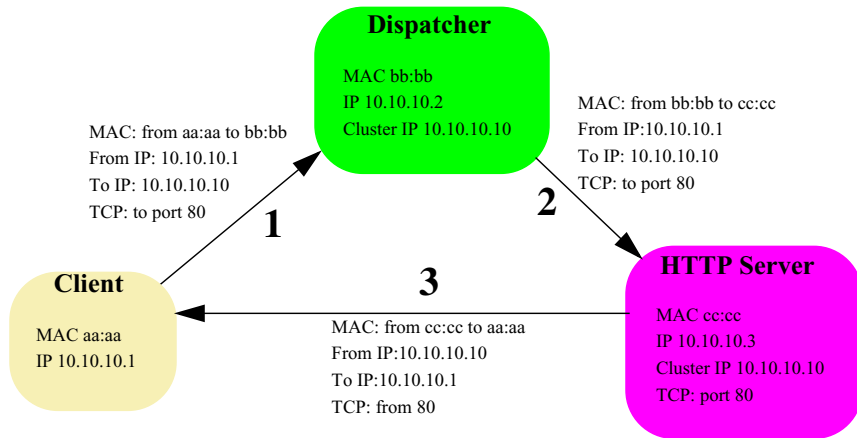


Figure 9.3 on page 143 illustrates how ports, MAC and IP addresses come into play when MAC forwarding is used.

Figure 9.3 Dispatcher MAC forwarding - address flow



'From' IP address is the client system. This implies that the connection to the Web server will be between the Client system and the Web server. So in this scenario the Dispatcher just routes the request and the response from the server is back to the client. The only server affinity option here is via “stickyness” to source IP address enabled by configuring the clustered port to be sticky. In doing so, all subsequent connections from the same source IP address are dispatched to the same server until a configurable time out (sticky time) expires.

When using MAC based routing one must configure a virtual IP address of 10.10.10.10 on the iSeries running the clustered Web server(s). In the example above one would configure a virtual IP 10.10.10.10 on the iSeries hosting the Web servers. This allows the packet sent to the Dispatcher cluster to be accepted by the iSeries server.

The Dispatcher MAC forwarding load balancing process is completely transparent to both the client and back-end server. The Dispatcher sees only the inbound client-to-server flows.

Considerations

The load balanced back-end servers must be configured with the cluster address on their loopback adapter.

The same data content must be available on each back-end server in a cluster.

- Advantages:
 - Requests account only for about 5% of the network traffic. Therefore, older server models are capable of balancing quite high volumes of requests.
 - A separate high capacity return path may be configured to send requested data from the back-end servers directly to the client bypassing the Dispatcher subnet. This makes it more flexible to adapt to existing networks or changes to them.
- Limitations:
 - For MAC forwarding to work, the load balanced back-end servers must be on the same subnet at the Dispatcher machine. However, the back-end servers can be on different subnets from one another.

Network Address Translation (NAT) Forwarding

NAT forwarding supports the following load balancing configurations:

- Multiple back-end server machines running on the same subnet as the Dispatcher machine.
- Multiple back-end server machines running on a different subnet from the Dispatcher machine.
- Multiple server daemons (each with their own IP address) running on the same back-end server machine.

How it works

NAT forwarding load balances client requests across multiple back-end servers based on load calculations and input from advisors. Clients see one IP address for the entire site: this address is called the *cluster* address. The Dispatcher advertises that address. The firewall or router knows how to route to the Dispatcher server. The Dispatcher selects a back-end server to satisfy a client request and forwards the *modified* packet to the selected server. The packet modifications involve the Dispatcher changing the source IP address from the client IP address to the *return* IP address of the Dispatcher machine, and changing the destination IP address from the cluster IP address to the IP address of the selected back-end server. The back-end servers respond to the client through Dispatcher server (using the *return* IP address) where the Dispatcher again modifies the source and destination IP addresses to their original values.

In Figure 9.4 on page 145, the Dispatcher load balanced the client request to back-end Server D on a different subnet. Note that inbound and outbound flows must both go through the Dispatcher machine. If the Dispatcher had load balanced the client request to back-end Server C on the same subnet as the Dispatcher machine, inbound and outbound traffic flows would still go through the Dispatcher machine. Note that Servers D and E are two different server daemons running on the same back-end machine, each with its own IP address.

Figure 9.4 Dispatcher NAT forwarding to a different subnet

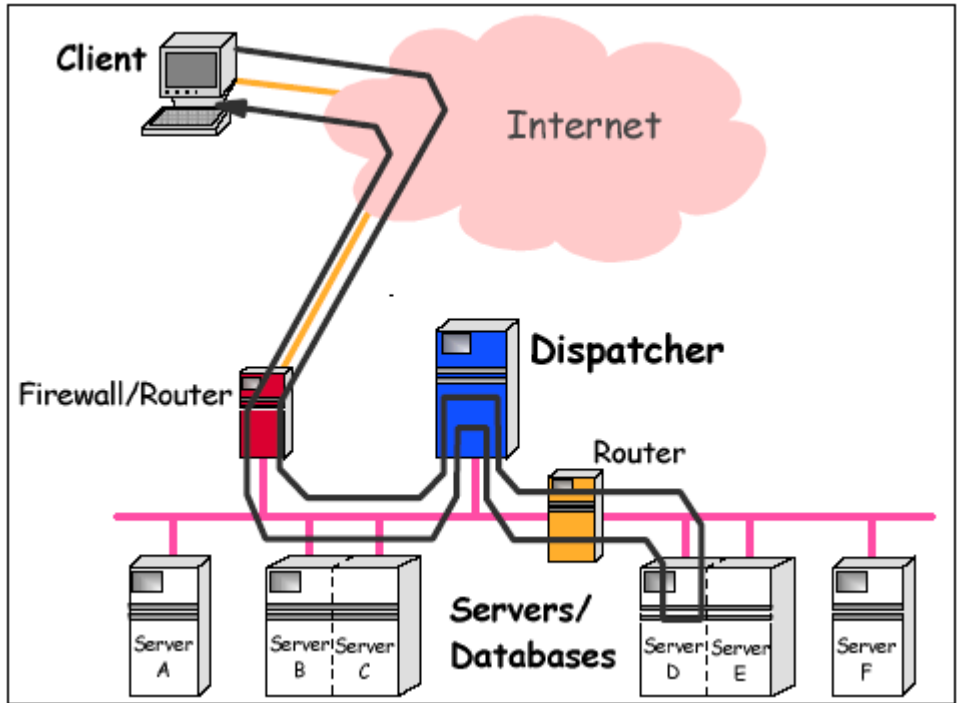
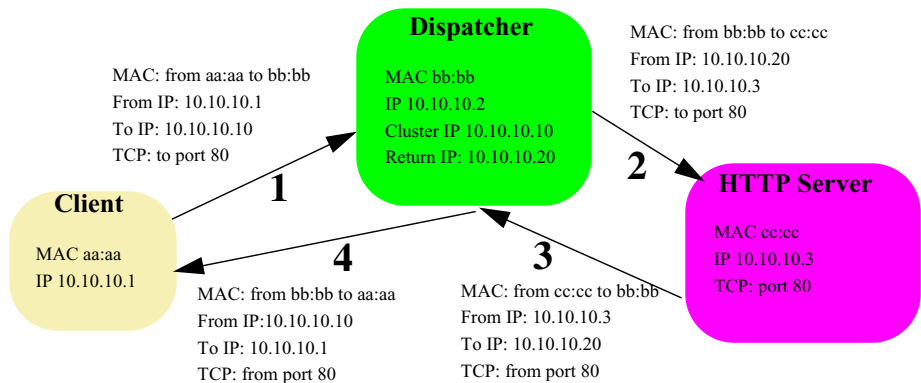


Figure 9.5 on page 145 illustrates how ports, MAC and IP addresses come into play when NAT forwarding is used.

Figure 9.5 Dispatcher NAT forwarding - address flow



Note in this case the packet sent to the Web server has both to and from IP address changed. This will assure a returned response to the Dispatcher. Before the response is

returned to the client, the Dispatcher will change from and to address to what the client tier TCP/IP tier would expect. All of this allows the Dispatcher to reside in a different network than the clustered Web servers.

Note when using this scheme it is not necessary to configure a virtual IP address on the iSeries systems where the Web servers reside. As with MAC, the only server affinity option here is via “stickyness” to source IP address. This is enabled by configuring the clustered port to be sticky. In doing so, all subsequent connections from the same source IP address are dispatched to the same server until a configurable time out (sticky time) expires.

The Dispatcher NAT forwarding load balancing process is completely transparent to both the client and back-end server.

Considerations

The load balanced back-end servers do not require the cluster address to be configured on their loopback adapter.

NAT works well with upper-level application protocols such as HTTP, SSL, IMAP, POP3, NNTP, SMTP, Telnet, etc.

- Advantages:
 - NAT allows you to load balance content servers that are not on the local subnet.
- Disadvantages:
 - All response traffic is returned to the NAT server and the IP address is re-translated. This requires considerably more computing power than the MAC forwarding method
 - Packet responses sent from the back-end servers to clients must return through the NAT server. Therefore, the Dispatcher machine must have bandwidth equal to the sum of all the back-end servers' bandwidth.
 - The Load Balancer's implementation of NAT is a simple implementation of this feature. It only analyzes and operates upon the contents of TCP/IP packet headers. It does not analyze the contents of the data portion of the packets. For Load Balancer, NAT will not work with application protocols, such as FTP, which imbed the addresses or port numbers in the data portion of the messages. This is a well-known limitation of header-based NAT.

NOTE If you do not set client gateway address to a non-zero value, the forwarding method can only be MAC.

Network Address Port Translation (NAPT) Forwarding

NAPT forwarding supports the following load balancing configurations:

- Multiple back-end server machines running on the same subnet as the Dispatcher machine.
- Multiple back-end server machines running on a different subnet from the Dispatcher machine.
- Multiple server daemons running on the same back-end server machine with the same IP address; however, each server daemon listens on a different port.

Note when using this scheme it is not necessary to configure a virtual IP address on the iSeries systems where the Web servers reside. As with MAC, the only server affinity option here is via “stickyness” to source IP address. This is enabled by configuring the clustered port to be sticky. In doing so, all subsequent connections from the same source IP address are dispatched to the same server until a configurable time out (sticky time) expires.

Considerations

The load balanced back-end servers do not require the cluster address to be configured on their loopback adapter.

The Load Balancer's Network Address Port Translation (NAPT) forwarding method works well with upper-level application protocols such as HTTP, SSL, IMAP, POP3, NNTP, SMTP, Telnet, etc.

- Advantages:
 - Allows you to configure multiple server daemons (running on the same physical server) to listen on different port numbers.
 - Provides an additional layer of security by allowing a port change.
- Disadvantages:
 - All response traffic is returned to the NAPT server and the IP address is re-translated. This requires considerably more computing power than the MAC forwarding method.
 - Packet responses sent from the back-end servers to clients must return through the NAPT server. Therefore, the Dispatcher machine must have bandwidth equal to the sum of all the back-end servers' bandwidth.
 - The Load Balancer's implementation of NAPT is a simple implementation of this feature. It only analyzes and operates upon the contents of TCP/IP packet headers. It does not analyze the contents of the data portion of the packets. For Load Balancer, NAPT will not work with application protocols, such as FTP, which imbed the addresses or port numbers in the data portion of the messages. This is a well-known limitation of header-based NAPT.

Dispatcher Content-Based Routing (cbr forwarding)

The Dispatcher component allows you to perform content-based routing for HTTP (using the “content” type rule) and HTTPS (using SSL session ID affinity) without having to use Caching Proxy. For HTTP and HTTPS traffic, the Dispatcher component's cbr forwarding method can provide faster content-based routing than the CBR component, which requires Caching Proxy.

NOTE The Load Balancer has a Content Based Routing (CBR) component that, in conjunction with the Caching Proxy component, allows HTTP requests to be distributed based on URL or other administrator-determined characteristics. We do not cover the CBR component or Caching Proxy in this document. Information in these topics can be found in “Resources for Learning” on page 154.

How it works

With the Dispatcher cbr forwarding method, Dispatcher load balances the incoming request to the server. The server returns the response to Dispatcher and the Dispatcher machine then returns the response to the client. The diagram would look very similar to

NAT, as a matter of fact cbr does use NAT. This use of NAT allows forwarding the Dispatcher to reside in a different network than the clustered Web servers.

For HTTP requests, server selection for Dispatcher's content-based routing is based upon the contents of a URL or an HTTP header. It is configured using the "content" type rule. When configuring the content rule, specify the search string "pattern" and a set of servers to the rule. When processing a new incoming request, this rule compares the specified string with the client's URL or with the specified HTTP header in the client request.

If Dispatcher finds the string in the client request, Dispatcher forwards the request to one of the servers within the rule. Dispatcher then relays the response data from the server to the client ("cbr" forwarding method).

If Dispatcher does not find the string in the client request, Dispatcher does not select a server from the set of servers within the rule.

For HTTPS (SSL) requests, Dispatcher's content-based routing load balances based on the SSL ID session field of the client request. With SSL, a client request contains the SSL session ID of a prior session, and servers maintain a cache of their prior SSL connections. Dispatcher's SSL ID session affinity allows the client and server to establish a new connection using the security parameters of the previous connection with the server. By eliminating the renegotiation of SSL security parameters, such as shared keys and encryption algorithms, the servers save CPU cycles and the client gets a quicker response. In order to enable SSL session ID affinity the `protocol` type specified for the port must be SSL and `port stickytime` must be set to a nonzero value. When stickytime has been exceeded, the client may be sent to a different server from the previous.

Considerations

For HTTP and HTTPS traffic, the Dispatcher component's cbr forwarding method can provide faster content-based routing than the CBR component, which requires Caching Proxy.

The considerations for NAT should be reviewed since Dispatcher cbr forwarding uses NAT.

Advisors

Advisors are lightweight clients that run inside the Dispatcher, providing information about the load of a given server. The Edge Component product provides protocol-specific advisors for most popular protocols (HTTP, SSL, FTP, Ping, DB2, POP3, DNS, Telnet, and others). The one we shall find very useful is the HTTP advisor. Standard advisors send transactions periodically to determine the status of the servers (for example, for HTTP an HTTP HEAD request is sent, for FTP, a SYST). If the transaction succeeds, the server is considered up.

To obtain the load value, the advisor:

- 1 Pings and opens a connection with each server.
- 2 Sends a protocol specific request message. An example of this would be a HTTP head request.
- 3 Listens for a response from the server.
- 4 Calculates the load value.

After getting the response, the advisor makes an assessment of the server. To calculate this “load” value, most advisors measure the time for the server to respond, and then use this value (in milliseconds) as the load and reports this value to the manager function.

If the server does not respond, the advisor returns a negative value (-1) for the load.

The manager function obtains the load value reported by the advisor, which is available in the manager report, in the Port column. The manager obtains these values from all of its sources and sets proportional weight values for the executor function.

A server that is down is given a weight of zero, and packets will not be forwarded to it until the server responds to the advisor again.

Custom Advisors

You can also write your own advisors for specific applications. These are called *custom advisors*. The custom (customizable) advisor is a small piece of Java code, which you provide as a class file that gets called by the base Dispatcher code. The base code provides all administrative services, such as starting and stopping an instance of the custom advisor, providing status and reports, and recording history information in a log file. It also reports results to the Manager component. Periodically the base Dispatcher code will perform an advisor cycle, where it individually evaluates all servers in its configuration. It starts by opening a connection with a server machine. If the socket opens, the base code will call the `getLoad` method (function) in the custom advisor. The custom advisor then performs whatever steps are necessary to evaluate the health of the server. Typically, it will send a user-defined message to the server and then wait for a response. (Access to the open socket is provided to the custom advisor.) The base Dispatcher code then closes the socket with the server and reports the load information to the Manager. The Manager takes this information and decides whether it should continue to route client requests or not to the Web server. The base code and custom advisor can operate in either normal or replace mode. Choice of the mode of operation is specified in the custom advisor file as a parameter in the constructor method. In normal mode, the custom advisor exchanges data with the server, and the base advisor code times the exchange and calculates the load value. The base code then reports this load value to the Manager. The custom advisor needs only return a zero (on success) or negative one (on error). To specify normal mode, the replace flag in the constructor is set to false. In replace mode, the base Dispatcher code does not perform any timing measurements. The custom advisor code performs whatever operations are desired for its unique requirements, and then returns an actual load number. This load value is given to the Manager which determines the load given to each Web server. For best results, normalize your load number between 10 and 1000, with 10 representing a fast server, and 1000 representing a slow server. To specify replace mode, the replace flag in the constructor is set to true. With this feature, you can write your own advisors that will provide the precise information about servers that you need.

Dispatcher Availability Features

The Dispatcher component offers a built-in high availability feature, eliminating the Dispatcher as a single point of failure from your network. This feature involves the use of a second Dispatcher machine that monitors the main, or primary, machine and stands by to take over the task of load balancing should the primary machine fail at any time. The Dispatcher component also offers mutual high availability which allows two machines to be both primary and secondary (backup) for each other.

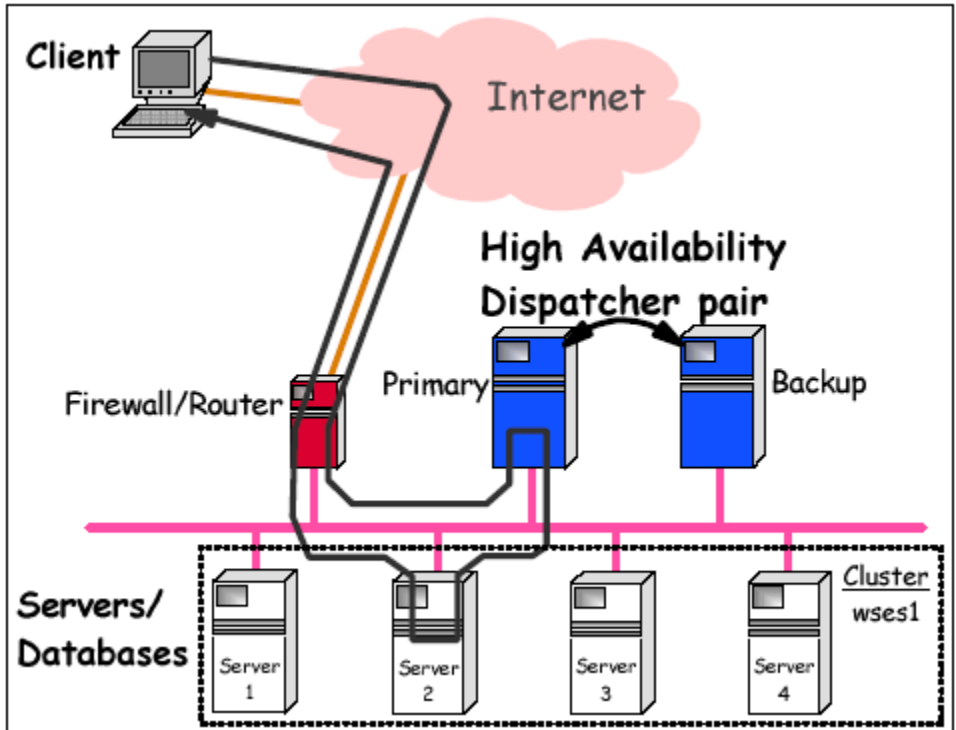
Dispatcher High Availability

Dispatcher high availability uses a pair of Dispatchers in a primary/backup configuration, where one dispatcher is active and the other is in standby mode.

How it works

High availability involves the use of a second Dispatcher machine that monitors the main, or primary, machine and stands by to take over the task of load balancing should the primary machine fail at any time (see Figure 9.7 on page 151).

Figure 9.7 Dispatcher high availability



Each Dispatcher machine has a specific role: primary or backup. In addition, each Dispatcher is in one of two states: active or standby. The primary machine is normally active and routes packets based on load balancing algorithms. The backup machine is normally in standby state and does not route. While the primary is active (load balancing), the backup is in a standby state, continually updated and ready to take over, if necessary. The active machine continually informs the standby machine of all the connections it is routing.

The standby machine is ready to take over and preserve connections in case the primary machine fails. If the backup machine detects that the active machine has failed, it will take over and begin load balancing. At that point, the statuses of the two machines are reversed: the backup machine becomes active and the primary machine becomes standby.

A “heartbeat” mechanism between the two Dispatcher machines is used to detect a Dispatcher failure. The standby machine makes the decision to take over if it detects that all heartbeats have stopped for two seconds. The standby machine then changes state to active. It also broadcasts gratuitous Address Resolution Protocol (ARP) commands so that everyone on the subnet (including the router) will now send packets for the cluster addresses to the standby (now active) machine.

Considerations

High availability is supported in MAC, NAT and NAPT forwarding configurations.

Both Dispatcher machines must be on the same LAN segment. In addition, both Dispatcher machines must have connectivity to the same clients and the same cluster of back-end servers.

When a standby Dispatcher machine takes over after a failure of the primary machine, most (but not all) of the existing in-flight connections are preserved. Clients see no disruption. Those connections that fail are typically connections that have just started or just ended.

The two Dispatcher configurations should be identical.

Dispatcher Mutual High Availability

Dispatcher mutual high availability uses a pair of Dispatchers in a primary/backup configuration. However, both dispatchers are active.

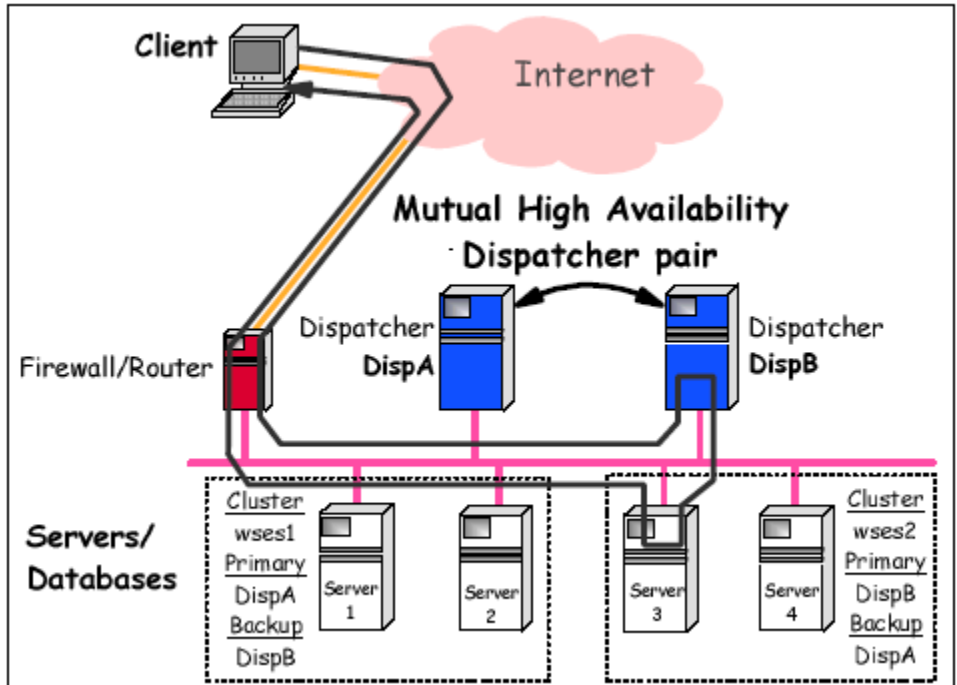
How it works

Mutual high availability is similar to high availability in that it uses a second Dispatcher machine to monitor the main, or primary, machine. However, in the case of mutual high availability, the backup (standby) machine has clusters of its own that it is load balancing while acting as a standby machine for clusters on the other Dispatcher.

Both machines actively perform load balancing of a portion of the client traffic, and both machines provide backup for each other.

Each machine in a Dispatcher mutual high availability configuration has clusters defined to it for which it performs load balancing functions. Each machine performs the role of primary for the clusters it is load balancing, while at the same time acting as backup for clusters on the other Dispatcher machine (see Figure 9.8 on page 153).

Figure 9.8 Dispatcher mutual high availability



In Figure 9.8 on page 153, Dispatcher DispA is the primary machine for cluster wses1 and the backup machine for cluster wses2. Likewise, DispB is the primary machine for cluster wses2 and also the backup machine for cluster wses1. If either dispatcher machine fails, scripts on the other dispatcher machine are automatically executed to take over the load balancing responsibilities for the clusters on the failed machine.

The “heartbeat” mechanism described in “Dispatcher High Availability” on page 151 also applies to a mutual high availability configuration.

Mutual high availability is different from high availability in that it is based specifically on cluster addresses rather than on a Dispatcher machine as a whole.

Considerations

Mutual high availability is supported in MAC, NAT and NAPT forwarding configurations.

Both Dispatcher machines must configure both their own clusters and the clusters they back up in the same way.

Both Dispatcher machines must be on the same LAN segment. In addition, both Dispatcher machines must have connectivity to the same clients and the same clusters of back-end servers.

When a standby Dispatcher machine takes over after a failure of the primary machine, most (but not all) of the existing in-flight connections are preserved. Clients see no disruption. Those connections that fail are typically connections that have just started or just ended.

Resources for Learning

Use the following links to find relevant supplemental information about various topics discussed in this chapter. The information resides on IBM and non-IBM Internet sites, whose sponsors control the technical accuracy of the information.

These links are provided for convenience. Often, the information is useful all or in part for understanding a topic discussed in this chapter. When possible, links are provided to technical papers and Redbooks that supplement the broad coverage of the release documentation with in-depth examinations of particular product areas.

WebSphere Edge Components

- WebSphere Application Server - Edge Component InfoCenter
<http://www.ibm.com/software/webservers/appserv/ecinfocenter.html>
This WebSphere Information Center Web page contains links to online and printable product documentation for WebSphere Application Server, Edge Component, Version 5.
- *Patterns for the Edge of Network* (November 2002)
<http://www.redbooks.ibm.com/redbooks/pdfs/sg246822.pdf>
This IBM Redbook describes guidelines and options for the selection of Runtime patterns that include high availability and high performance considerations in the design process.
- *IBM WebSphere V5.0 Performance, Scalability, and High Availability: WebSphere Handbook Series* (July 2003)
<http://www.redbooks.ibm.com/redbooks/pdfs/sg246198.pdf>
This IBM Redbook discusses various options for scaling applications based on IBM WebSphere Application Server Network Deployment V5.0. It also discusses high availability. A must-read. A lot of the information in this document was obtained from this Redbook.

CHAPTER 10

HTTP SERVER CONSIDERATIONS FOR HIGH AVAILABILITY

If Web serving is a critical aspect of your business, you may want a highly available Web server environment. Highly available HTTP servers make it possible to build a highly available Web site. This improves the availability of business-critical Web applications built with static Hypertext Markup Language (HTML) pages, Common Gateway Interface (CGI) programs, or J2EE applications.

This chapter focuses on how the IBM HTTP Server for iSeries (powered by Apache) can be used as a highly available (HA) Web server.

Highly Available Web Server

The IBM HTTP Server for iSeries (powered by Apache) high availability feature is based on iSeries clustering technology (see “Clusters” on page 65 for more information). Clusters are a collection of complete systems that work together to provide a single, unified computing capability.

How It Works

There are two parts to the high availability support provided by the HTTP server: heartbeat monitoring and session state.

Heartbeat Monitoring

The HTTP server performs heartbeat monitoring by using a *liveness monitor* that periodically checks the state of the Web server. It interacts with the Web server and the clustering resource services in the event that a Web server fails (failover), or a manual switchover takes place (ensures no interruption of Web server services).

The HTTP server configuration directives relating to high availability are shown in Table 10.1 on page 156.

Table 10.1 Directives for highly available HTTP server

Directive ¹	Description
HACGI	The HACGI directive specifies if CGI programs in a directory can be highly available. The CGI programs in the specified directory must use the highly available HTTP Server APIs.
HAModel	The HAModel directive establishes which highly available model is to be used (PrimaryBackupWithIpTakeover, PrimaryBackupWithDispatcher, or PurePeer). See “Highly Available Web Server Models” on page 157 for more information.
LmExitProgram	The LmExitProgram directive configures a QSYS program that is started by the Liveness monitor whenever it initiates a change in the HA model of a server instance from the primary model or to the primary model. When the server instance is going to become the primary HA server instance, then this program is called with a parameter of '1'. When the current HA primary server instance is no longer going to be the primary instance, then this program is called with a parameter of '0'. For example a program can be created which will start or end a job, depending on the function of the server.
LmIntervalTime	The LmIntervalTime directive is used by the Liveness Monitor to specify how often (in seconds, between performing Web server Liveness checks (HEAD or GET)) a liveness check should be performed on the server. The LmResponseTime and LmIntervalTime directives are independent. One sends out checks (LmIntervalTime), while the other tests for responses (LmResponseTime). The LmResponseTime value should always be larger than the LmIntervalTime value. It is recommended that LmResponseTime be at least 3 times larger than LmIntervalTime.
LmMaxReactivation	The LmMaxReactivation directive specifies how many times the Liveness Monitor should attempt to reactivate the Web server after a detected failure.
LmResponseTime	The LmResponseTime directive specifies how long the Liveness Monitor should wait for a response from the Web server before taking appropriate action (based on the other Liveness Monitor directive settings). The LmResponseTime and LmIntervalTime directives are independent. One sends out checks (LmIntervalTime), while the other tests for responses (LmResponseTime). The LmResponseTime value should always be larger than the LmIntervalTime value. It is recommended that LmResponseTime be at least 3 times larger than LmIntervalTime.
LmUrlCheck	The LmUrlCheck directive specifies a fully qualified URL that is used by the Liveness Monitor to perform liveness checks on HTTP Server. Specifying a domain name is not valid for this directive. Only one IP address can be specified in a highly available HTTP Server configuration.
LmUrlCheckBackup	The LmUrlCheckBackup directive specifies a fully qualified URL that is used by the Liveness Monitor to perform liveness checks on the HA backup server instance. If this directive is not configured, then the URL passed is the URL parameter value specified on the LmUrlCheck directive. This directive is ignored when HAModel PurePeer is configured.
¹ A LoadModule is required in the configuration file prior to using any of these directives. The statement should be as follows: LoadModule ha_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM	

A portion of a HTTP configuration file with high available directives is shown in Figure 10.1 on page 10-157:

Figure 10.1 Sample HTTP server configuration file with HA directives

```
LoadModule ha_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
HAModel PrimaryBackupWithIPTakeover
LmUrlCheck http://194.170.2.5:8000/web/docs/spec/wscheck.html
LmIntervalTime 20
LmMaxReactivation 3
LmResponseTime 60
```

Session Data

The *clustered hash table* (part of the state replication mechanism) is used by the Web server to replicate highly available CGI program state data across cluster nodes. That way the state data is available to all nodes in the event that a Web server fails (failover) or is switched-over manually (switchover). See “Clustered hash table” on page 72 for further information.

NOTE J2EE applications do not need to use the clustered hash table/replication mechanism since WebSphere takes care of that via WebSphere session management (see “Session Management” on page 102 for further information).

The data is stored in non-persistent storage. This means the data is retained only until the cluster node is no longer part of the clustered hash table, or the cluster is no longer active. See “Clustered hash table” on page 72 for further information.

For more information on how to enable a CGI program to use the clustered hash table to save state information, see “Enabling CGI Programs for High Availability” on page 161.

Highly Available Web Server Models

Three Web server cluster models are supported:

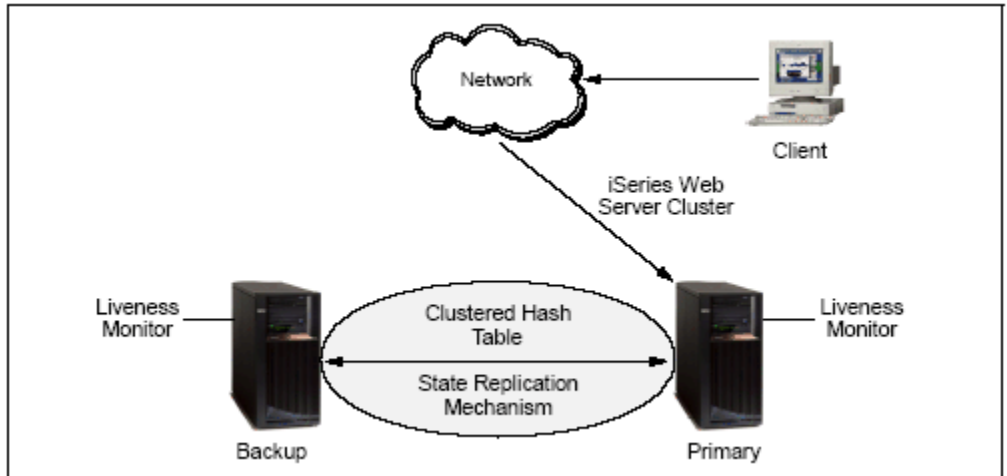
- Primary/backup with takeover Internet Protocol (IP) model
- Primary/backup with a network dispatcher model
- Peer model

Primary/Backup with Takeover IP Model

In this model, the Web server runs on the primary and all backup nodes. The backup node or nodes are in an idle state, ready to become the primary Web server should the primary Web server fail (failover) or a switchover takes place. All client requests are always served by the primary node.

Figure 10.2 on page 10-158 illustrates a primary/backup with takeover IP model.

Figure 10.2 Highly available Web server: primary/backup with takeover IP model



When the primary node fails (failover), or is brought down by the administrator, the failover/switchover process begins. The following steps are performed during failover/switchover:

- 1 One of the backup servers becomes the primary (the first backup in the switchover order).
- 2 Client requests are redirected to the new primary node. Assuming this client was not in the process of running a persistent CGI application, the failover is completely transparent.

TIP

You can provide a high availability (HA) environment with two iSeries servers, each with a one instance of the HTTP Server (powered by Apache). Each of these instances serves the identical (a copy) HTML and images and from identical (a copy) httpd.conf configuration files. You can do this with a simple configuration. You do not need third-party software. This is assuming, however, that you are not providing a HA environment for your CGI application.

- 3 If the new primary receives a user request that belongs to a long-running-session (a CGI program that is updated to be a highly available CGI program), the server restores the request's state. The new primary retrieves that highly available CGI program's state information from the clustered hash table. The clustered hash table is part of the state replication mechanism.

Most non-HA CGI applications behave in the following manner:

- a The client clicks the Submit button to send a new request to the Web server and your CGI application.
- b Your persistent CGI application “wakes up”. Its state information is saved in static variables to determine what happened in the past with this client and what to do now. Parameters found in the URL are parsed and actions are taken. New state information is saved in the static variables for the next time this client returns to this iSeries server. HTML code is generated and sent to standard out for presentation to the remote client.
- c The above “cycle” continues until the entire transaction is complete.

Most HA CGI applications behave in the following manner:

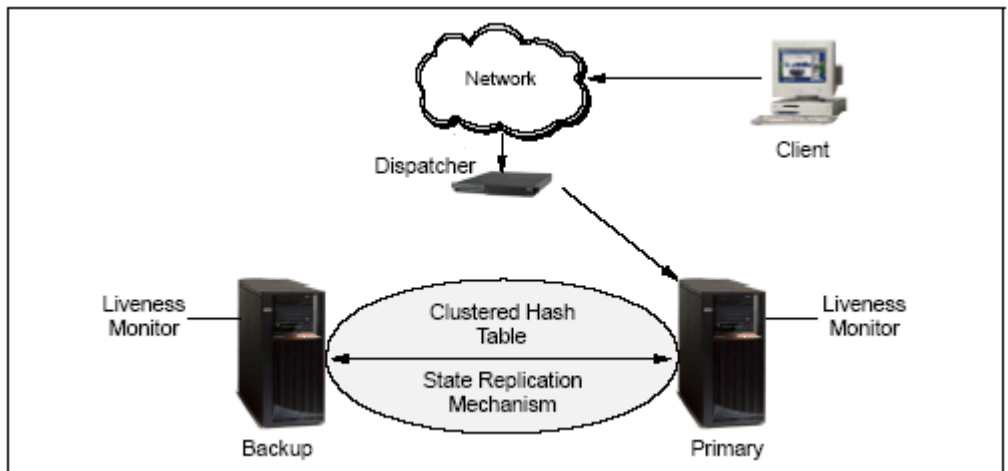
- a The client clicks the Submit button, which sends a new request to the Web server and your CGI application.
 - b Your persistent CGI application “wakes up”. Its state information is saved in the clustered hash table. It reads from the clustered hash table and updates local variables to determine what happened in the past with this client and what it must do now. Parameters found in the URL are parsed and actions are taken. New state information is written to the clustered hash table for the next time this same client returns to this (or the backup) iSeries server. HA support ensures that the information written to the clustered hash table on one iSeries server is replicated to the backup server. HTML code is generated and sent to standard out for presentation to the remote client.
 - c The above “cycle” continues until the entire transaction is complete.
- 4 After the failed node recovers, you can restart the highly available Web server instance, which then becomes the backup system. If the system administrator wants the failed node to become primary again, they must perform a manual switchover. They can accomplish this with the IBM Simple Cluster Management interface available through iSeries Navigator (Operations Navigator in V5R1), a 5250 CL interface, or a business partner tool.

Primary/Backup with a Dispatcher Model

In this model, as with the primary/backup with takeover IP model, the Web server runs on the primary and all backup nodes. The backup nodes are in an idle state and all client requests are served by the primary node. A network dispatcher sends client requests to the Web server.

Figure 10.3 on page 10-159 illustrates a primary/backup with a network dispatcher model.

Figure 10.3 Highly available Web server: primary/backup with a network dispatcher model.



When the primary node fails (failover), or a switchover takes place, the failover/ switchover process begins. The following steps are performed during failover/ switchover:

- 1 One of the backup servers becomes the primary (the first backup in the switchover order).
- 2 The client requests are sent to the new primary node by the network dispatcher.
- 3 If the new primary receives a user request that belongs to a long-running-session, the server needs to restore the request's state. The new primary searches for the state either locally or in the clustered hash table. The clustered hash table is part of the state replication mechanism.
- 4 After the failed node recovers, the system administrator can restart the Web server instance and it becomes a backup Web server. If the system administrator wants the failed node to become primary again, they must perform a manual switchover.

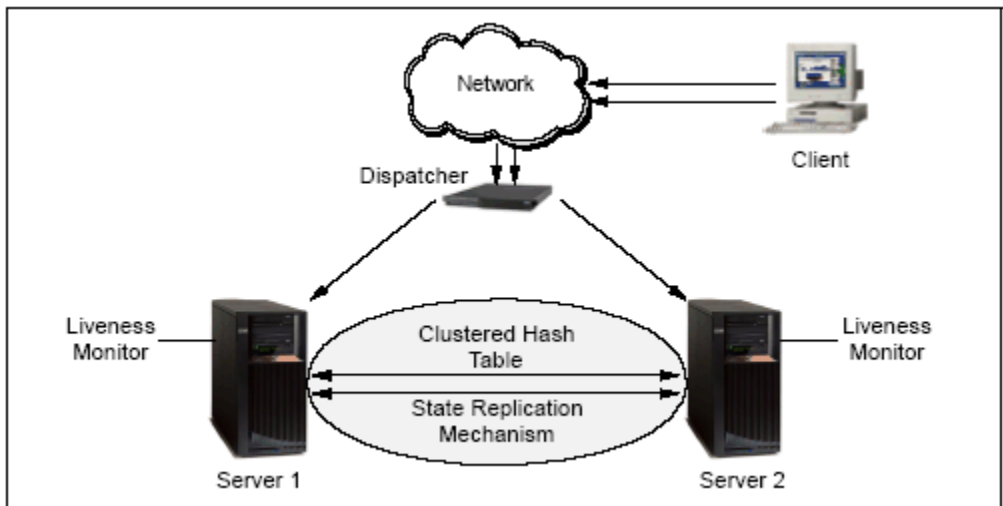
NOTE A node can join a recovery domain as a primary only if the cluster resource group is in inactive mode.

Peer Model

In this model, there is no declared primary node. All nodes are in an active state and serve client requests. A network dispatcher distributes requests to different cluster nodes. This guarantees distribution of resources in case of a heavy load. Linear scalability is not guaranteed beyond a small number of nodes. After some nodes are added, scalability can disappear, and the cluster performance can deteriorate.

Figure 10.4 on page 10-160 illustrates the peer model.

Figure 10.4 Highly available Web server: peer model



Model Considerations

For J2EE applications, the only useful model is the primary/backup with takeover IP model. The reason being that J2EE applications do not need to use the clustered hash table/replication mechanism since WebSphere takes care of that via WebSphere session

management. See “HTTP Session Failover” on page 172 for some considerations pertaining to session data.

Enabling CGI Programs for High Availability

High availability CGI programs use APIs to preserve the CGI’s state between successive client requests. A High availability CGI program stores its state data on the Web server and can retrieve its state even after a failure or switchover of the HTTP Server or system.

NOTE Although maintaining a CGI program’s state across multiple requests is a concept used by both Persistent CGI and High availability CGI programs, the mechanisms used by the two types of programs are significantly different and a High Availability CGI program should not be confused with a Persistent CGI program.

During the configuration of an Web server, the server administrator indicates whether CGI programs are allowed to be cluster-enabled High Availability CGI programs. If the server receives a request for a CGI program that is allowed to be highly available, the Web server passes to the CGI an environment variable that indicates the CGI may be cluster-enabled. The server also creates and passes a unique session handle to the CGI. The CGI program must then acknowledge that it is a cluster-enabled HA CGI program to the server, otherwise the server will regard the CGI as not being cluster-enabled.

The Web server associates a HA CGI program’s state with the unique session handle that was passed as an environment variable to the CGI. If a request to run the CGI is sent to the Web server, and the requested URL includes the specific session handle, the Web server will be able to correctly restore the previous state of the CGI. For this reason it is important that the session handle appear in all URLs that were generated by the HA CGI program to be returned to the client.

An HA CGI program will use two APIs to maintain its state. To store state information on the Web server, the CGI calls the API `QzhhCgiSendState()`. To receive its previous state from the Web server, the CGI should call the API `QzhhCgiRecvState()`.

Interaction Between HA CGI Program and HTTP Server

A Web server passes information to CGI programs using environment variables. If the HTTP Method is POST, the CGI program also obtains information from the Web Server through stdin. The CGI program uses stdout to send its response back to the Web server. The response consists of a set of headers (such as Content-Length and Content-Type) followed by the response body which is frequently HTML data.

The “Cluster-Enabled” and “Accept-HTSession” headers should be returned in each response from a HA CGI program.

```
Cluster-Enabled:1
```

An error will result if the “Cluster-Enabled” header is returned by a CGI program with a value of “1”, but the Web Server is not configured to allow that CGI program to be Highly available.

When the Web server receives the “Cluster-Enabled” header with a value of “1”, the server will create a new session entry and indicate that the session is cluster-enabled.

Cluster-enabled CGI programs will return the “Accept-HTSession” header to the Web server with a value equal to the value passed to the CGI in the `QZHBNEXT_SESSION_HANDLE` environment variable. An error will result if the value specified with “Accept-HTSession” does not match the value passed to the CGI in

QZHBNEXT_SESSION_HANDLE. For CGI programs that are not cluster-enabled, the “Accept-HTSession” CGI header remains unmodified.

In addition to the standard environment variable that are passed to a CGI program, the following environment variables are passed by the Web server to HA CGI programs:

Table 10.2 HA CGI environment variables set by Web server

Environment variable	Description
QZHBIS_FIRST_REQUEST	This environment variable indicates to a CGI program if this is a subsequent request of some session. The Web server sets this variable to 1 if this is not a subsequent request of any session (this is potentially the first request of a new session). The Web server sets this variable to 0 if this is a subsequent request of some session.
QZHBIS_CLUSTER_ENABLED	This environment variable indicates to the CGI program that the CGI program is allowed to be cluster-enabled if the request does not belong to any existing session (QZHBIS_FIRST_REQUEST is set to 1). This environment variable indicates to the CGI program that the CGI program is cluster-enabled (QZHBIS_FIRST_REQUEST set to 0). When the Web server receives a first request to a CGI, it decides if the CGI program is allowed to be cluster-enabled. If the CGI program is allowed to be cluster-enabled, the Web server sets the QZHBIS_CLUSTER_ENABLED environment variable to 1; otherwise the Web server does not define the QZHBIS_CLUSTER_ENABLED environment variable. When the Web server receives a subsequent request to a CGI, it looks to see if the session is cluster-enabled. If the session is cluster-enabled, the Web server sets the QZHBIS_CLUSTER_ENABLED environment variable to 1; otherwise the Web server does not define the QZHBIS_CLUSTER_ENABLED environment variable.
QZHBNEXT_SESSION_HANDLE	This environment variable contains a new session handle for a CGI program to use. If the CGI program is cluster-disabled, it may ignore this session handle. The Web server generates a session handle and sets the QZHBNEXT_SESSION_HANDLE environment variable to this value. If the CGI program decides to be cluster-enabled, it must use the passed session handle in the URLs of subsequent requests; otherwise, the Web server will not associate subsequent requests with this session.
QZHBRECOVERY	Contains whether the highly available Web server has gone through a recovery (primary to backup or backup to primary). If this environment variable is present, recovery has occurred. If it is not present, then recovery has not occurred
QZHBHA_MODEL	Model of the highly available Web server. Can be either PUREPEER or PRIMARYBACKUP.

All CGI programs existing today are cluster-disabled. A CGI program developer should follow the following rules when writing cluster-enabled CGI programs:

- Write the CGI in such a way that running them with the same state more than once does not cause any problem.
- Store the CGI program’s state between client requests only in the Web server.
- Use only cluster-safe mechanisms. For example, if a CGI program uses shared memory (which is currently not a cluster-safe mechanism), the program is almost certainly not suitable to be cluster-enabled.

There are two categories of high availability Web server programming models to consider when writing high availability CGI programs or enabling an existing CGI program for use as a high availability CGI program. The two categories are primary/backup model and the peer model. A HA CGI program can determine what model is being used by checking the QZHBHA_MODEL environment variable.

When running in a primary/backup model, an HA CGI program will save session information using the QzhbCgiSendState() API on every request. However, the same session data can be saved in static variables so that the retrieval of session data using the QzhbCgiSendState API is conditional on the value of the QZHBRECOVERY environment variable:

- If the value of the environment variable is NULL, the HA CGI program can continue to use the session information stored in the static variables. Not using the QzhbCgiRecvState() API saves time and resources.
- If the value of the environment variable is not NULL, then the HA CGI program must use the QzhbCgiRecvState() API to retrieve the session data and restore the static variables.

When running in a peer model, the HA CGI program must save and retrieve session data on every request using the QzhbCgiRecvState() and QzhbCgiSendState() APIs.

The Web server still limits the total number of persistent CGIs (both cluster-enabled and cluster-disabled) using the MaxPersistentCGI directive.

Resources for Learning

Use the following links to find relevant supplemental information about various topics discussed in this chapter. The information resides on IBM and non-IBM Internet sites, whose sponsors control the technical accuracy of the information.

These links are provided for convenience. Often, the information is useful all or in part for understanding a topic discussed in this chapter. When possible, links are provided to technical papers and Redbooks that supplement the broad coverage of the release documentation with in-depth examinations of particular product areas.

HTTP Server (powered by Apache)

- IBM HTTP Server for iSeries Home Page
<http://www.ibm.com/servers/eserver/series/software/http/>
The home page of IBM HTTP Server for iSeries.
- IBM HTTP Server for iSeries
<http://publib.boulder.ibm.com/series/v5r2/ic2924/index.htm?info/rzaie/rzaie.htm>
This HTTP Server Information Center Web page contains links to online and printable product documentation for the HTTP Server.
- *HTTP Server (powered by Apache): An Integrated Solution for IBM iSeries Servers* (October 2003)
<http://www.redbooks.ibm.com/redbooks/pdfs/sg246716.pdf>

This IBM Redbook helps you to plan, install, configure, troubleshoot, and understand the HTTP Server (powered by Apache) running on the IBM iSeries server.

iSeries clusters

- Clusters

<http://publib.boulder.ibm.com/series/v5r2/ic2924/index.htm?info/rzaig/rzaigiclust.htm>

This iSeries Information Center topic contains information on iSeries clusters.

CHAPTER 11

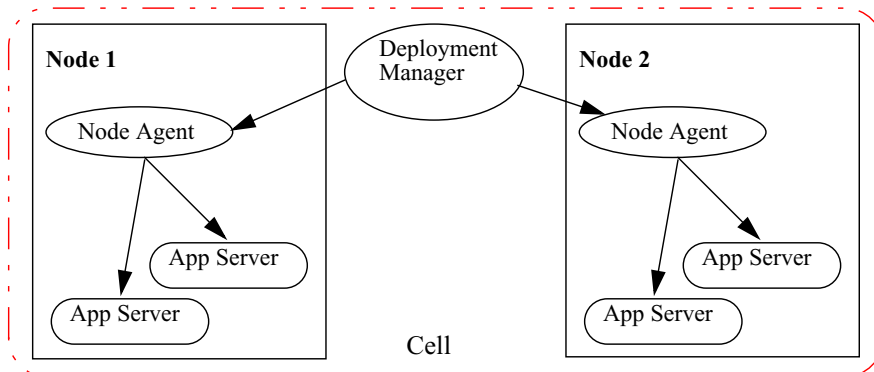
WEBSPHERE APPLICATION SERVER CONSIDERATIONS FOR HIGH AVAILABILITY

WebSphere system availability includes WebSphere Application Server and administrative server process availability. WebSphere server process availability is achieved through the Work Load Management (WLM) mechanism. This chapter will cover how application server process availability is handled by WebSphere and what things need to be taken into consideration from an application point of view, in addition to discussing deployment manager availability.

Overview

As discussed in earlier chapters, a cell is a collection of nodes, managed as a unit via a Deployment Manager. From a WebSphere system management perspective, a node is a logical box such as a LPAR partition coupled with a particular WebSphere base instance on that node. A node can have one or more application servers associated with it all administered with the Network Deployment manager.

Figure 11.1 A WebSphere cell with multiple nodes and application servers



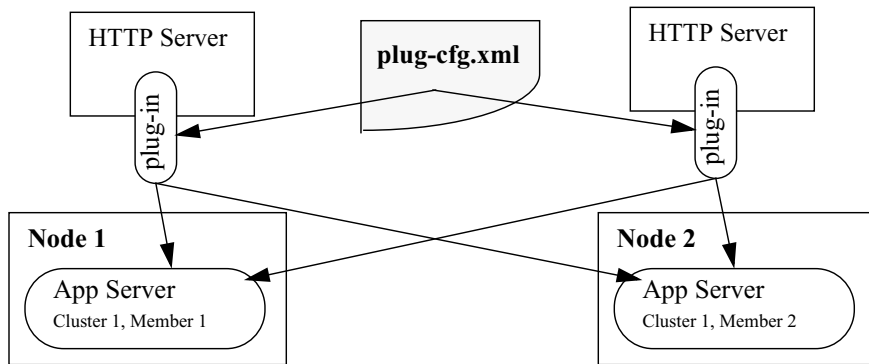
An application server is a JVM (Java process) which is capable of having one or more J2EE applications deployed to it. A Node Agent process located on the node works in conjunction with the Network deployment manager to provide administration capabilities. Neither the Deployment Managers nor associated Node Agents need to be active to run the contained WebSphere application servers. In other words, the Network Deployment Manager and Node Agent provide only administrative capabilities.

A WebSphere cluster is a collection of application servers running identical applications. Clustering along with WebSphere HTTP workload management code, provide the clients of these application both load balancing and failover amongst the cluster members. The Network Deployment Manager allows clusters to be defined and administered across all nodes within the cell. When an application is installed on a cluster, the application gets automatically copied to all cluster members via the node agents.

The members of a cluster can be located on a single node (vertical cluster), across multiple nodes (horizontal cluster) or on a combination of the two.

Once the cluster is created and applications installed, the Network Deployment Manager is used to generate the plugin-cfg.xml file for the cluster. As depicted in Figure 11.2 on 166, the WebSphere plug-in code takes the plugin-cfg.xml file as input and acts as a router, deciding what traffic is for WebSphere cluster members and what traffic should be handled by the Web server itself. The plug-in code is able to route requests to all cluster members contained within the cluster.

Figure 11.2 Plug-in using plugin-cfg.xml file to route requests to application servers



You may have thin Web clients or/and thick Java/C++ clients. When using clustered WebSphere Application Servers, your clients can be redirected either automatically or manually (depending on the nature of the failure) to another healthy server in the case of a failure of a clustered application server.

When an EJB client makes calls from the Web container or client container or from outside, the request is handled by the EJB container in one of the clustered application servers. If that server fails, the client request is redirected to another available server.

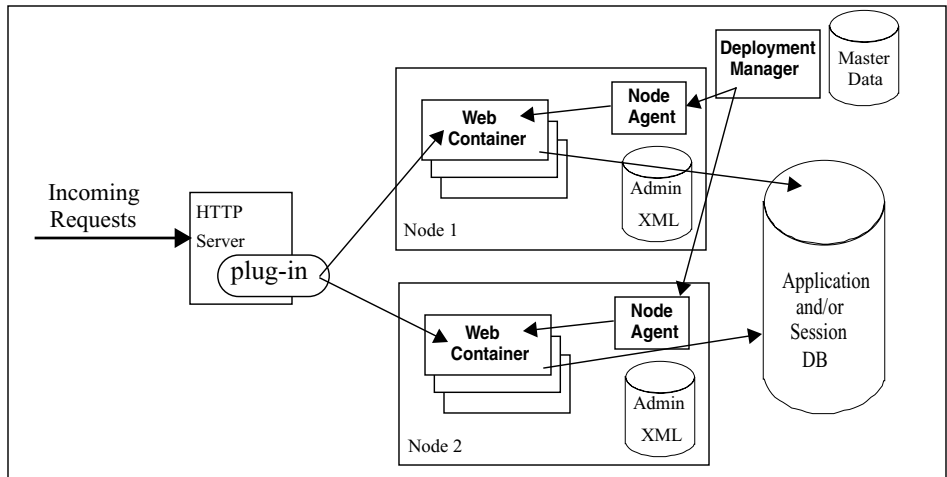
If you don't workload manage your WebSphere Application Servers and you don't use any other clustering software, your system will not provide any failover support. All of your Web or Java clients will fail if your WebSphere Application Server fails.

If you workload manage your WebSphere cluster servers, your system provides satisfactory failover support for most cases.

Web Container Clustering and Failover

A WebSphere environment may include several HTTP server instances. Each HTTP server is configured to run the WebSphere HTTP plug-in in its process. Each request coming into the Web server is passed through this plug-in, which uses its configuration information to determine if the request should be routed to WebSphere, and if so, which Web container the request should be routed to (see Figure 11.3 on page 167). These Web containers may be running on the same machine as the HTTP server, a different machine, or a combination of the two.

Figure 11.3 WebSphere Application Server (Web container) high availability



The Web container failover support in WebSphere is provided by three mechanisms:

- WebSphere Web container server cluster, which creates server process redundancy for failover support.
- The workload management routing technique built into the WebSphere Web server plug-in. It controls the routing of client requests among redundant server processes.
- Session management and failover mechanism, which provides HTTP session data for redundant server processes.

When a Web container fails, it is the responsibility of the HTTP server plug-in to detect this failure and mark the Web container unavailable. Web container failures are detected based on the TCP response values, or lack of response, to a plug-in request. There are five types of failover scenarios for Web containers:

- Expected server process failures, for example after stopping the server.
- Unexpected server process failures, for example the server JVM crashes.

- Server network problems, for example the network cable is disconnected or a router is broken.
- Unexpected and expected system problems, for example a system shutdown, operating system crashes, or power failures.
- Overloading of Web clients, for example a denial of service attack, system is too small to handle a large number of clients, or server weight is inappropriate.

In the first two cases, the physical machine where the Web container is supposed to be running will still be available, although the Web container port will not be available. When the plug-in attempts to connect to the Web container port to process a request for a Web resource, the machine will refuse the connection, causing the plug-in to mark the application server as down.

In the third and fourth events, however, the physical machine is no longer available to provide any kind of response. In these events, if non-blocking connection is not enabled, the plug-in will wait for the local operating system to time out the request before marking the application server unavailable. While the plug-in is waiting for this connection to time out, requests routed to the failed application server appear to hang. The default value for the TCP timeout varies based on the operating system. While these values can be modified at the operating system level, adjustments should be made with great care. Modifications may result in unintended consequences in both WebSphere and other network dependent applications running on the machine. This problem can be eliminated by enabling non-blocking connection. Please refer to “ConnectTimeout Setting” on page 169 for more information.

In the fifth case, client overloading can make a healthy server unavailable and cause a server overloading failover. This is explained in “Stream and Overloading Failover” on page 170.

RetryInterval and Operating System TCP Time-out

If a request to an application server in a cluster fails, and there are other application servers in the group, the plug-in will transparently reroute the failed request to the next application server in the routing algorithm. The unresponsive application server is marked unavailable and all new requests will be routed to the other application servers in the server cluster.

The amount of time the application server remains unavailable after a failure is configured by the `RetryInterval` property on the `<ServerGroup>` attribute. If this attribute is not present, the default value is 60 seconds.

```
<ServerCluster Name="HACluster" RetryInterval="1200">
```

This would mean that if a cluster member were marked as down, the plug-in would not retry it for 1200 seconds.

When the `RetryInterval` expires, the plug-in will add the application server back into the routing algorithm and attempt to send a request to it. If the request fails or times out, the application server is again marked unavailable for the length of the `RetryInterval`.

The proper setting for the `RetryInterval` will depend on your environment, particularly the value of the operating system TCP timeout value and how many application servers are configured in the cluster. Setting the `RetryInterval` to a small value will allow an application server that becomes available to quickly begin serving requests. However, too small of a value can cause serious performance degradation, or even cause your plug-in to appear to stop serving requests, particularly in a machine outage situation.

To explain how this can happen, let's look at an example configuration with two machines, which we will call A and B. Each of these machines is running two clustered application servers (CM1 and CM2 on A, CM3 and CM4 on B). The HTTP server and plug-in are running on an iSeries system with a TCP timeout of 75 seconds, the `RetryInterval` is set to 60 seconds, and the routing algorithm is weighted round robin. If machine A fails, either expectedly or unexpectedly, the following process occurs when a request comes in to the plug-in:

- 1 The plug-in accepts the request from the HTTP server and determines the server cluster.
- 2 The plug-in determines that the request should be routed to cluster member CM1 on system A.
- 3 The plug-in attempts to connect to CM1 on machine A. Because the physical machine is down, the plug-in waits 75 seconds for the operating system TCP timeout interval before determining that CM1 is unavailable.
- 4 The plug-in attempts to route the same request to the next cluster member in its routing algorithm, CM2 on machine A. Because machine A is still down, the plug-in must again wait 75 seconds for the operating system TCP timeout interval before determining that CM2 is also unavailable.
- 5 The plug-in attempts to route the same request to the next cluster member in its routing algorithm, CM3 on system B. This application server successfully returns a response to the client, over 150 seconds after the request was first submitted.
- 6 While the plug-in was waiting for the response from CM2 on system A, the 60-second `RetryInterval` for CM1 on system A expired, and the cluster member is added back into the routing algorithm. A new request will soon be routed to this cluster member, which is still unavailable, and this lengthy waiting process will begin again.

There is no way to recommend one specific value; the value chosen depends on your environment. For example, if you have numerous cluster members, and one cluster member being unavailable does not affect the performance of your application, then you can safely set the value to a very high number. Alternatively, if your optimum load has been calculated assuming all cluster members to be available or if you do not have very many, then you will want your cluster members to be retried more often to maintain the load. An option is to configure your application servers to use a non-blocking connection. This eliminates the impact of the operating system TCP/IP timeout. “`ConnectTimeout` Setting” on page 169 explains how to configure this option.

An important thing to take into consideration when coming up `RetryInterval` value is the the time it takes to restart your server. If a server takes a long time to boot up and load applications, then you will need a longer retry interval.

ConnectTimeout Setting

When a cluster member exists on a machine that is removed from the network (because its network cable is unplugged or it has been powered off, for example), the plug-in, by default, cannot determine the cluster member's status until the operating system TCP/IP timeout expires. Only then will the plug-in be able to forward the request to another available cluster member.

It is not possible to change the operating system timeout value without side effects. For instance, it might make sense to change this value to a low setting so that the plug-in can fail over quickly. However, the timeout value on some of the operating systems is not

only used for outgoing traffic (from Web server to application server) but also for incoming traffic. This means that any changes to this value will also change the time it takes for clients to connect to your Web server. If clients are using dial-up or slow connections, and you set this value lower, they will not be able to connect.

In any case, if you are running the Apache server on OS/400 then the operating system connection timeout value is 2 minutes and this value cannot be changed from an OS/400 operating system point of view, but can from a plug-in perspective.

WebSphere offers an option within the plug-in configuration file that allows you to bypass the operating system timeout. It is possible to add an attribute to the Server element called `ConnectTimeout`, which makes the plug-in use a non-blocking connect. Setting `ConnectTimeout` to a value of 0 is equal to not specifying the `ConnectTimeout` attribute, that is, the plug-in performs a blocking connect and waits until the operating system times out. Set this attribute to an integer value greater than zero to determine how long the plug-in should wait for a response when attempting to connect to a server. A setting of 10 will mean that the plug-in waits for 10 seconds to time out. An example is shown below:

```
<Server CloneID="v60aiiv5" ConnectTimeout="30"
      Name="rchas840_CLONE839_CLONE839">
```

To determine what setting should be used, you need to take into consideration how fast your network and servers are. Complete some testing to see how fast your network is, and take into account peak network traffic and peak server usage. If the server cannot respond before the `ConnectTimeout`, the plug-in will mark it as down.

Since this setting is determined on the Server tag, you can set it for each individual cluster member. For instance, you have a system with four cluster members, two of which are on a remote node. The remote node is on another subnet and it sometimes takes longer for the network traffic to reach it.

Network Failures and the Node Agent

When the network becomes unavailable, the Node Agent, if active, will force its managed application servers to stop if the loopback is configured for 127.0.0.1. If the network becomes available within 10 minutes (which is the timeout value for the Node Agent), the Node Agent can restart the stopped servers. If the network is unavailable beyond the 10 minutes downtime, the application servers will remain down and you will need to manually start the application servers.

To resolve this problem, configure a loopback alias to a systems' real IP address, not the default loopback of 127.0.0.1. After a short-time network outage, the Node Agent can restart servers automatically.

Stream and Overloading Failover

WebSphere Application Server V5.0 is designed to handle a large number of client requests concurrently. However, sometimes overloading a server can cause client requests to fail even though the server is really healthy. It is difficult to catch overloading failures because of the differences between a test environment and a production system. Thus, you need to carefully test your application with the expected load.

WebSphere V5 provides an embedded HTTP transport in its Web container that is connected to the external HTTP server through the plug-in. An open connection between the Web server and the Web container is called a *stream*. If there are more connections than available threads, the connections start to backlog, waiting for free threads. If the maximum number of backlog connections is reached, new connections will be refused. The plug-in will treat this healthy server as a failed server, and mark it as down. This leads to an overloading failover.

There are several parameters you can tune to reduce overloading failovers:

- Connection backlog
- Maximum keep-alive connections
- Maximum requests per keep-alive connection
- Keep-alive timeout
- I/O timeout
- Minimum thread size
- Maximum thread size
- Thread inactivity timeout

Please refer to the WebSphere InfoCenter for more information on these parameters.

It is important to stress once again that performance tuning is not an exact science. Factors that influence testing vary from application to application, and also from platform to platform.

Primary and Backup Servers Cluster Two-level Failover

WebSphere supplies a two-level failover functionality. When the plugin-cfg.xml is generated, all servers are initially listed under the PrimaryServers tag. You can manually move selected servers to be listed under the BackupServers tag, as shown below:

```
<PrimaryServers>
  <Server Name="HAClusterServer1" />
  <Server Name="HAClusterServer3" />
</PrimaryServers>
<BackupServers>
  <Server Name="HAClusterServer2" />
  <Server Name="HAClusterServer4" />
</BackupServers>
```

The Web server plug-in will not route requests to any server in the Backup Server list as long as there are application servers available from the Primary Server list. Within the PrimaryServers, the plug-in routes traffic according to server weight and/or session affinity. When all servers in the Primary Server list are unavailable, the plug-in will route traffic to the first available server in the Backup Server list. If the first server in the Backup Server list is not available, the request will be routed to the next server in the Backup Server list. Weighted round robin routing is not performed for the servers in the Backup Server list. The plug-in just uses one server at a time.

The following steps are taken by the plug-in to determine primary and backup server selection:

- 1 The request comes in and is sent to the plug-in.
- 2 The plug-in chooses the next primary server, checks whether the cluster member has been marked down and the retry interval. If it has been marked down and the retry timer is not 0, it continues to the next cluster member.
- 3 A stream is opened to the application server (if not already open) and a connection is attempted.
- 4 The connection to the cluster member fails. When a connection to a cluster member has failed, the process begins again.
- 5 The plug-in repeats steps 2, 3 and 4 until all primary servers are marked down.
- 6 When all primary servers are marked as down, the plug-in will then repeat steps 2 and 3 with the backup server list.
- 7 It performs steps 2 and 3 with a successful connection to the backup server. Data is sent to the Web container and is then returned to the user. If the connection is unsuccessful, the plug-in will then go through all the primary servers again and then all the servers marked down in the backup server list until it reaches a backup server that is not marked down.
- 8 If another requests comes in and one of the primary server's retry timer is at 0 the plug-in will try and connect to it.

This two-level failover mechanism provides more advanced failover support, allowing better use of available server capacity.

HTTP Session Failover

HTTP session objects can be used within a Web application to maintain information about a client across multiple HTTP requests. For example, on an online shopping Web site the Web application needs to maintain information about what each client has placed in its shopping cart. The session information is stored on the server, and a unique identifier for the session is sent back to the client as a cookie or through the URL rewriting mechanism.

So how does the plug-in route requests to the proper server? When the server is created, WebSphere assigns a unique CloneID for each server, the CloneID is shown in the plugin-cfg.xml file:

```
<ServerCluster Name="HACluster">
<Server CloneID="u307p2vq" LoadBalanceWeight="2" Name="HAClusterServer1"></Server>
<Server CloneID="u307p48r" LoadBalanceWeight="3" Name="HAClusterServer2"></Server>
<Server CloneID="u307p62u" LoadBalanceWeight="4"
Name="HAClusterServer3"></Server>
<Server CloneID="u307p7lm" LoadBalanceWeight="5"
Name="HAClusterServer4"></Server>
</ServerCluster>
```

The CloneID is used to create the serverGroup object. The plug-in will bind the CloneID and a client's session ID (the session ID contains includes the CloneID, see

“Session Identifiers” on page 105 for further information) when it receives the client's first request.

To achieve session affinity, the plug-in will parse the client's session information, extract the previous CloneID, and match this CloneID with the CloneIDs in the server cluster object. When the matching cluster member is found, the server will retrieve this client's session from the in-memory cache. When the matched application server is not available, the plug-in will route the client to another server, and appends the new server's CloneID into the client's session.

Session Data Considerations

The critical point that cannot be over emphasized is that HTTP session data must be serialized properly in order for session failover to occur successfully.

Session ID Considerations

The HTTP protocol itself is stateless and it cannot carry stateful information for subsequent requests by itself. WebSphere can work around this HTTP protocol defect and provide session support as well as session failover support.

WebSphere keeps the user's session information on the server, using different options:

- In-memory, which provides no failover support.
- Into a database or into message brokers, which provides failover support.

WebSphere recognizes the user's session information in three ways:

- SSL session ID
- Cookies
- URL rewriting

Please refer to “Session Management” on page 102 for more information.

Session Affinity and Failover

WLM works different with session affinity. We will discuss the following options:

- No session support needed
- No session information in the client
- Session affinity without session persistence
- Session persistence and failover

No Session Support Needed

By default, session affinity is enabled in WebSphere. If none of your applications needs session support, you can remove the CloneID in the plugin-cfg.xml file to make your request processing faster. Doing so eliminates the time-consuming comparison process:

```
<ServerCluster Name="HACluster">  
<Server CloneID="u307p2vg" LoadBalanceWeight="2"
```

```

    Name="HAClusterServer1">
</Server>
<Server CloneID="u307p48r" LoadBalanceWeight="3"
    Name="HAClusterServer2">
</Server>
<Server CloneID="u307p62u" LoadBalanceWeight="4"
    Name="HAClusterServer3">
</Server>
<Server CloneID="u307p71m" LoadBalanceWeight="5"
    Name="HAClusterServer4">
</Server>
</ServerCluster>

```

However, if any application in your application servers is stateful, you will lose all state information during a failover. Remember that this setting is for the whole server cluster, not for a single Web application. It is not necessary to disable the server session support for session-less clients.

No Session Information in the Client

When server session affinity is enabled, the plug-in will look for session information in each incoming request. If the plug-in fails to find any session information from the incoming client, the plug-in will assume that no session affinity is needed for this client. The plug-in will ignore session affinity and route this client using weighted round robin.

Session Affinity Without Session Persistence

When server session support is enabled and an incoming client request has session information, HTTP sessions are kept in the memory of the application server containing the Web application. The plug-in will route multiple requests for the same HTTP session to the same server by examining the CloneID information that is stored with the session key. Thus, the session is kept for a client during the following requests.

The plug-in will route the client to an alternative server in case the original server is not available. If the alternative server cannot retrieve the session information, all session information is lost and the client needs to start over again. The failover process fails, but the client can continue without the previous session information.

Session Persistence and Failover

WebSphere offers three different methods to persist sessions:

- In-memory

In-memory session management provides the best performance for HTTP requests as long as the user's requests are always directed to the same server and this server never fails.

The local in-memory session manager doesn't persist the session in the host. Thus, when the host becomes unavailable, all session information is lost and when the server becomes available again, it cannot recover the previous session.

The local in-memory session manager doesn't share session information with other servers in the cluster. Thus, if the server fails, the adoptive server (another server in the cluster) cannot retrieve the session information, and the failover will fail.

- Persistent to database

If session data is persisted to a database and the application server fails, another server can retrieve the session from the database. Therefore, the client request can continue to be processed.

If the database service is not available, the adoptive server cannot retrieve the session from the database. The WebSphere session manager will retry for a while and the client appears to hang. If the database is still not available, the failover process will fail. Therefore, the session database is a single point of failure. We discuss how to enable database high availability in “Data Considerations for High Availability” on page 209.

- Memory-to-memory

Memory-to-memory replication is done using WebSphere Data Replication Service (DRS). There are three topological models for memory-to-memory replication:

- Buddy system with single replica
- Dedicated replication server
- Peer-to-peer replication (default)

If the original server fails, the plug-in will route the user's request to another server in the server cluster (adoptive server). The session manager in the adoptive server will look whether there is a copy of the user's session locally. If not, it will go to another store to retrieve this user's session.

Considerations for configuring the replication topology are:

- At least two replication stores are needed for high availability in case a single replicated copy is also unavailable
- The message complexity of session replication should be minimized

While the buddy system and the dedicated replication server may have fewer replication messages, they are also prone to failures if many servers fail at the same time. The peer-to-peer or client/server models provide more resistance to multiple server failures.

The HTTP WLM plug-in doesn't know which servers have the replicated session locally. It simply routes the user's request to an adoptive server in the cluster, according to server weights, without any session store consideration. With the peer-to-peer model, the session information is distributed across all application servers, that is, each application server retrieves sessions from other application servers, and each application server provides sessions to other application servers. Therefore, no matter which server the HTTP plug-in routes the user's request to, it can always get the session locally.

There are some moderate performance advantages to using memory versus database. On the other hand, for large HTTP session objects, memory to memory can consume large amounts of memory in networks with many users, since each cluster member has a copy of all sessions. For example, assuming that a single session consumes 10 KB and 100,000 users have logged into the system, each cluster member consumes 1 GB of memory in order to keep all sessions in its own memory. Another disadvantage is that every change to a session must be replicated to all cluster members. On OS/400 we recommend the use of database for sessions along with a write frequency of moderate (see “Time-based (at a specified time interval)” on page 176 for information on the write frequency).

How much session information might be lost during a failover depends on the frequency of the session persistence, which is configurable on the application server from the Administrative Console. This is discussed in the following section.

Session Update Methods and Failover Session Data Loss

The WebSphere session manager drives session replication and persistence. WebSphere has three methods to trigger the session data persistence, each of which has different failover impacts. The fewer the updates and the longer the update interval, the better the server performance, but the more session data is lost in case of a failover.

At the End of Servlet Service

When the write frequency is set to the end of servlet service, WebSphere will write/publish the session into database/replication stores at the completion of the `HttpServletRequest.service()` method call. All new session data between two calls is lost after a failover. The adoptive server will retrieve the session data from the last `HttpServletRequest.service()` method call.

Manually sync() in the Code Using the IBM Extension

IBM extended the Servlet specification to add the functionality of letting an application decide when to update its session data. The application needs to invoke the `sync()` method explicitly to write/publish the session into database/replication stores. All new session data since the last invoke of `sync()` is lost after a failover.

Time-based (at a specified time interval)


The session is written/published into database/replication stores at the preset time interval. All new session data since the last update is lost during a failover.

WebSphere offers several choices for setting the time interval using the Administrative Console as shown in Figure 11.4 on page 177.

Figure 11.4 WebSphere HTTP session failover optimization


[Application Servers](#) > [BASE839](#) > [Web Container](#) > [Session Management](#) > [Distributed Environment Se](#)

Tuning Parameters

Session Manager provides various tuning options for managing session data in a distributed environment. 

Configuration

General Properties

Tuning level:	<p><input type="radio"/> Very high (optimize for performance) Write frequency Time based: 300 seconds Write contents Only updated attributes Schedule sessions cleanup: true</p> <p><input type="radio"/> High Write frequency Time based: 300 seconds Write contents All session attributes Schedule sessions cleanup: false</p> <p><input checked="" type="radio"/> Medium Write frequency End of servlet service Write contents Only updated attributes Schedule sessions cleanup: false</p> <p><input type="radio"/> Low (optimize for failover) Write frequency End of servlet service Write contents All session attributes Schedule sessions cleanup: false</p> <p><input type="radio"/> Custom settings Write frequency Time based: 10 seconds Write contents Only updated attributes Schedule sessions cleanup: false</p>	<p> The Session Manager predefined settings optimize performance.</p>
---------------	--	--

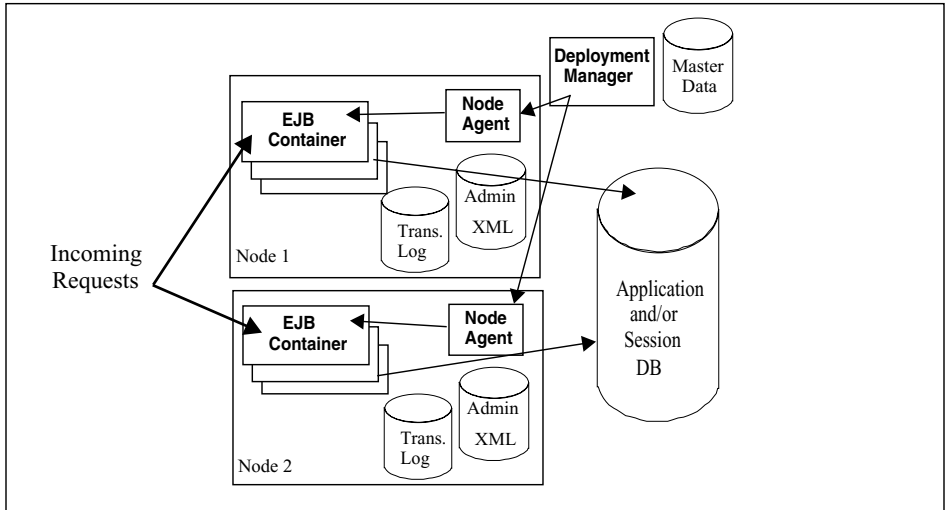
For OS/400, we recommend the write frequency set to medium.

EJB Container Clustering and Failover

Many J2EE applications rely on Enterprise JavaBeans (EJBs) to provide business logic. EJB clients can be servlets, JSPs, J2EE client, stand-alone Java applications, or even other EJBs. The approach for EJB failover in WebSphere is to have server redundancy (by using WebSphere server cluster support) and smart routing techniques, subject to the constraints of resource availability, data integrity, and transaction and process affinity.

As discussed earlier, WebSphere server clusters allow multiple instances of an application server to be created from a template. These multiple application servers, or cluster members, have a single administration point and the ability to share workload. Workload management for EJBs is enabled automatically when a cluster is created within a cell. Figure 11.5 on page 178 shows horizontal and vertical scaling being used to have application server redundancy to tolerate possible process and machine failures.

Figure 11.5 WebSphere EJB container failover



The mechanisms for routing workload managed EJB requests to multiple cluster members are handled on the client side of the application. In WebSphere, this functionality is supplied by a workload management plug-in to the client ORB and the routing table in Location Service Daemon (LSD) hosted in the Node Agent. The WLM failover support for EJBs is to maintain the routing table and to modify the client ORB to redirect traffic in case of a server failure.

There are the following possible failures:

- Expected server process failures, for example stopping the server.
- Unexpected server process failures, for example the server JVM crashes.
- Server network problems, for example a network cable is disconnected or a router is broken.
- Unexpected and expected machine problems, for example a system shutdown, operating system crashes, or power failures.
- Overloading of EJB clients, for example a denial of service attack, the system is too small to handle a large number of clients, or the server weight is inappropriate.

We discuss these issues in the following sections.

EJB Client Redundancy and Bootstrap Failover Support

The first thing that is done by any EJB client is to look up the home of the bean (except message-driven beans). Java clients use the Java Naming Directory Interface (JNDI) to obtain references to objects within an EJB container, such as enterprise bean homes. Within the application server these objects are bound in a hierarchical structure called a name space. A Java client uses an InitialContext to acquire access to objects within the name space. To connect to a name space, the client must supply a host name and port. If the host name and port are not specified, the default values are localhost and 2809, respectively.

In WebSphere Application Server Network Deployment V5.0, there are separate name spaces for the cell, each node agent, each application server and the deployment manager. Because node agents are not clustered, they should not be used as the bootstrap location in a clustered environment. For failover purposes it is recommended that a client bootstrap into a clustered EJB application server.

The application server bootstrap host and port are configurable and can be found or changed using the Administrative Console. To find the bootstrap host and port, use the Administrative console and navigate to **Application Servers -> servername -> End Points -> host and port fields**.

Keep in mind that InitialContext requests participate in workload management when the provider URL is a clustered resource (cluster member). Bootstrapping into the name space and the positioning of the JNDI InitialContext are controlled by URLs (iiop, corbaloc and corbaname URLs) and other properties that are passed on the creation of the InitialContext. The URLs used to identify the server in which to bootstrap can contain multiple host and port addresses. This eliminates single points of failure when bootstrapping into a name space that is part of an EJB application server cluster. If one host and port is down then another will be automatically attempted.

There are two kinds of EJB clients need to be considered:

- Multiple copies of the same EJB client may exist in a client request chain.

For example, Web clients from Web containers that are workload managed, EJB clients from another EJB container server cluster, or EJB clients from their own server cluster. In this case, EJB clients can use their own server to bootstrap with the default provider URL. If the bootstrap fails, the EJB client fails. This failure should be handled by the previous server, for example the HTTP plug-in. Another version of the same client in a different container may bootstrap from its server successfully. By using client redundancy, EJB failover and high availability is achieved.

- Only one copy of the same client.

For example, a Java client, J2EE client, C++ client, or third-party ORB client. In this case, if the client is bootstrapped to only one server, it will fail if that server fails since the client is not redundant. You need to bootstrap this client to as many bootstrap servers as possible. The following snippet of code is an example of an InitialContext lookup using multiple hosts and ports using a CORBA object:

```
import java.util.Hashtable;
import javax.naming.Context;
import javax.naming.InitialContext;
Hashtable prop = new Hashtable();
prop.put(Context.INITIAL_CONTEXT_FACTORY,
"com.ibm.websphere.naming.WsnInitialContextFactory");
```

```

prop.put(Context.PROVIDER_URL, "corbaloc::host1:9810, :host2:9810");
Context initialContext = new InitialContext(prop);
try { java.lang.Object myHome = initialContext.lookup("MyEJB");}

```

In addition to the CORBA object URL as specified in J2EE 1.3, WebSphere also supports IIOP URL. The IIOP type of URL is a legacy JNDI format which is not as flexible as CORBA object URLs. However, URLs of this type are still supported. Here is an example:

```

Properties prop;
prop.put(javax.naming.Context.PROVIDER_URL, "iiop://
Server1.xyz.ibm.com:9000")
InitialContext c = new InitialContext(prop);

```

Note that only one host/port combination may be specified. This limitation may be overcome by enhancing the code to monitor for a naming exception when performing the bootstrap operation. If an exception is caught, then the application code can retry one of the other clustered servers host/port combinations. The following example uses an array hostURL that contains the bootstrap server and port for three clustered EJB application servers: Server1.xyz.ibm.com:9000, Server2.xyz.ibm.com:9000, and Server3.xyz.ibm.com:9000:

```

Properties prop;
int i = 0;
while ( i < hostURL.length)
{
    try
    {
        prop.put(javax.naming.Context.PROVIDER_URL, "iiop://" + hostURL[i]);
        c = new InitialContext(prop);
        break;
    }
    catch(NamingException ex)
    {
        if ( i == (hostURL.length - 1)) {
            System.out.println("All servers are not available, program terminating...");
            System.exit(1);
        }
        else {
            System.out.println("Server" + hostURL[i] + "is not available, will use next one");
            i++;
        }
    }
}

```

Once you look up the EJB home, the naming server will return an indirect Interoperable Object Reference (IOR), LSD, and routing table, and the WLM plug-in will redirect the client to one of many clustered EJB containers.

EJB WLM Routing

The implementation of the routing pattern for IIOP (the communications protocol used to send messages between two ORBs) requests to an application server is a combination of the two client routing patterns: client and client with advisor. This implementation is to support both those clients with the distributed platform WebSphere code and also those clients that support the IIOP interface without the WebSphere extensions.

It is expected that the non-WebSphere distributed clients will consist of WebSphere for z/OS, CORBA C++, or non-IBM ORBs. The WebSphere distributed clients are cluster aware and have WLM support built into the client. This support has knowledge of the servers in the cluster and implements the routing policy resulting in the client making the routing decision and directing the request to a specific server. The non-WebSphere distributed clients are cluster unaware clients and do not have any WLM routing or cluster knowledge and must be given a specific server to direct the client's request to. In this case we employ a LSD, which acts as an advisor to a client, directing incoming requests to the appropriate server.

Within the LSD approach, a client obtains an indirect IOR (possibly from a lookup in JNDI, or as a return argument) which contains routing information to the LSD rather than to the application server. The client request is then sent to the LSD to determine a direct IOR to be used for the object request. As part of the resolution of the IOR, some WLM logic is added to the LSD. The LSD's ORB will contain hooks that will be used in the WLM code to determine how to handle the request.

The following sections describe the different functions based on the client type.

Cluster Aware Clients

If the client contains the distributed WebSphere WLM code and has yet to obtain the initial routing information, the initial request will flow to the LSD. The WLM director code in the LSD will determine the correct server and return the direct IOR to the client. This direct IOR will then be used to route the request to the specific server. However, since this client contains the distributed WebSphere WLM code, the request to the server will contain WLM service context data, and the WLM code on the server will look at this context data to determine whether the client has back-level cluster information. If required, the WLM code on the server adds cluster information into the service context list on the response back to the client. The WLM code on the client then updates its cluster information (if newer data was available). From this point on, the WLM code on the client has updated cluster information. Thus for all subsequent requests the client has the capability to choose the specific server to handle the request. This client-based routing is basically how Java EJB clients work in WebSphere today.

Cluster Unaware Clients

If the client does not have the distributed WLM code active, the LSD will invoke the WLM director where it is the director's responsibility to handle all of the WLM policy and affinity decisions that need to be made. The client does not have any knowledge of the cluster, so the LSD must determine the specific server the request will be sent to. This direct IOR will then be returned to the client and be used to invoke all requests on the object. What this means is that every request the client makes on this object will go to the server that the LSD determined. This is different from cluster aware clients, which will determine what server to use for every request. For cluster unaware clients, a load balancing mechanism would be added to the server that would detect when it was getting overloaded and would stop a request by returning an error to the client. This

error would trigger the client to return to the LSD again to determine another direct IOR to use. In addition, if a server being used by a cluster unaware client goes down or is disconnected, an error will be returned and the client will return to the LSD and start the process over.

Routing Algorithm

While the weighted round robin algorithm described in “EJB Workload Management Server Selection Policy” on page 100 is the main routing algorithm, there are situations that will override how the request is routed. These situations are described in the next sections.

In-process

If a client request is to an object that is already in the same process as the client, the request will automatically be routed to the in-process object. In this case, no weights are decremented. The in-process optimization overrides all other selection criteria.

Prefer local

“Prefer local” is an additional attribute that can be selected on a given server cluster. This attribute will instruct the selection logic to keep all client requests local to the same node as the client if possible. Thus, if the Prefer local setting is enabled, and a request from a client can be serviced by an application server that is local to the same machine as the client, the request will be routed to that local server. Prefer local still decrements the weight for each request, and if multiple application servers (cluster members) are local to the client, the standard weighted round robin algorithm will be used to route requests between all of the local servers. The “weight set” will be refreshed when all local weights reach zero.

Affinity

It is possible to establish an affinity between a client and an application server such that all client requests are sent to the same application server for the life of the affinity.

Examples:

- Transactional (strong) affinity: The first request after the transaction begins will determine to which application server the requests will be routed. All subsequent requests from this client will be routed to the selected application server until the end of the transaction.
- Applied (weak) affinity: The first time a cluster unaware client needs to use an object, the ORB (via an indirect IOR) will route the request to the LSD. The LSD will return the application server (via a direct IOR) to route the request to. All subsequent requests from this client for that object will be routed to the selected application server.

Thus, the standard weighted round robin algorithm can be overridden by various affinities. In affinity situations, the server weight value is still decremented for each request, as discussed in “EJB Workload Management Server Selection Policy” on page 100. However, requests with affinity continue to be sent to the associated application server even when the weight value reaches zero.

Location Service Daemon (LSD) Failover

If only one LSD is configured for each cluster, the potential for a single point of failure exists. To prevent the single point of failure, a set of LSDs per cluster will be configured and if an LSD fails, one of the other LSDs in the group will be used.

If the failure occurs after the routing table is available on the client, the WLMClient code does not go to the LSD to determine what server to route WLM'able objects. The WLMClient code handles the routing decisions. If the failure occurs before the first client request that retrieves the WLM information, then WLM depends on the LSD request to fail over to another LSD.

The WLM component is dependent on LSD failover because the first request for a client will require going to the LSD to obtain the first direct IOR. In order to support clients that do not have any WLM code running on them, for example z/OS clients, other ORB providers, and C++ CORBA clients, the WLM routing decisions will utilize the LSD.

Performance of clients that run in WebSphere Java ORBs with the WLM client code will not be affected at all. The IORs are manipulated and routed by the WLM client code, so indirect and direct IORs are handled the same.

Clients without the WLM code would only route to the LSD on the first request of each object. The direct IOR will be used by the client until the client either drops the reference to the object or the server becomes unavailable.

EJB WLM Failover Behavior and Tuning

Automatic failover cannot always happen due to constraints of transaction affinity, server stoppage, completion status, and server conditions. For situations where WLM cannot determine whether the request is completed, WLM will not automatically fail over the request. It is the client's responsibility to check if a retry is needed.

The EJB workload management plug-in uses the following failover strategies:

- If the workload management plug-in cannot contact a cluster member, it automatically redirects the request to another cluster member, providing automatic failover.
- If the application throws an exception, automatic failover does not occur. The workload management plug-in does not retry the request because it cannot know whether the request was completed.
- The workload management plug-in will catch and wrap communication failure exceptions in a `TRANSIENT` exception in those cases where transparent failover could not be achieved because it is unknown exactly how much processing the server did before the failure occurred. But should the application execute some compensation logic to get the state in a server back to where it should be (by performing a rollback or whatever needs to be done), then the request could be re-executed with potentially a different result (success). So, while the request could not transparently be failed over, there are still other cluster members available and if the request were tried again it might succeed. The application compensation and retry logic should stop retrying when a `NO_IMPLEMENT` exception is thrown instead of a `TRANSIENT` exception, because this indicates that no servers are available to handle the request.

The following is a list of exceptions that the EJB workload management, depending on the exception, may be able to transparently handle by redirecting the failed request to another cluster server:

- `org.omg.CORBA.COMM_FAILURE` or `org.omg.CORBA.NO_RESPONSE`

The return value of these exceptions will determine whether automatic failover occurs:

- With a `COMPLETION_STATUS` of `COMPLETED_NO`, automatic failover occurs because the request was not completed.
- With a `COMPLETION_STATUS` of `COMPLETED_YES`, failover does not occur because the request was successfully completed.
- With a `COMPLETION_STATUS` of `COMPLETED_MAYBE`, automatic failover does not occur. The workload management plug-in cannot verify whether the request was completed. In this situation, the client application must anticipate a failure and retry the request. The workload management plug-in then attempts to direct the request to a surviving cluster member.

- `org.omg.CORBA.COMM_FAILURE`

This exception is thrown by the ORB when a communications failure occurs. Any current transactions are rolled back and non-transactional requests are redone.

- `org.omg.CORBA.NO_RESPONSE`

This exception is thrown by the ORB when a communication failure occurs after a connection has been established, for example because a timeout occurs before getting a response.

- `org.omg.CORBA.TRANSIENT`

This exception is thrown by the ORB when a connection could not be established when the application tried to invoke a request.

The following is a list of exceptions that the EJB workload management does not handle and will propagate the exception to the client:

- If a `org.omg.CORBA.TRANSIENT` with minor code: 1229066304 (0x49421040) exception is thrown, the workload management plug-in found that the cluster member was in server quiesce mode.
- If a `org.omg.CORBA.TRANSIENT` with minor code: 1229066306 (0x49421042) exception is thrown, the workload management plug-in had a connection problem with a server.
- If a `com.ibm.CORBA.INTERNAL` with minor code: 1229066304 (0x49421040) exception is thrown, the workload management plug-in is no longer operating properly and no failover occurs.
- If a `com.ibm.CORBA.NO_IMPLEMENT` with minor code: 1229066304 (0x49421040) exception is thrown, the workload management plug-in has attempted repeatedly to contact all the servers without success. Workload management resumes when servers become available again.

If the failure occurs on the first initial request where the routing table information is not yet available, a `COMM_FAILURE` exception will be returned and the ORB will recognize that it has an indirect IOR available and re-send the request to the LSD to determine another server to route to. If the failure occurs after the client retrieved the routing table information, the WLM client will handle the `COMM_FAILURE`. The

server will be removed from the list of selectable servers and the routing algorithm will be used to select a different server to route the request to.

Consider the following sequence of a client making a request to an application server in the EJB container:

- 1** For the initial client request, no server cluster and routing information is available in the WLM client's runtime process. The request is therefore directed to the LSD that is hosted on a Node Agent to obtain routing information. If the LSD connection fails, the request will redirect to an alternative LSD. If this was not the first request, the WLM client would already have routing information for WLM-aware clients. However, for WLM-unaware clients, the LSD will always route requests to available servers. For future requests from the client, if there is a mismatch of the WLM client's routing information from what is on a server's, new routing information (as service context) will be added to the response.
- 2** After getting the InitialContext, a client does a lookup to the EJB's home object (an indirect IOR to the home object). If a failure occurs at this time, the WLM code will transparently redirect this request to another server in the cluster that is capable of obtaining the Bean's home object.

3 Server(s) become unusable during the life cycle of the request.

- If the request has strong affinity, there cannot be a failover of the request. The request will fail if the original server becomes unavailable. The client must perform recovery logic and resubmit the request.
- If the request is to an overloaded server, its unresponsiveness makes it seem as though the server is stopped, which may lead to a timeout. In these circumstances it may be helpful to change the server weight and/or tune the ORB and pool properties such as:

- com.ibm.CORBA.requestTimeout
- com.ibm.CORBA.RequestRetriesCount
- com.ibm.CORBA.requestRetriesDelay
- com.ibm.CORBA.locateRequestTimeout

These can be command-line properties or changed using the Administrative Console.

- If the com.ibm.ejs.wlm.MaxCommFailures threshold has been reached for a cluster member, it is marked unusable. By default, the MaxCommFailures threshold is 0, so that after the first failure the application server is marked unusable. This property can be modified by specifying
-Dcom.ibm.ejs.wlm.MaxCommFailures=<number>
as a command-line argument when launching a client.
- If a machine becomes unreachable (network and/or individual machine errors) before a connection to a server has been established, the operating system TCP/IP keep-alive timeout dominates the behavior of the system's response to a request. This is because a client will wait for the OS-specific keep-alive timeout before a failure is detected. This value can be modified, but as described in "ConnectTimeout Setting" on page 169, only with caution.

If a connection is already established to a server, com.ibm.CORBA.requestTimeout dominates (the default value is 180 seconds), and a client will wait this length of time before a failure is detected. The default

value should only be modified if an application is experiencing timeouts repeatedly, and great care must be taken to tune it properly. If the value is set too high, the failover will be very slow, and set too low, requests will time out before the server has a chance to respond.

The two most critical factors affecting the choice of a timeout value are the amount of time to process a request and the network latency between the client and server. The time to process a request in turn depends on the application and the load on the server. The network latency depends on the location of the client. For example, those running within the same LAN as a server may use a smaller timeout value to provide faster failover. If the client is a process inside of a WebSphere Application Server (the client is a servlet), this property can be modified by editing the request timeout field on the Object Request Broker property sheet. If the client is a Java client, the property can be specified as a runtime option on the Java command line. For example:

```
java -Dcom.ibm.CORBA.requestTimeout=<seconds> MyClient
```

- A failed server is marked unusable, and a JMX notification is sent. The routing table is updated. WLM-aware clients are updated during request/response flows. Future requests will not route requests to this cluster member until new cluster information is received (for example, after the server process is restarted), or until the expiration of the `com.ibm.ejs.wlm.unusable.interval`. This property is set in seconds. The default value is 300 seconds. This property can be set by specifying `-Dcom.ibm.ejs.wlm.unusable.interval=<seconds>` on the command-line arguments for the client process.
- When a request results in an `org.omg.CORBA.COMM_FAILURE` or `org.omg.CORBA.NO_RESPONSE`, the return value of `COMPLETION_STATUS` determines whether a request can be transparently redirected to another server. In the case of `COMPLETED_NO`, the request can be rerouted. If the completed status is `COMPLETED_YES`, no failover is required. In the case of `COMPLETED_MAYBE`, where the request was successful, but some communication error was encountered during the marshaling of the response, WLM cannot verify whether the request was completed successfully and cannot redirect the request. For example, consider a transaction that must be "at most once". In that case had WLM redirected the request, and it is possible the request would be serviced twice. The programming model is for the client to receive this exception and to have logic in place to decide whether or not to retry the request.
- If all servers are unavailable, the request will result in a `org.omg.CORBA.NO_IMPLEMENT`. At this point, either the network is down, or some other error has caused the entire cluster to be unreachable.

Please note that, similar to the situation of the Web container as discussed earlier, the application servers on a node will be forced to stop when the network is down if the loopback is not configured with the alias of a host IP.

EJB Caching and Failover

When a client accesses a workload managed EJB, it may be routed to one of a number of instances of the EJB on a number of servers. Not all EJB references can be utilized this way. Table 11.1 shows the types of EJBs that can be workload managed.

Table 11.1 EJB types and failover

EJB Type and Option	Type	WLM'able
Entity Bean, Caching option A	Home	Yes
	CMP	No
	BMP	No
Entity Bean, Caching options B & C	Home	Yes
	CMP	Yes
	BMP	Yes
Message-Driven Bean	MDB	Yes
Session Bean	Home	Yes
	Stateless	Yes
	Stateful	No

The only type of EJBs that cannot be workload managed are instances of a given stateful session bean and entity beans with commit time caching option A (WebSphere supports Option A, Option B, and Option C caching, see “EJB types” on page 82 for a detailed description of the three caching options).

A stateful session bean is used to capture state information that must be shared across multiple client requests that are part of a logical sequence of operations. To ensure consistency in the logic flow of the application, a client must always return to the same instance of the stateful session bean, rather than having multiple requests workload managed across a group of available stateful session beans. Because of this restriction, stateful session bean instances are not considered failover tolerant. If the process running the stateful session bean fails, the state information maintained in that bean is unrecoverable. It is the responsibility of the application to reconstruct a stateful EJB with the required state. If this is not the case, an alternative implementation, such as storing state in an Entity EJB, should be considered.

An entity bean represents persistent data. It is common for a client to make a succession of requests targeted at the same entity bean instance. It is also possible for more than one client to independently access the same entity bean instance concurrently. The state of an entity bean must be kept consistent across multiple client requests.

Within a transaction, the WLM ensures that the client is routed to the same server. Between transactions, the state of the entity bean can be cached. Entity beans can be workload managed if they are loaded from the database at the start of each transaction. By providing either Option B caching or Option C caching (the default), entity beans can participate in WLM. These two caching options ensure that the entity bean is always reloaded from the database at the start of each transaction.

Please note that WebSphere also supports the optimistic concurrent control, where the cached data is checked and a collision is detected during the commit stage. Loading data from the database may not be required at the transaction start if the application design is in place to stamp cached entity beans.

Resource Redundancy

EJB failover relies on data and resource availability. If the data and resources are unavailable, the EJB client request will fail. For example, if the database fails, the entity bean cannot complete the transaction, and the EJB client will fail. See Chapter 12, “Data Considerations for High Availability” on page 209 for further information.

WebSphere and IP-Based Database Failover

Web applications running in the Web container rely on data and resource availability. If the data and resources are unavailable, the Web application will fail. For example, if a Web application relies on data in a database to complete the request, and the database fails, the Web application cannot complete the request.

This section examines WebSphere behaviors during and after database failovers for IP-based highly available data management systems.

WebSphere Connection Manager

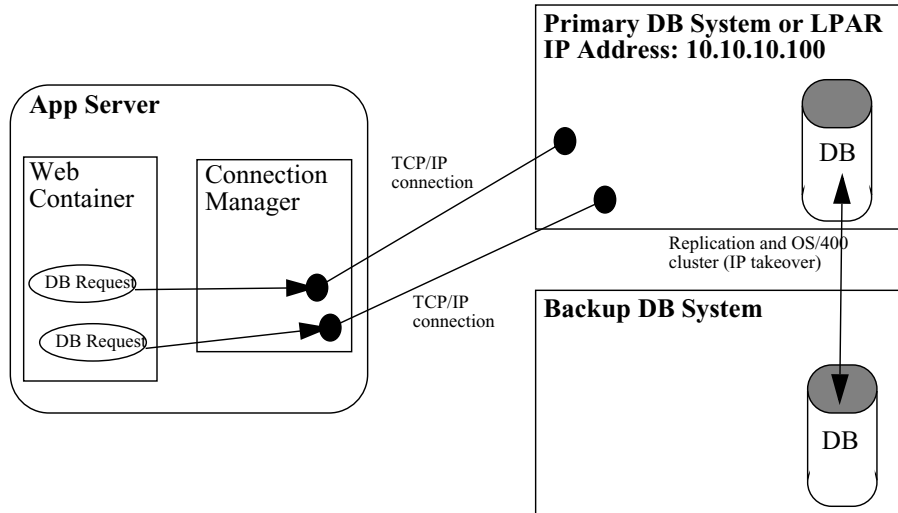
The WebSphere Connection Manager is the link between applications deployed in an application server and the database in support of these applications. When writing applications it is important to understand what happens when access to database is lost to such things as the database server node failing or the network experiencing problems. In all cases, the connections established between the application and the backend data server cannot be used and the pool of these connections needs to be flushed.

There are two vary vital functions performed by the WebSphere connection manager:

- Each time a client (servlet or EJB) attempts to access a database, it must acquire a connection to the database (via JDBC provider). The connection made by the connection manager is a TCP/IP sockets connection. A new connection can be costly, so the connection manager allows the administrator to create a data source that has a pool of database connections that can be shared by applications. Prepared SQL statement cache coupled with the Connection Manager provides performance benefits for applications.
- The Connection Manager provides architected failover semantics when the DB fails. With these semantics applications can be coded to properly handle this situation. The following diagram illustrates normal runtime behavior.

To illustrate the above points, we will take you through a scenario where a high available database (see Chapter 12, “Data Considerations for High Availability” on page 209 for further information on high available database) is unavailable. In Figure 11.6, clients (e.g. servlets) obtain a connection from the WebSphere Connection Manager, perform the operation and then return the connection.

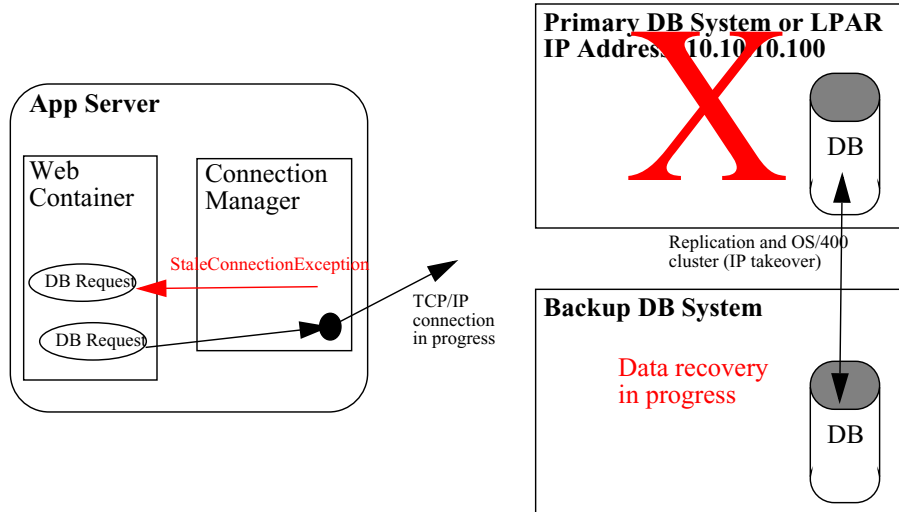
Figure 11.6 WebSphere Connection Manager interaction with DB - all is well



One of two JDBC providers can be used on OS/400. The providers are Java Toolbox for iSeries or the Native driver. We recommend you use Java Toolbox for iSeries for various reasons. In our example we elected to use HABP data replication to insure a HA DB solution. The Java Toolbox driver is configured to send all database requests to 10.10.10.100 and the proper host server port. There is a one to one correlation between a connection and host server job which lasts as long as the connection is established. High availability database is assured using OS/400 clustering and a HABP product to replicate data between the two systems. The clustering framework manages the IP address and works hand in hand with HABP to perform planned switchovers and failover between the primary and backup database systems. The failover mechanism to redirect database requests is IP takeover.

Figure 11.7 on page 190 illustrates what happens when the backend database goes down whether by planned switchover or unplanned failover.

Figure 11.7 WebSphere Connection Manager interaction with DB – Database fails



When the backend database fails, at least one client will encounter a `StaleConnectionException`. When this happens the entire pool of connections is destroyed and a new pool will be created to handle subsequent client requests. Applications must be coded to properly handle a `StaleConnectionException` (note that this exception extends `SQLException`, which must be handled in an application that performs database operations). One must understand that a host server job is assigned to each JDBC connection for the life of that connection. At any instance in time the host server job may be at a non-committed transactional state. If either side of the connection goes down, the host server jobs will rollback all open transactions. The application of course will incur a `StaleConnectionException` and in doing so should consider any open transaction rolled back. Applications considerations will be addressed later in this section. Let's understand how a JDBC connection could be lost. This is important in order to understand the TCP/IP configuration settings for the cluster member system, host server system and the Java Toolbox customer properties configuration.

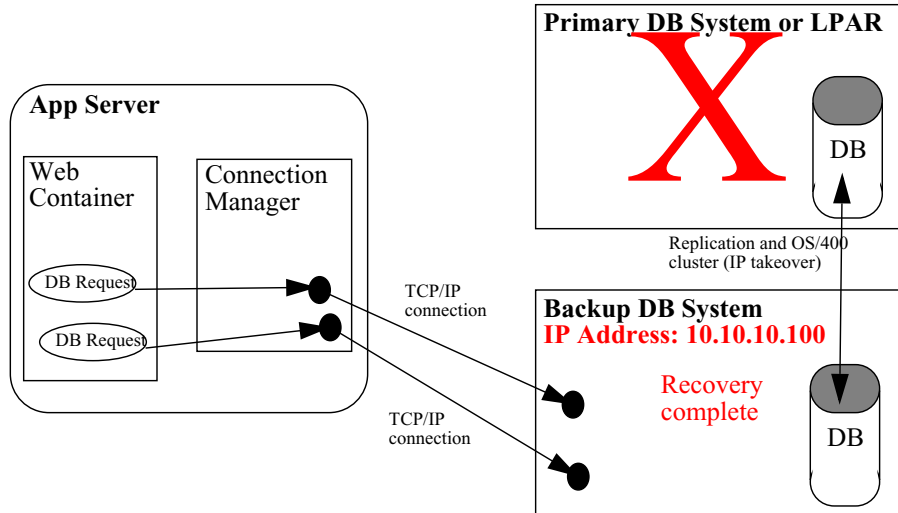
The application server goes down and as part of process termination the connection is closed. The host server side of the connection is still active but not for long. The host servers use the keep-alive socket option when establishing the connection. This will cause a TCP probe to be sent out every so often to determine if the other end of the connection is gone. How often the probe is sent is dependent on the TCP/IP configuration for the host server system. TCP/IP configuration recommendations will be addressed later in this section. The host server upon encountering the bad connection will roll back any non-committed transactions.

The host server job goes down gracefully and as part of process termination the connection is closed and any open commit cycles are rolled back. The WebSphere application server need to assure that any thread waiting (socket receive) for response from the host server does not wait forever. Again TCP/IP keep-alive comes into play. The Java Toolbox configuration has provisions enabling keep-alive on a connection basis. This will assure that if the host server side of the connection goes away, the application server thread will be notified in a reasonable time and not potentially hang forever waiting for a response.

The host server happens to crash. Here the OS/400 Cluster services and HABP recovery kick in. The HABP software performs database recovery. Non-committed transactions will not be applied to the database. Again TCP/IP keep-alive on the application server side of the connection will assure threads are not hung forever.

In our example here we assume the third case where the host server crashes and database recovery occurs. The thread attempting the DB connection succeeds and most clients, if patient, may notice no failure. Keep in mind this outage may be several minutes; therefore, some clients may select 'stop' followed by 'reload' in their browsers. Figure 11.8 illustrates what happens once the database comes back online:

Figure 11.8 WebSphere Connection Manager interaction with DB – Backup is ready



At some point the backup DB node is ready and the IP address becomes active (IP takeover). At that time an ARP is sent out causing all ARP caches within the network to be updated with the backup DB MAC address. The threads attempting the DB connections succeed and most clients, if patient, may notice no failure. Keep in mind this outage may be several minutes; therefore, some clients may select 'stop' followed by 'reload' in their browsers.

TCP/IP and Java Toolbox Configuration

As mentioned above upon encountering a `StaleConnectionException` the WebSphere connection manager will clear the pool and create a new pool attempting to establish new connections. The connection manager will attempt to connect to the IP address 10.10.10.100. Of course the database failover recovery may not be complete and OS/400 clustering has not yet activated the IP address 10.10.10.100. An OS/400 TCP/IP connect operation will wait for two minutes for the remote side to respond. At that time a timeout will be signaled to WebSphere connection manager. Typically WebSphere Connection Manager will retry a connect which fails with a timeout (2 minutes). It will retry up to three times or approximately 6 minutes when the database tier is down. This activity can consume many WebSphere application server threads, thereby preventing all Web requests from being serviced in a timely fashion. An ARP cache timeout value

of 2 minute (on the application server machines) will assure that the connect will fail in such a way that WebSphere will not retry the connect. This will prevent the consumption of threads which could be processing other Web requests not dependent upon the failed database. ARP cache is scoped to unique IP address/mac address pairs which should minimize the performance overhead of the ARP. Also the TCP/IP configuration items (TCP R1 and TCP R2) retransmission counts should be set to 3 and 9 respectively. This will assure if a sending thread has incurred flow control and entered MI wait (sleeping), that it will wake up if the remote side of the connection goes down. And the receiving side has gone down. The retransmission will wake the sending thread up to prevent a hang.

So let us summarize the configuration values to set in order to assure timely failover.

For Java Toolbox for iSeries (change following custom properties for all version 5.0 and 4.0 datasources):

- keepAlive to “true”
- receiveBufferSize to 1 meg
- sendBufferSize to 1 meg

For WebSphere cluster member machine(s) and database host server machine(s), use CHGTCPA and set the following values:

- TCP keep alive -- 2 minutes
- TCP R1 retransmission count – 3
- TCP R2 retransmission count – 9
- ARP cache timeout – 2

Application Considerations for StaleConnectionException

When a StaleConnectionException is encountered, the application should consider any non-committed transactions rolled back. Explicitly catching a StaleConnectionException is not required by most applications. Because applications are already required to catch java.sql.SQLException, and StaleConnectionException extends SQLException, StaleConnectionException is automatically caught in the general catch-block. However, explicitly catching StaleConnectionException makes it possible for an application to perform additional recovery steps from bad connections. Before continuing, it is important to understand that WebSphere will automatically reconnect to the database when future requests arrive without any application intervention. The use of the StaleConnectionException exception should be limited to applications that feel they need an extra level of transparency in addition to the automatic recovery provided by WebSphere. This transparent recovery code can be very complex to develop in more complex scenarios involving multiple resources or complex logic flows. The following discussion is aimed at teams that feel that this additional complexity is warranted.

The most common time for a StaleConnectionException to be thrown is the first time that a connection is used, just after it is retrieved. Because connections are pooled, a database failure is not detected until the operation immediately following its retrieval from the pool, which is the first time communication to the database is attempted. And it is only when a failure is detected that the connection is marked stale. StaleConnectionException occurs less often if each method that accesses the database

gets a new connection from the pool. Examining the sequence of events that occur when a database fails to service a JDBC request shows that the failure occurs less often because all connections currently handed out to an application are marked stale. The more connections the application has, the more `StaleConnectionException`'s which occur.

Generally, when a `StaleConnectionException` is caught, the transaction in which the connection was involved needs to be rolled back and a new transaction begun with a new connection. Details on how to do this can be broken down into two categories:

- A connection in auto-commit mode
- A connection not in auto-commit

Connections in auto-commit mode

By default, any connection obtained from a one-phase datasource (implementing `javax.sql.ConnectionPoolDataSource`) is in auto-commit mode when there is no scoping transaction. When in auto-commit mode, each database action is executed and committed in a single database transaction. Servlets often use connections in auto-commit mode, because transaction semantics are not necessary. Enterprise applications do not usually use connections in auto-commit mode. Auto-commit can be explicitly disabled by calling `setAutoCommit()` on a `Connection` object. When a `StaleConnectionException` is caught from a connection in auto-commit mode, recovery is a simple matter of closing all of the associated JDBC resources and retrying the operation with a new connection. Note: In some cases the cause of the database outage might be transient. In these cases, it might be worthwhile to add a pause to the retry logic to allow for database service restoration. The number of retries as well as any pause should be kept small so as to not keep a web site user waiting indefinitely.

An example of this follows:

```
public void myConnPool() throws java.rmi.RemoteException
{
    boolean retry = false; // retry or not
    int numOfRetries = 0; // total retries attempted
    java.sql.Connection conn = null;
    java.sql.Statement stmt = null;

    try {
        do {
            try {
                //Assumes that a datasource has already been obtained from JNDI
                conn = ds.getConnection();
                stmt = conn.createStatement();
                stmt.execute("INSERT INTO ORG VALUES (10, 'Pacific', '270', 'Western', 'Seattle')");
                retry = false;
            }
            catch (com.ibm.websphere.ce.cm.StaleConnectionException sce) {
                // if a StaleConnectionException is caught rollback and retry the action
                if (numOfRetries < 2) {
                    retry = true;
                    numOfRetries++;
                    sleep(10000) ; // add an optional pause
                }
                else
            }
        }
    }
```

```

        retry = false;
    }
    } while (retry);
}
catch (java.sql.SQLException sqle) { //deal with other DB exception }
finally {
    //always cleanup JDBC resources
    try {
        if (stmt != null) stmt.close();
    }
    catch (java.sql.SQLException sqle) { // usually can ignore }

    try {
        if (conn != null) conn.close();
    }
    catch (java.sql.SQLException sqle) { }
}
}
}

```

Connections not in auto-commit mode

If a connection does not have auto-commit enabled, multiple database statements can be executed in the same transaction. Because each transaction uses a significant number of resources, fewer transactions result in better performance. Therefore, if a connection is used for executing more than one statement, turn off auto-commit mode and use transactions to group a number of statements into one unit of work.

If a transaction is begun in the same method as the database access, recovery is straightforward and similar to the case of using a connection in auto-commit mode. When a `StaleConnectionException` is caught, the transaction is rolled back and the method retried. If a `StaleConnectionException` occurs somewhere during execution of the try block, the transaction is rolled back, the retry flag is set to true, and the transaction is retried. As is the case with connections in auto-commit mode the number of retries should be limited as well as any pause, because the exception might not be transient. This is illustrated in the following:

```

do {
    try {
        //begin a transaction
        tran.begin();

        //Assumes that a datasource has already been obtained from JNDI
        conn = ds.getConnection();
        conn.setAutoCommit(false);
        stmt = conn.createStatement();
        stmt.execute("INSERT INTO ORG VALUES (10, 'Pacific', '270', 'Western', 'Seattle')");
        tran.commit();
        retry = false;
    }
    catch (com.ibm.websphere.ce.cm.StaleConnectionException sce) {
        // if a StaleConnectionException is caught rollback
        // and retry the action
        try {
            tran.rollback();
        }
    }
}

```

```

        catch (java.lang.Exception e) {
            // deal with exception. In most cases, this can be ignored
        }
        // deal with other database exceptions and clean up as before
    }
}

```

When a transaction is begun in a different method from the database access, an exception needs to be thrown from the data access method to the transaction access method so that it can retry the operation. In an ideal situation, a method can throw an application-defined exception, indicating that the failure can be retried. However this is not always allowed, and often a method is defined only to throw a particular exception. This is the case with the `ejbLoad` and `ejbStore` methods on an enterprise bean.

A more comprehensive discussion of each of these scenarios as well as code samples is available in the whitepaper; *WebSphere Connection Pooling*, see resources below.

Impact of Database Failures

The following discusses the impacts of a database failure on various components of WebSphere.

Administrative Servers and Administrative Actions

IBM WebSphere Application Server Network Deployment V5.0 does not use a central repository to store configuration information.

When the database is down, WebSphere V5 administrative server processes are unaffected and still run as usual; all administrative actions still function. We discussed WebSphere V5.0 master repository and Deployment Manager high availability in “Deployment Manager and Node Agent High Availability” on page 197.

Persistent Session

You have the options to put the session data in memory or to store it in a database. Thus, you can use the WebSphere V5 Data Replication Service (memory-to-memory replication) or database as mechanisms for session failover.

If you choose to store your session data into a database, a highly available session database is achieved through clustering software or parallel database.

During a failover, the session data will be temporarily unavailable and any request that requires a persistent session will fail. After a failover, the session data will become available again and any request will succeed after the first retry.

See “HTTP Session Failover” on page 172 for more information on persistent sessions.

Java/C++ Applications and Application Reconnecting

During the failover, any in-flight application that uses database access will fail. After the failover, any application will succeed on retry. For the Java applications, you can catch the `StaleConnectionException` in your database try block and retry your connection a limited number of times, with the sample code discussed in “Application Considerations for `StaleConnectionException`” on page 192.

We have three comments regarding this coding practice:

- This coding practice is not required for the HA solution to work; it is only an option.
- If a database failover takes approximately N minutes and you code a try block in your application with an unlimited number of retries, then your application will hang there for one to N minutes during a failover. There is not much to gain by trying hundreds of times during a failover; it may impact the performance of other WebSphere and application processes that do not need to access the database. We suggest that you either do nothing in your application code, or code a limited number of retries (no more than five times with a controlled retry interval). We suggest that you consider the trade-off for each option before you develop your application.
- You need to pay attention to transaction ACID properties (atomicity, consistency, isolation, and durability) when coding retry.

Enterprise beans

Entity EJBs store data in databases, and session beans may be coded directly with JDBC to access the database or work with Entity EJBs. EJBs that use database access will fail during a failover and succeed after a failover with no more than one retry. For CMP EJB, when a part of data access is not controlled by WebSphere containers and BMP EJB data access, you have the option to code a retry block in your code as described above. If you code the retry block in your EJBs, clients will hang during a failover. For CMP EJBs, restarting application servers will clean the connection pool after a failover.

Web-based Clients, Applets, Servlets, and JSPs

If these clients do not need database access, they run as usual, without interruption. If database access is needed, requests will fail until the failover has finished. In addition to direct database access, these clients may also use persistent HTTP sessions. Once the database failover has finished, with no more than one retry, these clients should run as usual.

Web-based clients are easy to retry on the client side (just go back in the browser and submit the request again). Therefore, coding a retry block is not as important as in applications. But you can choose to code the retry block in your servlets and applets.

Naming Service and Security Service

WebSphere V5 does not use a database to store its name bindings. If a database is used for the security repository, users cannot log in during a failover, but will succeed after a failover.

In order to minimize lookup failures, you can code your applications as follows:

```
if (obj==null)
    ctx.lookup(obj);
```

Since the system has run for a while before a failure, the object you are looking up may already exist, so you do not need to look it up.

Workload Management

No matter which cluster member is used in the WebSphere WLM environment, during a database failover all requests to access data will fail. After a failover, any request should succeed after no more than one retry. The workload-managed WebSphere Application Servers do prevent some node or process failures, but not database network, database node and database process failures. The WebSphere WLM mechanism protects the system from process failures. We still need HA data management for WebSphere data

availability. See Chapter 12, “Data Considerations for High Availability” on page 209 for more information.

Deployment Manager and Node Agent High Availability

In IBM WebSphere Application Server Network Deployment V5.0, the distributed administrative work is accomplished by the Node Agent server that resides on each node and the Deployment Manager that acts as a central point for administrative tasks.

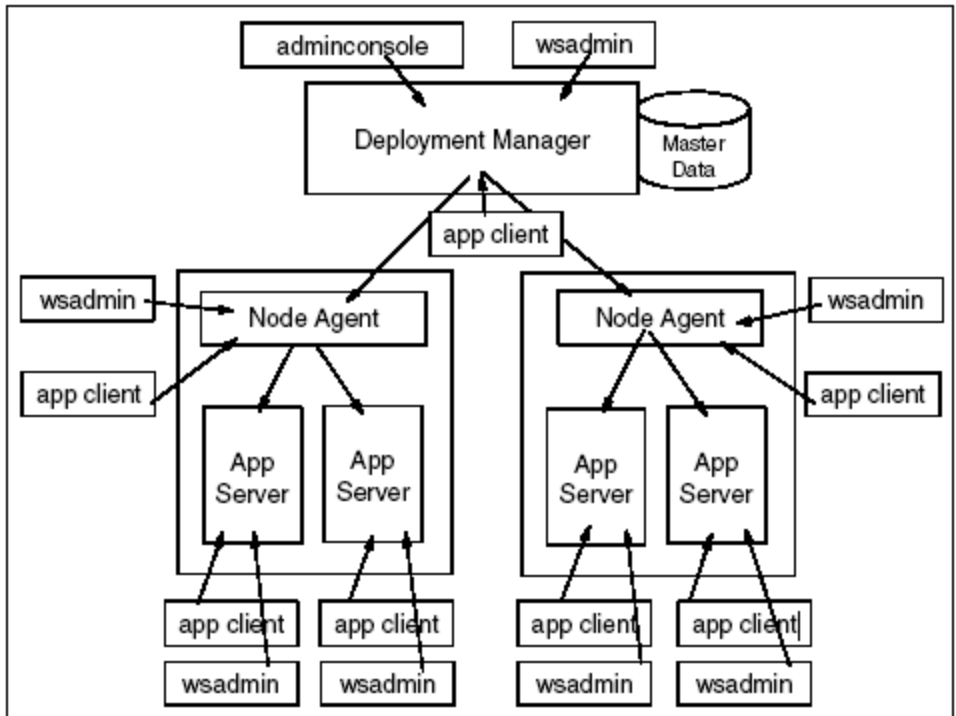
The Node Agent server and the Deployment Manager server both use a repository of XML files on their own nodes. The master repository data is stored on the Deployment Manager node. That data is then replicated to each node in the Administrative domain (or cell). The Deployment Manager server and the Node Agent servers keep these files synchronized. The default synchronization interval is one minute. The synchronization process is unidirectional from the Deployment Manager to the Node Agents to ensure repository integrity. That means that any changes made on the Node Agent level are only temporary and will be overwritten by the Deployment Manager during the next synchronization. Only changes to the master data through the Deployment Manager are permanent and replicated to each node.

Administrative and configuration actions as well as operational changes can be applied to either the Node Agent server, the application server, or the Deployment Manager through wsadmin scripting. To communicate with these servers you can use either the SOAP or the RMI protocol. The Administrative Console (adminconsole.ear) is an application that is installed in the Deployment Manager server by default in a WebSphere Network Deployment environment.

The Node Agent server provides runtime support for the Location Service Daemon (LSD), file transferring and synchronization, JMX server, distributed logging, naming server, security server, and EJB workload management configuration.

The Deployment Manager provides runtime support for WLM services, file transferring and synchronization, configuration management, JMX server, distributed logging, naming server, security server, and for the master configuration.

Figure 11.9 WebSphere Deployment Manager and Node Agent for configuration administration and application bootstrapping



There are six types of errors in the process, network, disk, and machine that may cause a Node Agent or the Deployment Manager servers to fail:

- Expected server process failure, for example stopping the server.
- Unexpected server process failure.
- Server network problem, for example the network cable is disconnected or a router is broken.
- Unexpected and expected machine problems, for example a machine shutdown, operating system crashes, or power failures.
- Disk is faulty and the configuration file cannot be read.
- Overloading with heavy hacking or denial of service (DOS) attacks.

We discuss these issues in the following sections.

Node Agent High Availability Considerations

We discuss the behaviors of Node Agent server failures and how WebSphere V5 mitigates the impacts of these failures on WebSphere Application Servers and application clients. Also, we describe how to set up Node Agents to tolerate these failures or to minimize the impact of these failures.

Impact of Node Agent Failures

The Node Agent provides several important services to the Deployment Manger, application servers, and to the application clients. These services include administrative services, the Location Service Daemon, file transfer service, synchronization service, naming service, security service, and RAS service.

A node agent does not have to be active in order for the application server to run. However, it does have to be active if you want to manage the application server from the administrative console. In addition, if the node agent is not active then a failure in the application server will not be detected, which means no attempts to activate the application server will be performed.

Application servers

The Node Agent hosts the Location Service Daemon, publishes and synchronizes configuration data to other application servers, and monitors and launches the managed application server process.

An application server can only be started if the local Node Agent is available.

If the Node Agent is not active, the application server cannot register itself with the LSD and an error message will be logged.


Once the application server is started, it will run normally even if the Node Agent becomes unavailable. However, configuration data cannot be updated consistently.

The application server processes are managed by the Node Agent server. When the Node Agent is unavailable, the application server loses the benefit of being a managed server. Therefore, an application cannot be restarted automatically if the application server dies unintentionally. You can tune the managed-by-nodeagent server process monitoring parameters on the Administrative Console as shown in Figure 11.10 on page 200.






Figure 11.10 Managed server monitoring parameters tuning

[Application Servers](#) > [BASE839](#) > [Process Definition](#) >

MonitoringPolicy

Policy settings for performance monitoring of the application server.. 

Configuration

General Properties		
Maximum startup attempts	* <input type="text" value="3"/> attempts	 Specifies the number of attempts to attempt to start before giving up.
Ping interval	<input type="text" value="60"/> seconds	 Specifies the frequency of communication attempts between the parent process, smc, and the process it monitors. An application server based on your configuration will fail if it fails to respond to ping requests. Decreasing the ping interval value reduces the system load.
Ping timeout	* <input type="text" value="300"/> seconds	 The interval after which the monitored process is considered failed if it does not respond to ping requests.
Automatic restart	<input checked="" type="checkbox"/>	 Specifies whether the system should automatically restart the process if it fails.
Node restart state	* <input type="text" value="STOPPED"/> ▼	 Specifies the state of the process when autoRestart is enabled. The default is STOPPED.

It is recommended that each application server on a node be restarted if the Node Agent is restarted.

Deployment Manager

Node Agents and the Deployment Manager work together to manage administrative tasks and to make configuration and operational changes. If a Node Agent fails, the Deployment Manager may remove servers on that node from its WLM routing table and will publish this information to other nodes where the Node Agent is still active. In addition, administrative tasks from the Deployment Manager to that node will fail if the Node Agent is not available, any configuration and operational changes will not be reflected in this node. Once the failed Node Agent recovers, the routing table will be updated and published to all other nodes again.

Location Service Daemon

The LSD was redesigned in WebSphere V5.0 to mitigate the impact of Node Agent failures. We rely on multiple Node Agents and built-in HA LSD to provide high availability. The Node Agent itself is not workload managed, but WebSphere V5.0 provides the mechanism to make indirect IORs (Interoperable Object Reference) aware of all LSDs.

Naming servers

In IBM WebSphere Application Server Network Deployment V5.0, a naming server exists in every server process (application server, Node Agent, and Deployment Manager). You can bootstrap your client to any of these naming servers. Usually servlets, EJB clients, and J2EE clients start their bootstrap from their local application server and end up on the server where the objects are found.

For Java clients, you can bootstrap to more than one naming server on different Node Agents as shown below:

```
prop.put(Context.INITIAL_CONTEXT_FACTORY,
"com.ibm.websphere.naming.WsnInitialContextFactory");
prop.put(Context.PROVIDER_URL, "corbaloc::host1:2809,host2:2809");
Context initialContext = new InitialContext(prop);
try { java.lang.Object myHome =
initialContext.lookup("cell/clusters/MyCluster/MyEJB");
myHome = (myEJBHome) javax.rmi.PortableRemoteObject.narrow(myHome,
myEJBHome.class);
} catch (NamingException e) { }
```

The client automatically tries the bootstrap servers on host1 and host2 until it gets an InitialContext object. The order in which the bootstrap hosts are tried is not guaranteed.

The Node Agent is not workload managed. If a client gets a cached InitialContext object and that Node Agent is unavailable, the InitialContext object will not automatically fail over to another Node Agent, so the lookup with this InitialContext object will fail. In order to avoid this problem, you need to disable the cache in your client by supplying the following property when initializing the naming context:

```
prop.put(PROPS.JNDI_CACHE_OBJECT, PROPS.JNDI_CACHE_OBJECT_NONE)
```

Where PROPS.JNDI_CACHE_OBJECT is a Java constant defined in com.ibm.websphere.naming.PROPS.

Or more conveniently by setting the Java command line property as:

```
java -Dcom.ibm.websphere.naming.jndicacheobject=none MyClient
```

Using Node Agents as bootstrap servers is not recommended because they are not workload managed. Because application servers in a cluster are workload managed, you should use application servers as bootstrap servers instead as follows:

```
prop.put(Context.PROVIDER_URL,
"corbaloc::host1:9810,host1:9811, :host2:9810, :host2:9811");
```

The above discussion is only relevant for naming lookup (read) operations. If your application is binding (writing), a failure will occur if the Node Agent is unavailable, since the Node Agent coordinates the binding process.

Security server

Security servers are also present in every application server process in WebSphere V5. Security credentials can fail over from one server to another in a cluster until they are expired.

Application clients

An application client receives its routing information from the LSD (routing table) hosted in the Node Agent. If the application server environment does not have routing information for a client failover available, the client will fail to run if all Node Agents are unavailable. Once the application server environment has routing information available, the application client will run successfully if it is not using a single Node Agent to bootstrap and is not binding (writing). All EJB IORs contain the list of LSD host and ports. However, the Deployment Manager may remove servers on a node with a failed Node Agent and publish a new routing table to all nodes where Node Agents are still alive, which disturbs the WLM routing.

Synchronization service and file transfer service

All the following services will fail to function when the Node Agent is unavailable.

RAS service, PMI and monitoring

A node with a failed Node Agent will not be able to provide RAS service, PMI, and monitoring information to the cell.

Administrative clients

There are two kinds of administrative clients: the Administrative Console and the wsadmin scripting interface. wsadmin can be attached to any server (Node Agent, Deployment Manager, or application servers). The Administrative Console is a Web application that is usually installed in the Deployment Manager server in a WebSphere Network Deployment environment.

Administrative console

The following limitations when working with the Administrative Console apply when a Node Agent is not available:

- Any query for any information on that node will fail and leave the server status for that node as “unknown” or “unavailable”.
- Any configuration changes will not appear on that node until the Node Agent is restarted.
- Any operational actions (start, stop, delete, create) from the Administrative Console to the servers on the node with a failed Node Agent will fail.

However, you can still do any administrative and configuration tasks using the Administrative Console for other nodes where the Node Agents are still alive.

Scripting interface - wsadmin

You can connect the wsadmin scripting interface to every server in the WebSphere administrative domain (or cell). By default, wsadmin is attached to the Deployment Manager. You can change the default by editing the wsadmin properties file located in the <install_root>/properties directory or you can specify the conntype, host, and port parameters in the command line:

```
wsadmin -conntype RMI -host mynode -port 2809
```

Obviously, wsadmin cannot connect to the Node Agent when it is unavailable.

Enhancing Node Agent High Availability

In WebSphere Network Deployment, we usually have more than one Node Agent in the cell, and HA LSD and IOR contain the information of all nodes, so another Node Agent can service client requests if one Node Agent fails. Therefore, having more than one Node Agent in a cell is the basic solution to ensure that LSD failover will occur if a Node Agent process is lost.

The Node Agent also monitors its managed application servers and maintains configuration data currency and integrity in runtime. If the Node Agent is unavailable, the application servers will lose the protection and the configuration data may become stale.

Loopback alias configuration

Expecting an application server to run when the machine is disconnected from the network requires that the loopback port address (127.0.0.1) is configured as an alias endpoint for the machine. The reason for this is that the Node Agent normally pings all application servers on the node. When the Node Agent detects that an application server is not available, it stops and tries to restart the application server. This goes on for 10 minutes, at which time the Node Agent no longer tries to restart the application server. The ping from the Node Agent to the application server uses the Internet Protocol (IP). If the machine is disconnected from the network, the communication occurs on the loopback port. However, the loopback port is only considered a valid port for the application server when the loopback port address (127.0.0.1) is configured as an alias for the machine.

The behavior when the loopback port is not configured as an alias is that the application server is stopped (because the Node Agent cannot reach the application server), and the Node Agent will not be able to restart it until the machine is again connected to the network. If the time required to reconnect to the network is less than 10 minutes, then the application servers will be started on the next retry of the Node Agent. However, when the reconnection time exceeds the 10-minute interval, then an explicit start of the application servers has to be performed.

Monitor Node Agent process

Specialized software would need to be used to monitor the Node Agent process. When the Node Agent process crashes and stops unintentionally, the software monitoring the process will restart the Node Agent server automatically. However, if the disk where the configuration files reside is defective, or the operating system itself crashes, etc., the Node Agent will not recover until the manual recovery process has finished.

Deployment Manager High Availability Considerations

The Deployment Manager is not clustered and therefore is a single point of failure in WebSphere V5.0. However, its impact on the application client processing is limited because all configuration data is replicated to every Node Agent in the cell (although the configuration data may be stale or not current).

As soon as the Deployment Manager server becomes available again, all Node Agents in the domain will discover it, and all functions will return to normal.

In the following sections, we discuss the behaviors of Deployment Manager server failures, how WebSphere V5.0 mitigates the impacts of these failures on application

servers and application clients, and how to set up the Deployment Manager to tolerate these failures or to minimize the impact of these failures.

Impact of Deployment Manager Failures

The Deployment Manager is the central control point for all administrative and configurational tasks as well as operational actions. It provides a single image of the whole cell and a mechanism to administer the entire cell.

The Deployment Manager provides several important services to application servers, Node Agents, and application clients. It supports WLM services, administrative services, file transfer service, synchronization service, naming service, security service, PMI, and RAS service.

Configuration management

In WebSphere V5.0, each individual node is “added” to the Deployment Manager cell. All nodes in the domain are federated to create a master repository of XML files on the Deployment Manager node. The master repository is then replicated to all other nodes in the domain. The Deployment Manager keeps all copies of the configuration files synchronized across all nodes in the domain. During normal operation, the Deployment Manager consistently synchronizes the repository data on each node with the master repository. In addition to the configuration repository, you have the option to also synchronize application binaries.

You can turn synchronization on/off and change configuration parameters to tune synchronization using the Administrative Console.


WebSphere V5 also provides the option to manually synchronize copies of the repository and binaries by using the command:

```
<install_root>\WebSphere\AppServer\bin\syncNode.bat
```


or by using the Administrative Console as shown in Figure 11.11 on page 205.


Figure 11.11 Node synchronization







Nodes

A list of nodes in this cell. You can add new nodes into the cell by clicking on "Add Node" and specifying a remote running WebSphere Application Server instance. 

Total: 3

 Filter

 Preferences

<input type="checkbox"/>	Name 	Status  
<input type="checkbox"/>	LP02UT4_BASE839	
<input type="checkbox"/>	LPAR247M_CLONE839	
<input type="checkbox"/>	ND839Manager	

NOTE If the Deployment Manager is not available, the repository cannot be synchronized. You can use `wsadmin` to change the configuration at the node level. Repository files across all nodes in the domain may become inconsistent or stale. Once the Deployment Manager comes online again, all changes made to Node Agents or application servers are lost, and are overwritten with the master repository.

Node Agent

The Node Agent interacts directly with the Deployment Manager to perform all administrative and configuration tasks. If the Deployment Manager is unavailable, all cell-wide tasks cannot be executed and all tasks done locally will be overwritten by the master repository in the Deployment Manager node.

When the Deployment Manager is not available, configuration synchronization will fail.

The Deployment Manager and the Node Agents can be started in any order. They will discover each other as soon as they are started.

Application server

Application servers have no direct interaction with the Deployment Manager. However, they rely on up-to-date configuration data and on the fact that the Deployment Manager server replicates and initiates any changes. For example, the Deployment Manager controls the WLM master routing table. Application servers will not be informed of failed servers on other nodes if the Deployment Manager fails, and client requests may be routed to failed servers.

Naming server

The Deployment Manager also hosts a cell naming server that controls the whole naming context structure. Clients can bootstrap to the cell naming server using:

```
prop.put(Context.PROVIDER_URL, "corbaloc::dmgrhost:9809");
```

The Deployment Manager server is not workload managed in WebSphere V5.0, and neither is the naming service inside the Deployment Manager process. Thus, you should not use it to bootstrap your application clients.

No matter where your client bootstraps, any naming binding or update writing at cell scope will fail if the Deployment Manager is not available.

Security server

The cell security server is not available when the Deployment Manager is down. This has no impact on application clients since they use the local security service in every application server.

WLM runtime service

When the Deployment Manager is not available, the cell WLM runtime service that maintains and publishes the master routing table is also not available. Any cluster server operational status changes (such as start/stop servers) cannot be reflected on the master routing table, and cluster servers on different nodes will not get configuration updates.

Application clients

There is very limited direct impact on application clients when the Deployment Manager is not available. They will notice the failure only if they are bootstrapped to the cell naming server or when the application clients do bindings and updates. However, application clients may be impacted indirectly by unavailable configuration and operational services due to the failed Deployment Manager, for example inconsistent configuration data, stale WLM routing table, or aged application binaries.

Synchronization service and file transfer service

The Deployment Manager provides file transfer and synchronization of configuration files and application binaries across nodes in the domain. These services are not available when the Deployment Manager is down.

RAS service, PMI and monitoring

The cell-wide information and remote logs are not available when the Deployment Manager is unavailable.

Administrative clients

This section discusses the impacts of Deployment Manager failures on the Administrative Console and the wsadmin scripting interface.

Administrative Console

The default adminconsole application cannot be started if the Deployment Manager is unavailable.

Scripting interface - wsadmin

By default, you are connected to the Deployment Manager server. If it is unavailable, wsadmin will fail. You can change the wsadmin properties file or add parameters on the command line to attach wsadmin to a Node Agent server or application server.

For example, you can use the wsadmin interface with the following parameters:

```
wsadmin -conntype RMI -host localhost -port 2809
```

Alternatively, you can modify the wsadmin properties file with the application servers' or Node Agents' host and port as shown below:

```
#-----  
# The connectionType determines what connector is used.  
# It can be SOAP or RMI.The default is SOAP.  
#-----  
#com.ibm.ws.scripting.connectionType=SOAP  
com.ibm.ws.scripting.connectionType=RMI  
#-----  
# The port determines what port is used when attempting a connection.  
# The default SOAP port for a single-server installation is 8880  
#-----  
#com.ibm.ws.scripting.port=8880  
com.ibm.ws.scripting.port=2809  
#-----  
# The host determines what host is used when attempting a connection.  
# The default value is localhost.  
#-----  
com.ibm.ws.scripting.host=localhost  
#com.ibm.ws.scripting.host=ws-coral
```

However, any change made to the Node Agent or the application server will be lost when the Node Agent synchronizes its repository with the master repository once the Deployment Manager comes up again.

Enhancing Deployment Manager High Availability

To ensure that your cell configuration data is not lost, make backups of your cell on a regular basis (in case you were to experience an outage). WebSphere Application Server V5 provides a shell script command for this purpose: backupConfig.

There are several options to enhance the availability of the Deployment Manager, all are discussed below.

Monitor Deployment Manager process

Specialized software would need to be used to monitor the Deployment Manager process. When the Deployment Manager process crashes and stops unintentionally, the software monitoring the process will restart the Deployment Manager server automatically. This will minimize the process failure. However, doing so will not prevent the failure of the Deployment Manager due to any network, disk, operating system, JVM, and host machine problems.

Backup Cell

What we recommend is to have a an exact copy of your production cell, but where the Deployment Manager is installed and run on another system. Thus, if the you are unable to restart a failing Deployment Manager, you can end all the nodes and application servers running on the production cell and then start the application servers and node agents on the backup cell. This will allow you to perform application updates by using the backup Deployment Manager until you can get the production Deployment Manager up and running again.

Resources for Learning

Use the following links to find relevant supplemental information about various topics discussed in this chapter. The information resides on IBM and non-IBM Internet sites, whose sponsors control the technical accuracy of the information.

These links are provided for convenience. Often, the information is useful all or in part for understanding a topic discussed in this chapter. When possible, links are provided to technical papers and Redbooks that supplement the broad coverage of the release documentation with in-depth examinations of particular product areas.

WebSphere high availability

- *IBM WebSphere V5.0 Performance, Scalability, and High Availability: WebSphere Handbook Series* (July 2003)

<http://www.redbooks.ibm.com/redbooks/pdfs/sg246198.pdf>

This IBM Redbook discusses various options for scaling applications based on IBM WebSphere Application Server Network Deployment V5.0. It also discusses high availability. A must-read. A lot of the information in this document was obtained from this Redbook.

WebSphere

- *WebSphere Connection Pooling* (August 2001)

http://www.ibm.com/software/webservers/appserv/whitepapers/connection_pool.pdf

Comprehensive discussion on WebSphere connection pooling.

- *IBM WebSphere Version 5.0 System Management and Configuration* (April 2003)

<http://www.redbooks.ibm.com/redbooks/pdfs/sg246195.pdf>

This IBM Redbook provides the knowledge needed to implement WebSphere Application Server V5.0, Network Deployment runtime environment, to package and deploy Web applications, and to perform ongoing management of the WebSphere environment. For an in depth explanation of sessions and persistence, please see Chapter 14, “Configuring session management”.

- WebSphere Application Server for iSeries Home Page

<http://www.ibm.com/servers/eserver/series/software/websphere/wsappserver/>

The home page of WebSphere Application Server for iSeries.

CHAPTER 12

DATA CONSIDERATIONS FOR HIGH AVAILABILITY

Access to enterprise data is critical for most e-business applications. Since many of those applications must be available around the clock, the associated data must also be highly available.

In this chapter we will discuss what can be done to make data highly available, including the use of independent ASPs (iASPs) and replication technologies.

NOTE Critical to any attempt to make data highly available is a rock-solid database and a sound backup and recovery strategy. You should become familiar with the database availability features discussed in “OS/400 and System Software Availability Features” on page 55, including the autonomic computing characteristics built into the database. In addition, you should read “The Importance of Having a Backup and Recovery Strategy” on page 72.

Independent Disk Pools (aka Independent ASPs)

An independent disk pool is a collection of disk units that can be brought online or taken offline independent of the rest of the storage on a system, including the system disk pool, basic user disk pools, and other independent disk pools. An independent disk pool can be either switchable among multiple systems in a clustered environment or privately connected to a single system. The benefits, in both multi-system clustered environments and single-system environments, can be significant. For example, in a clustered environment, the use of independent disk pools can provide disk storage that is switchable amongst servers in the cluster, providing continuous availability of resources. In a single-system environment, independent disk pools could be used to isolate infrequently used data that does not always need to be present when the system is operational.

There are two basic environments in which you can take advantage of independent disk pools:

- Multi-system clustered environment

A group of servers in a cluster can take advantage of the switchover capability within clusters to move access to the independent disk pool from server to server. In this environment, an independent disk pool can be switchable when it resides on a switchable device: an external expansion unit (tower) or an input/output processor (IOP) on the bus shared by logical partitions.

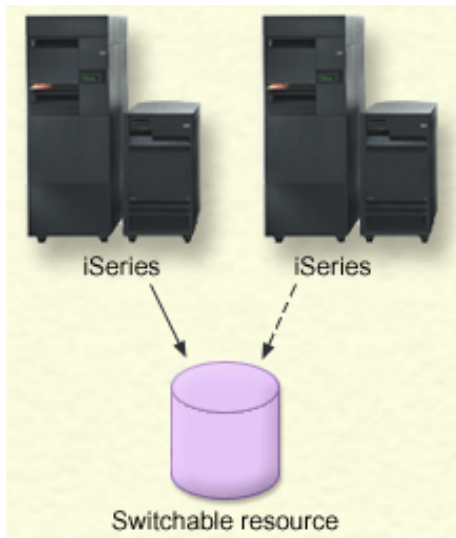
The server that owns, or is attached to, the switchable device containing the independent disk pool can then be switched, either automatically for some types of outages, or manually by administering a switchover.

- Standalone independent disk pools in a single-system environment

An independent disk pool in a single-system environment, with no clustering and no switchable devices, is said to be a private, standalone, or dedicated independent disk pool. While you cannot switch the access to the independent disk pool amongst servers in this environment, you can still isolate data in an independent disk pool, keeping it separate from the rest of the disk storage on the server. The independent disk pool can then be made available (brought online) and made unavailable (taken offline) as you see fit. This might be done, for example, to isolate data associated with a specific application program, or to isolate low-use data that is only needed periodically. This also allows you to isolate certain maintenance functions. Then, when you need to perform disk management functions that normally require the entire system to be at DST, you can perform them by merely varying off the affected independent disk pool.

Switched disk technologies can provide local failover capabilities when data is stored within one or more independent ASPs (see Figure 12.1 on page 211). There is only one copy of the data that is stored on disk units in one or more switchable towers that are connected via the system bus (HSL); therefore, this particular solution by itself cannot provide disaster recovery protection. But, for other types of failures that are not of the site disaster variety, it can provide local failover capabilities to increase the availability of data that reside within independent ASPs. It is important to remember to use either device parity or mirrored disk protection to protect against disk subsystem failures that could disrupt access to data stored within independent ASPs. System wellness monitoring provided through OS/400 cluster services and IP takeover addressing are used to accomplish the switchover and failover processing.

Figure 12.1 Switched disk using iASPs



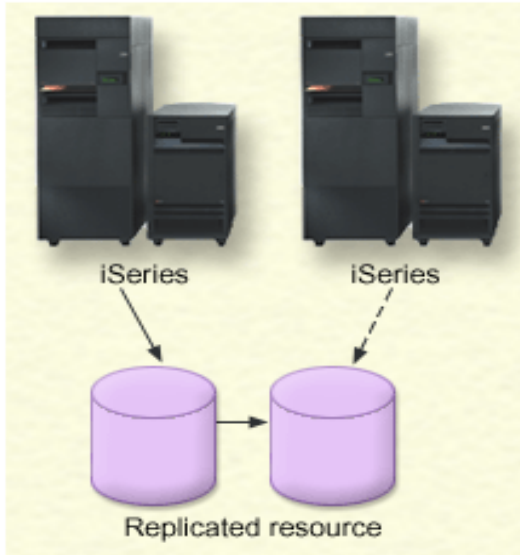
The benefits of using independent disk pools include:

- High availability of data stored within a switchable independent ASP
- Elimination of the extra cost and complexity of data replication solutions since there is only one copy of the data that is switchable amongst nodes in the cluster
- Isolation of the impacts of user application failures or disk subsystem problems due to the independence of the disk pools from one another
- Isolation of data associated with specific applications or certain groups of users
- Ability to perform application maintenance that will not affect other independent disk pools or the system ASP
- In single-system environments, the independent, private disk pools can be taken offline or brought online as needed without performing an IPL.
- Isolation of low-use or archival data
- Reduction in time for RCLSTG and IPL operations if some independent ASPs can be left offline.
- Ability to perform a RCLSTG operation against individual, independent disk pools
- Save and restore operations can be done on an independent ASP basis

Replication Solutions for High Availability

Replication is a process of maintaining a defined set of data in more than one location. It involves copying designated changes from one location (a source) to another (a target), and synchronizing the data in both locations. The source and target can be in logical servers that are on the same machine or on different machines in a distributed network.

Figure 12.2 Replication of resources



Replication is the foundation for any high availability environment. Any replication technology worth its salt must ensure the integrity and consistency of the data that is being replicated. On the iSeries server, this is achieved with journaling and commitment control. Customers who understand the need for highly available systems implement these features on their systems and in their applications.

Journaling allows changes in the database or an IFS file to be recorded and stored. Replication software scrapes journal transactions from the system audit journal and database journals as they occur on the production system and sends them to the target (recovery) system where the entries are applied in sequence to the replicated objects. Alternatively, the OS/400 remote journal support can be used as a more efficient transport mechanism for the journal entries while the replication software handles the journal apply processing on the target system. OS/400 remote journaling has two delivery modes:

- Synchronous delivery mode

Synchronous delivery means that the journal entry is replicated to the target system concurrently with the entry being written to the local receiver on the source system. The entry is known on the target system, in main storage, prior to returning control to the user application that deposited the journal entry on the source system. Therefore, the target system knows of all journal entries as they are being made in real-time on the source system. Using this mode allows for recovery without losing journal entries on the target system if the source system fails. Providing journal entries synchronously to a target system will have some impact to the journaling throughput on the local production system. Therefore, it makes the most sense to use it when the production and target database systems are connected via LAN or HSL transports or when the volume of journal entry deposits are such that the synchronous requirements still result in acceptable user or application response times on the

production system. Synchronous delivery mode is only supported when a remote journal is associated with a local journal.

- Asynchronous delivery mode

Replicating a journal entry asynchronously means that the journal entry is replicated to the target system after control is returned to the application depositing the journal entry on the source system. Using this mode allows for recovery that may lose some journal entries if the source system fails. However, this mode has less impact to the journal throughput on the local system in comparison with the synchronous mode. Most of the third-party software products tend to mirror transactions asynchronously. But, if the application dependent data is critical and the loss of journal entries can impact your business, then you should seriously consider using the synchronous delivery mode available with OS/400 remote journaling.

There are several advantages to using software replication that is performed at the database transactional level. Since journaling and commitment control are key to this implementation, the integrity of the databases on the target system is ensured. Rollback processing occurs when necessary to honor commitment control boundaries that are defined within the user applications. Secondly, only changes to the objects that are being replicated are sent across the network, which can be quite helpful in a WAN environment with cost considerations and limited bandwidth. Thirdly, save operations or queries can be performed against the databases on the target system. With respect to the save operations, one would end the journal entry apply process on the target system and then restart the apply process once the backup is complete. During that backup operation, end users and applications can still access the database on the production system. Lastly, one could implement a configuration that involves a synchronous delivery mode for local replica processing and an asynchronous delivery mode for another copy of the database located at a remote site for disaster recovery purposes. A backup of the data can then be done against one of the database copies while still maintaining real-time replication between the other two systems.

You must decide on a software technology to use for replication. The following solutions are available for achieving replication:

- Business partners products

Data replication software from business partners enables you to replicate objects across multiple nodes. See “Software Replication Products” on page 214 for further information.

- A custom-written replication application

IBM journal management provides a means by which you can record the activity of objects on your system. You can write an application taking advantage of journal management to achieve replication.

Once you have chosen a mechanism to achieve replication, you must also determine which systems to use for replication. Key considerations for determining which systems to use for replication are:

- Performance capacity
- Disk capacity
- Critical data
- Disaster prevention

If your system fails over, you need to know what data and applications you have running on your primary system and your backup system. You want to put the critical

data on the system that is most capable of handling the workload in case it fails over. You do not want to run out of disk space. If your primary system runs out of space and fails over, it is highly likely that your backup system is also going to fail over due to lack of disk space. To ensure your data center is not completely destroyed in case of a natural disaster, such as a flood, tornado, or hurricane, you should locate the replicated system in a remote location.

Software Replication Products

Cluster middleware is the name given to the group of applications that provide the replication and management of application data between iSeries servers and that provide cluster management utilities.

Data replication by cluster middleware products in an iSeries cluster environment consists of taking full advantage of OS/400 journaling functions for database, IFS objects, data areas, and data queues. This provides the most up-to-date copy of the data possible. When synchronous remote journaling is configured, there is no loss of in-flight data at the point of failure since the latest changes to the database are first recorded in the remote journal on the backup server. Cluster middleware products use various techniques to save changes to other data types, including use of the security audit journal.

Cluster management is where cluster middleware products differentiate themselves from other available iSeries business partner products. Specially designed OS/400 objects are used to manage not only the replication process, but also each application in the cluster and the cluster itself. The monitoring of the cluster is controlled by iSeries. Finally, a user interface is provided to manage the cluster nodes and the applications. IASP and switch disk towers are part of the cluster architecture and their management is also supported by cluster management utilities provided by cluster middleware products.

The following iSeries business partner applications qualify as Cluster Middleware:

- iCluster from Data Mirror
<http://www.datamirror.com/>
- MIMIX Cluster Server from Lakeview Technology
<http://www.lakeviewtech.com/>
- Vision Suite from Vision Solutions
<http://www.visionsolutions.com/>

In addition, the following data recovery/replication solution business partners have products that replicate data to a second iSeries server for the use of high availability and disaster recovery:

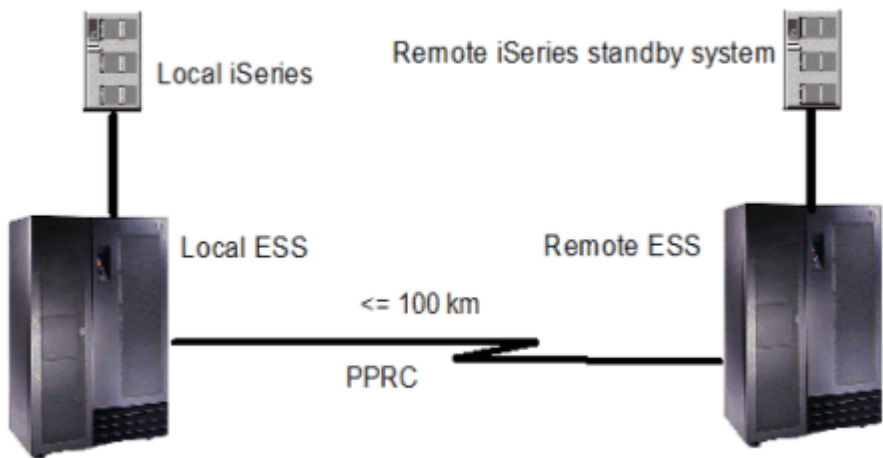
- iTera
<http://www.iterainc.com/>
- Maximum Availability
<http://www.maximumavailability.com/>

iSeries SAN Solutions

In addition to storage consolidation advantages in a heterogeneous server environment, IBM's Enterprise Storage Server can provide disaster recovery protection for enterprise data.

It is accomplished through the creation of resilient devices within the storage server as well as by creating one or more copies of the data among multiple storage servers. RAID-5 or RAID-10 disk technology is used with redundant and switchable hardware to provide highly resilient data.

Figure 12.3 ESS replication



ESS Copy Services

The ESS views data as a series of disks within a LUN. When data changes within a block, that block is mirrored to the backup ESS via peer-to-peer remote copy (PPRC) or to another LUN within the same ESS using FlashCopy. An entire image of the production volumes is maintained. There is an iSeries Copy Services for ESS Toolkit available to assist you with Copy Services tasks, including FlashCopy and PPRC support for independent ASPs. FlashCopy and PPRC can be used with both traditional ASPs and independent ASPs.

Since the implementation is at the storage server level, this implementation doesn't directly address application resiliency. Moreover, in the event of a site disaster, the target volume remains closed until the data is logically incorporated into the backup iSeries server and an OS/400 IPL is performed. Steps involving remote load source recovery procedures, an IPL, and network reconfiguration are required before applications can access the data again after a site disaster.

Synchronous, rather than asynchronous, connections are highly recommended for PPRC solutions that involve critical enterprise data. The remote ESS can be up to 100 km from the source ESS using ESCON connectivity with synchronous PPRC. Even greater distances are possible with Fibre Channel protocol transports that are available for PPRC in the latest version of ESS. Performance analysis should be done as part of a PPRC install to understand the response time that can be expected at various distances and with various bandwidth networks.

Using a SAN solution for replication does have some advantages. It replicates a complete system image versus just a set of predefined objects. A large data volume can be refreshed relatively quickly. Thirdly, the PPRC technology is independent from the operating system; therefore, you can utilize a common methodology for data replication across heterogeneous server platforms. Lastly, if you have already made an investment in IBM ESS technology, it makes sense to explore the Copy Services options that can improve the availability of enterprise data.

Remote OS/400 Mirroring with ESS

As an alternative to PPRC, one can implement remote OS/400 mirroring with ESS. It replicates an entire disk pool image versus just a set of predefined system objects. Mirroring is turned on via the Dedicated Service Tools (DST) menu. OS/400 will review the disk-related hardware available and create a mirroring configuration that provides the maximum protection possible, that is, tower-level, bus-level, IOP-level (Input-Output Processor-level), or disk-level protection.

With ESS, OS/400 mirroring can be used to create a PPRC-like solution that offers slightly better function at a lower price than PPRC. The solution has distance limitations similar to PPRC for performance reasons. In this configuration, OS/400 has half of its disks attached locally and half attached remotely. The local set of disks can be either integrated disk or ESS.

The remote disk is on ESS to allow the extended distance.

It is critical that the hardware be attached in such a way that each mirrored pair has one unit on the local disk and one unit on the remote disk. To do this, a minimum of 2 buses is required on the primary system. The bus that holds the fibre cards for the remote ESS is marked as a remote bus by renaming it to start with the letter “R”. OS/400 will then attempt to make the appropriate pairs, assuming suitable disk exists for it to do so. For maximum protection, it is also beneficial to arrange the hardware to achieve tower-level protection when possible, just as would be done when mirroring integrated disk.

With this type of configuration there are only two copies of the load source: one on integrated disk and a mirror on the remote ESS. Contrast this with PPRC where the load source on integrated disk is mirrored into the local Shark, then propagated via PPRC to a third copy on the remote ESS.

Once in operation, this solution is very similar to PPRC in the coverage it provides if the primary system CPU fails. A secondary system would need to be available, and it would be attached to the remote copy of the mirrored disks using a remote load source recovery procedure followed by an abnormal IPL. Any workload previously running on the secondary system would be obliterated in the process, so it is usual to have the secondary system sitting idle. Similar to the PPRC scenario, a clever solution is to use a partition in a logically partitioned system for this, since its resources can be allocated elsewhere when not needed rather than sitting idle.

The key difference between this solution and PPRC is in the scenario where a component in the disk subsystem fails. Examples would include a fibre card in the iSeries or ESS, a fibre cable, a switch, or some disk drives. In the PPRC scenario, the primary system would fail since it was missing some of its disk, and a cutover to the secondary system would be required in order to accommodate the users. By comparison, in the remote mirroring scenario, the system would continue running, using the data on the other unit in the mirrored pair. In addition, this solution provides equivalent function to multi-path in this environment; that is, it provides alternate paths with load balancing and failover.

The other difference between the PPRC and remote mirroring solutions is cost. In each case, a duplicate copy of the disk and a secondary CPU are required. The difference is that the mirroring software is included with OS/400 at no extra charge, whereas PPRC is a chargeable function on the ESS. Customers who already own PPRC should do their own calculations to determine which solution is most cost-effective for them, and contrast that with the function provided.

The OS/400 remote mirroring function would be attractive to a customer who was interested in PPRC, but wanted the increased resilience provided by this solution in terms of multi-path like function, and coverage for a disk/SAN failure without having to do a failover procedure.

Summary

iSeries data can be made more highly available through a number of different technologies. iSeries single-system availability can be improved through various autonomic, journaling, and save-while-active features. For elimination of many single points of failure, one can utilize independent disk pools that can be switched between nodes in an OS/400 cluster. To meet full disaster recovery requirements, one can use HABP replication software involving a remote server or take advantage of PPRC functions between multiple IBM storage servers. It is important to note that each of these solutions does not exclude concurrent usage of the other alternatives. In fact, a combination of two or more of these technologies could be chosen to best meet your high-availability objectives. For instance, you could use HABP software to replicate data stored within independent ASPs. Another hybrid possibility may include usage of synchronous PPRC between two ESS systems as well as utilization of HABP software for asynchronous replication between the production copy and a backup volume for additional backup and disaster recovery advantages.

Resources for Learning

Use the following links to find relevant supplemental information about various topics discussed in this chapter. The information resides on IBM and non-IBM Internet sites, whose sponsors control the technical accuracy of the information.

These links are provided for convenience. Often, the information is useful all or in part for understanding a topic discussed in this chapter. When possible, links are provided to technical papers and Redbooks that supplement the broad coverage of the release documentation with in-depth examinations of particular product areas.

Independent ASPs

- Independent disk pools
<http://publib.boulder.ibm.com/series/v5r2/ic2924/index.htm?info/rzaly/rzalyoverview.htm>
This iSeries Information Center topic contains information on iASPs.
- *iSeries Independent ASPs: A Guide to Moving Applications to IASPs* (May 2003)
<http://www.redbooks.ibm.com/redbooks/pdfs/sg246802.pdf>
This IBM Redbook explains how to install and configure the new independent auxiliary storage pool (IASP) functionality of OS/400 V5R2. It is designed to help IBM technical professionals, Business Partners, and Customers understand and implement IASP in the IBM iSeries server and under OS/400 V5R2.
In addition, this redbook provides the background information that is necessary to plan, implement, and customize this functionality to your particular environment. It provides advice on running native OS/400 applications with either application data or most application objects residing in an IASP. Considering you can also use IASPs in a cluster environment, this redbook shows you the basic steps to make your IASP switchable between two iSeries servers in a high-speed link (HSL) loop.

Storage area networks (SANs)

- IBM TotalStorage Home Page
<http://www.storage.ibm.com/>
IBM TotalStorage home page.
- *IBM iSeries in Storage Area Networks: Implementing Fibre Channel Disk and Tape with iSeries* (December 2002)
<http://www.redbooks.ibm.com/redpieces/pdfs/sg246220.pdf>
The material in this IBM Redbook contains information on how to plan and implement a SAN-based solution on the iSeries server.
- *Introduction to Storage Area Networks* (April 2003)
<http://www.redbooks.ibm.com/redbooks/pdfs/sg245470.pdf>
This IBM redbook gives an introduction to SAN. It illustrates where SANs are today, who are the main industry organizations and standard bodies active in SANs, and it positions IBM's approach to enabling SANs in its products and services.
- *IBM TotalStorage Solutions for Disaster Recovery* (January 2004)
<http://www.redbooks.ibm.com/redbooks/pdfs/sg246547.pdf>
This IBM Redbook will help you design a Disaster Recovery solution and presents a Disaster Recovery Solution Selection Methodology to assist in this process. It also provides information on the IBM TotalStorage products which can be used in your solution.
- *IBM TotalStorage Enterprise Storage Server Implementing ESS Copy Services in Open Environments* (January 2004-Draft)
<http://www.redbooks.ibm.com/redpieces/pdfs/sg245757.pdf>

This IBM Redbook describes the copy functions available with the IBM TotalStorage Enterprise Storage Server (ESS). The powerful ESS Copy Services functions are explained in detail, and their respective characteristics are thoroughly covered. This redbook also gives information on how to manage the various ESS Copy Services functions, and finally discusses their implementations.

- *The IBM TotalStorage Solutions Handbook* (January 2004-Draft)

<http://www.redbooks.ibm.com/redpieces/pdfs/sg245250.pdf>

This IBM Redbook provides an overview of the IBM TotalStorage products.

IBM assistance

- IBM eServer and IBM TotalStorage Services

<http://www.ibm.com/servers/eserver/services/>

IBM services team that can help with building quality solutions.

- iSeries Technology Center

<http://www.ibm.com/servers/eserver/series/service/itc/>

The iSeries Technology Center (iTC) provides world class technical marketing assistance, education and implementation assistance for select technologies to IBM Sales and Service personnel, Business Partners, customers and consultants worldwide.

CHAPTER 13

BACK-END APPLICATION CONSIDERATIONS FOR HIGH AVAILABILITY

In the previous chapters, we discussed application high availability in the context of J2EE applications running within the WebSphere application server. However, in most customer sites today, transactions initiated by clients and handled by J2EE applications running in the application server include interactions between the J2EE application and back-end applications. No matter how highly available your infrastructure is, no matter how resilient the data is, if the back-end application is not available, transactions will fail and your business may be out a lot of money.

The availability principles discussed in “High Availability Principles” on page 10 apply to back-end applications as well. To summarize:

- 1** Every entity must be redundant.
- 2** Each entity must be monitored for failure.
- 3** A failed entity must not receive any work.

For an e-business Web site, the focus of application design must include the requirements of the customers for performance, availability, and reliability. Availability requirements must share the forefront of application design. Existing applications and processes may need to be redesigned with these increased requirements in mind. The consequences of ignoring or minimizing the requirement should no longer be tolerated.

The process for designing applications should be enhanced to meet the availability requirement:

- Educate your designers and developers about customer requirements and the cost of an outage.
- Include availability objectives in application design; involve availability experts early.
- Exploit availability features of current products; maintain installed products at current levels.
- Select components that have characteristics suitable for the negotiated availability requirement.

- Consider data access and data maintenance.
- Design to keep the scope of failure small:
 - Minimize impact to other components
 - Isolate and contain important functions such as the session engine and databases
 - Provide selective redundancy for those functions that have broad impact to the rest of the system (for example, directories and databases)
- Design application recovery and initialization processes to be fast and easy on everybody (customers, operators, administrators).
- Seek alternatives to applications that require planned outages.
- Design batch processes that do not affect online applications.
- Employ standard processes and procedures to help improve communications and reduce the likelihood of errors. Naming conventions, for example, help reduce errors and can also provide a built-in check during changes and other activities.
- Acquire monitoring and problem determination tools that are robust and easy to use. Implement application-level monitoring as appropriate.

Robust testing is necessary to ensure conformance to availability design specifications and to validate that a component is ready to be introduced into the production environment.

Programming Techniques for High Availability

Detailed programming recommendations for making back-end applications highly available are beyond the scope of this document. However, some techniques that are used for making an application highly available are listed below.

NOTE Implicit in the techniques listed below is ensuring that any data used by the back-end application is highly available. See Chapter 12, “Data Considerations for High Availability” on page 209 for more information on making data highly available.

Commitment Control

The use of commitment control gives you the ability to restart applications if a process, an activation group within a process, or the system ends abnormally. The application can be started again with the assurance that no partial updates are in the database due to incomplete logical units of work from a prior failure.

Application Checkpointing

In general, application checkpointing is a method used to track completed steps and pick up where a process last left off before a system or application failure. Typically a log file is used to determine the completed steps and at what points a transaction failed. During recovery, a process would read the log file and perform any required recovery actions.

OS/400 Clusters

OS/400 clustering can be used to make an application highly available. This involves the creation of an application cluster resource group (CRG), in addition to an exit program for the application CRG. For more information on OS/400 clusters, see “Clusters” on page 65.

CHAPTER 14

DISASTER RECOVERY CONSIDERATIONS FOR HIGH AVAILABILITY

The definition of what constitutes a disaster is being driven by the rate of technological advance. As the definition of what constitutes a disaster has changed, so too have the methods of assuring recoverability. As we will see, for some organizations the ability to recover data can be of secondary importance to assuring that their data remains available for use.

Historically, in the former years of data processing, Disaster Recovery referred to the recovery of an organization's central processing facility, the so-called "glass house." Recovery efforts were almost always triggered by a fire, flood, storm, or other physical devastation. These hazards still exist, of course; however, a disaster in today's frame of reference can mean the corruption of files or any event that makes the organization's data unavailable, even for relatively short periods of time. Data disruptions for some industries, such as financial institutions, can result in staggering financial losses.

The methods available to assist recovery efforts are further hampered by application architectures such as distributed applications, distributed processing, distributed data, and hybrid computing environments.

Then there is the issue of data volume. Applications such as decision support, data warehousing, data mining, and customer resource management can require petabyte-size investments in online storage. More and more data means more and more to manage and more and more to recover. On top of all this, you must be able to recover your essential business processes in less time than what was traditionally considered possible. Revenue losses in some environments are measured in millions of dollars per hour.

Data recovery and its corollary, high availability, no longer lend themselves to a one-dimensional approach. The complex IT infrastructure of most installations has just outstripped the ability of most shops to respond in the way they did just a few years ago.

Do we mean that all organizations need a real-time recovery strategy in place? No, some organizations by virtue of their processing needs and their level of reliance on automating these processes can withstand unplanned outages for a day or even several days without severe adverse consequences. However, whereas the loss of phone service for a catalog retailer might be sufferable for a short period of time, as the outage enters

day two and three the revenue losses begin to grow exponentially. Research conducted by the University of Minnesota has demonstrated that two companies out of five that suffer an unprepared catastrophic event are unable to continue their business. Of those that are able to recover in some limited fashion, one out of three are out of business within two years. These companies ceased to exist.

Before we go on, we need to clarify the terms Disaster Recovery and business continuity. These terms are sometimes used interchangeably, as are business resumption and contingency planning. The Disaster Recovery plan is limited to the *preservation and recovery of the IT infrastructure only*; this is the scope of this chapter. It is only one component of a business continuity plan. The business continuity plan has a much larger focus and includes such items as a crisis management plan, human resources management, and so forth.

Disasters: The Old, the New, and the Planned

Figure 14.1 on page 226 catalogs a series of unplanned outages. We know that you can probably add one or two from your own experience. Also perhaps we could have made our list more inclusive by including subtopics or cross-referencing some of our examples. Vandalism might include hacker and virus threats. We could have added worms, trojan horses, distributed denial-of-service attacks, and industrial espionage under a hacker subtopic. The list of things that can go wrong is seemingly endless.

We did include relocation delay, which is really a sort of planned outage that includes data center relocations, or the migration back to the primary computing facility from a hot or cold site. Descriptions of disasters usually imply the element of surprise or the unexpected. We want to draw specific attention to these well-planned outages because they very often prove that “Murphy” is alive and well. In the end, does it really matter to your customer why the data is unavailable to them?

Figure 14.1 Murphy’s shopping list of possible disasters

Q: I want Disaster Recovery.....you ask: “what kind?”

A/C Failure	Evacuation	Low Voltage	Sprinkler Discharge
Acid Leak	Explosion	Microwave Fade	Static Electricity
Asbestos	Fire	Network Failure	Strike Action
Bomb Threat	Flood	PCB Contamination	S/W Error
Bomb Blast	Fraud	Plane Crash	S/W Ransom
Brown Out	Frozen Pipes	Power Outage	Terrorism
Burst Pipe	Hacker	Power Spike	Theft
Cable Cut	Hail Storm	Power Surge	Toilet Overflow
Chemical Spill	Halon Discharge	Programmer Error	Tornado
CO Fire	Human Error	Raw Sewage	Train Derailment
Condensation	Humidity	Relocation Delay	Transformer File
Construction	Hurricane	Rodents	UPS Failure
Coolant Leak	HVAC Failure	Roof Cave In	Vandalism
Cooling Tower Leak	H/W Error	Sabotage	Vehicle Crash
Corrupted Data	Ice Storm	Shotgun Blast	Virus
Diesel Generator	Insects	Shredded Data	Water (Various)
Earthquake	Lighting	Sick building	Wind Storm
Electrical Short	Logic Bomb	Smoke Damage	Volcano
Epidemic	Lost Data	Snow Storm	

Disaster Cost

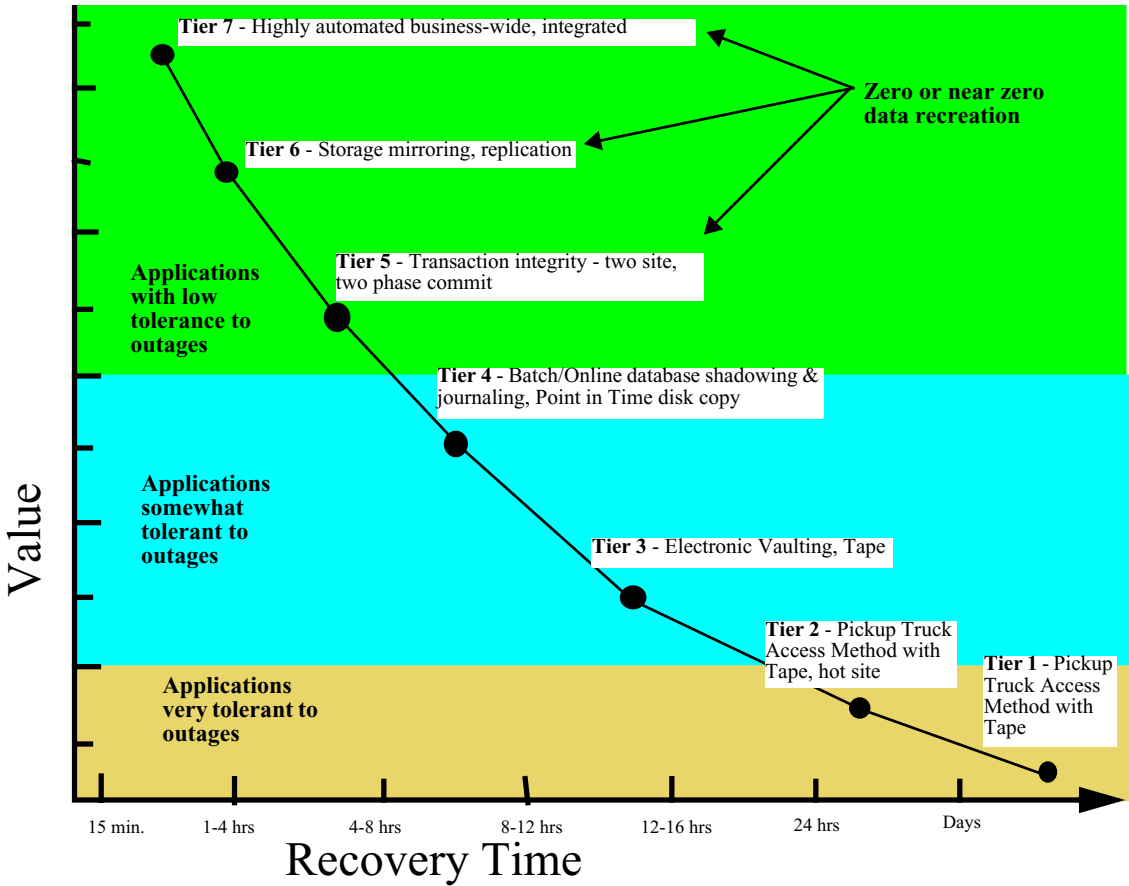
The results of any type of disaster can be costly on a variety of levels. Not only are there losses directly related to the value of the assets lost, such as data, infrastructure, network, and so forth; but also consequential losses that remain long after recovery or restart. They include organizational image, stock value, and loss of customers and market share.

The Tiers of Disaster Recovery

In 1992, the SHARE user group in the United States, in combination with IBM, defined a set of Disaster Recovery tier levels. This was done to address the need to properly describe and quantify various different methodologies for successful mission-critical computer systems Disaster Recovery implementations. Accordingly, within the IT Business Continuance industry, the tier concept continues to be used, and it is very useful for describing today's Disaster Recovery capabilities. They need only to be updated for today's specific Disaster Recovery technologies and associated Recovery Time Objective (RTO) and Recovery Point Objective (RPO). The RTO is how long should it take to recover to the point for an end user to be online again; the RPO, is how much data the user can afford to recreate after the disaster.

The Tiers chart, shown in Figure 14.2 on page 228, gives a generalized view of the tiers. As the recovery time becomes shorter, then more aggressive Disaster Recovery technologies must be applied to achieve that RTO (carrying with them their associated increase in value and capital cost).

Figure 14.2 Tiers of disaster recovery (tiers based on SHARE definitions)



The concept and shape of the Tiers chart continues to apply even as the scale of the application or applications changes. The general relationship of the various tiers and Disaster Recovery technologies to each other remains the same. Finally, although some Disaster Recovery technologies fit into multiple tiers, clearly there is not one Disaster Recovery technology that can be optimized for all the tiers.

Of course, your technical staff can and should, when appropriate, create a specific version of the Tiers chart for your particular environment. After the staff agrees on what tier or tiers and corresponding RTO a solution delivers for your enterprise, then Disaster Recovery technical evaluation and comparisons are much easier, and the technology alternatives can be tracked and organized in relation to each other. Although the technology within the tiers has obviously changed through time, the concept continues to be as valid today as when it was first described by the U.S. SHARE user group.

The following sections provide an overview of each of the tiers.

Tier 0: Do Nothing, No Off-Site Data

Tier 0 is defined as a single site data center environment having no requirements to backup data or implement a Disaster Recovery Plan.

On this tier, there is no saved information, no documentation, no backup hardware, and no contingency plan. There is therefore no DR capability at all. In our experience, some customers still reside in this tier. For example, while some customers actively make backups of their data, these backups are left onsite in the same computer room, or occasionally are not removed from the site due to lack of a rigorous vaulting procedure. A customer data center residing on this tier is exposed to a disaster from which they may never recover their business data!

The typical length of recovery time in this instance is unpredictable. In many cases complete recovery of applications, systems, and data is never restored.

Tier 1: Offsite Vaulting (PTAM)

A Tier 1 installation is defined as having a disaster recovery plan, backs up and stores its data at an offsite storage facility and has determined some recovery requirements. This environment does not have a site at which to restore its data, nor the necessary hardware on which to restore the data, for example, compatible tape devices.

Because vaulting and retrieval of data is typically handled by couriers, this tier is described as the Pickup Truck Access Method (PTAM). PTAM is a method used by many sites, as this is a relatively inexpensive option. It can, however, be difficult to manage, that is, it is difficult to know exactly where the data is at any point. There is probably only selectively saved data. Certain requirements have been determined and documented in a contingency plan and there is optional backup hardware and a backup facility available.

Recovery is dependent on when hardware can be supplied, or possibly when a building for the new infrastructure can be located and prepared.

While some customers reside on this tier and are seemingly capable of recovering in the event of a disaster, one factor that is sometimes overlooked is the recovery time objective (RTO). For example, while it may be possible to eventually recover data, it may take several days or weeks. An outage of business data for this period of time can have an impact on business operations that lasts several months or even years (if not permanently).

Examples of Tier 1 disaster recovery solutions include: Pickup Truck Access Method (PTAM), Disk Subsystem or Tape based mirroring to locations without processors.

Tier 2: Offsite Vaulting with a Hot Site (PTAM + Hot Site)

Tier 2 encompasses all requirements of Tier 1 (offsite vaulting and recovery planning) plus it includes a hot site. The hot site has sufficient hardware and a network infrastructure able to support the installation's critical processing requirements. Processing is considered critical if it must be supported on hardware existing at the time of the disaster.

Tier 2 installations rely on a courier (PTAM) to get data to an offsite storage facility. In the event of a disaster, the data at the offsite storage facility is moved to the hot site and

restored onto the backup hardware provided. Moving to a hot site increases the cost but reduces the recovery time significantly. The key to the hot site is that appropriate hardware to recover the data is present and operational.

Examples of Tier 2 disaster recovery solutions include: PTAM with Hot-site available.

Tier 3: Electronic Vaulting

Tier 3 encompasses all the components of Tier 2 (offsite backups, disaster recovery plan, hot site) and, in addition, supports electronic vaulting of some subset of the critical data. Electronic vaulting consists of electronically transmitting and creating backups at a secure facility, moving business-critical data offsite faster and more frequently than traditional data backup processes allow. The receiving hardware must be physically separated from the primary site and the data stored for recovery should there be a disaster at the primary site.

The hot site is kept running permanently, thereby increasing the cost. As the critical data is already being stored at the hot site, the recovery time is once again significantly reduced.

Examples of Tier 3 disaster recovery solutions include: Electronic Vaulting of Data.

Tier 4: Point-in-time Copies

Tier 4 solutions are used by businesses who require both greater data currency and faster recovery than users of lower Tiers. Rather than relying largely on shipping tape, as is common on the lower Tiers, Tier 4 solutions begin to incorporate more disk based solutions.

Several hours of data loss is still possible, but it is easier to make such point-in-time (PIT) copies with greater frequency than replicating data through tape based solutions.

Examples of Tier 4 disaster recovery solutions include: Batch/Online Database Shadowing and Journaling, FlashCopy, Peer-to-Peer Virtual Tape Server, and iSeries IASPs with FlashCopy.

Tier 5: Transaction Integrity

Tier 5 encompasses all the requirements of Tier 4 (offsite backups, disaster recovery plan, electronic vaulting, and active secondary site), and in addition, will maintain selected data in image status (updates will be applied to both the local and the remote copies of the database within a single-commit scope). Tier 5 requires that both the primary and secondary platforms' data be updated before the update request is considered successful.

Tier 5 also requires partially or fully dedicated hardware on the secondary platform with the ability to automatically transfer the workload over to the secondary platform. We now have a scenario where the data between the two sites is synchronized by remote two-phase commit. The critical data and applications are therefore present at both sites and only the in-flight data is lost during a disaster. With a minimum amount of data to recover and reconnection of the network to implement, recovery time is reduced significantly.

Examples of Tier 5 disaster recovery solutions include: software, two site, and Two-phase commit.

Tier 6: Zero or Little Data Loss

Tier 6 Disaster Recovery solutions maintain the highest levels of data currency. They are used by businesses with little or no tolerance for data loss and who need to restore data to applications rapidly. These solutions have no dependence on the applications to provide data consistency.

Examples of Tier 6 disaster recovery solutions include: PPRC, Asynchronous Cascading PPRC, replication software, and iSeries IASPs with PPRC.

Tier 7: Highly Automated, Business Integrated Solution

Tier 7 solutions include all the major components being used for a Tier 6 solution with the additional integration of automation. This allows a Tier 7 solution to ensure consistency of data above that which is granted by Tier 6 solutions. Additionally, recovery of the applications is automated, allowing for restoration of systems and applications much faster and more reliably than would be possible through manual Disaster Recovery procedures.

Examples of Tier 7 disaster recovery solutions include: iSeries High Availability Business Partner software.

Selecting the Optimum Disaster Recovery Solution

It is important to understand that the cost of a solution must be in reasonable proportion to the business value of IT. You do not want to spend more money on a Disaster Recovery solution than the financial loss you would suffer from a disaster.

Based on the following objectives it becomes relatively simple to decide, as a business, which solution to select according to how much you can afford to spend and the speed at which you need your data recovered. The quicker the recovery the higher the cost:

- Recovery Time Objective (RTO)
How long can you afford to be without your systems?
- Recovery Point Objective (RPO)
When it is recovered, how much data can you afford to recreate?
- Degraded operations objective (DOO)
What will be the impact on operations with fewer data centers?
- Network Recovery Objective (NRO)
How long to switch over the network?

Normally all the components that make up continuous availability are situated in the same computer room. The building, therefore, becomes the single point-of-failure. While you must of course be prepared to react to a disaster, the solution you select may be more of a recovery solution than a continuous-availability solution.

A recovery solution must then be defined by making a trade-off among implementation costs, maintenance costs, and the financial impact of a disaster, resulting from performing a business impact analysis of your business.

Resources for Learning

Use the following links to find relevant supplemental information about various topics discussed in this chapter. The information resides on IBM and non-IBM Internet sites, whose sponsors control the technical accuracy of the information.

These links are provided for convenience. Often, the information is useful all or in part for understanding a topic discussed in this chapter. When possible, links are provided to technical papers and Redbooks that supplement the broad coverage of the release documentation with in-depth examinations of particular product areas.

Disaster recovery

- *A Disaster Recovery Solution Selection Methodology* (February 2004)

<http://www.redbooks.ibm.com/redpapers/pdfs/redp3847.pdf>

In this Redpaper, a suggested Disaster Recovery Solution Selection Methodology that is designed to provide assistance to this problem is provided. The intent of this methodology is to allow one to navigate the seemingly endless permutations of Disaster Recovery technology quickly and efficiently, and to identify initial preliminary, valid, cost-justified solutions.

- *IBM TotalStorage Solutions for Disaster Recovery* (January 2004)

<http://www.redbooks.ibm.com/redbooks/pdfs/sg246547.pdf>

This Redbook will help you design a Disaster Recovery solution and presents a Disaster Recovery Solution Selection Methodology to assist in this process. It also provides information on the IBM TotalStorage products which can be used in your solution.

IBM offerings and services

- IBM Business Continuity and Recovery Services

<http://www.ibm.com/services/continuity/recover1.nsf/documents/home>

IBM Business Continuity and Recovery Services can help you map your current infrastructure and design a customized recovery plan for a variety of disaster scenarios. IBM professionals also have the knowledge and experience to help you recover from an actual disaster situation.

CHAPTER 15

RECOMMENDED TOPOLOGIES

The preceding chapters described the various technologies which combine to provide a scalable and highly availability e-business solution on OS/400. These technologies work in a cooperative manner through well defined interactions. This chapter compiles all this information and presents a few topologies which take full advantage of WebSphere flexibility coupled with OS/400 strengths. The following is a list of the technologies used:

- WebSphere (Network Deployment)
 - Two Network Deployment Cells
 - Clustered WebSphere Application Servers
- Load Balancing using Custom Advisors
- Two or more Apache Web Servers
- LPAR
 - iSeries Linux partition to run Network Dispatcher
 - OS/400 partitions for http servers, application servers, and database
- HABP (High Availability Business Partner) replication software
- IASP (Independent Auxiliary Storage Pool)
- OS/400 clustering
- Remote Journal
- Virtual IP to assure multiple NIC redundancy

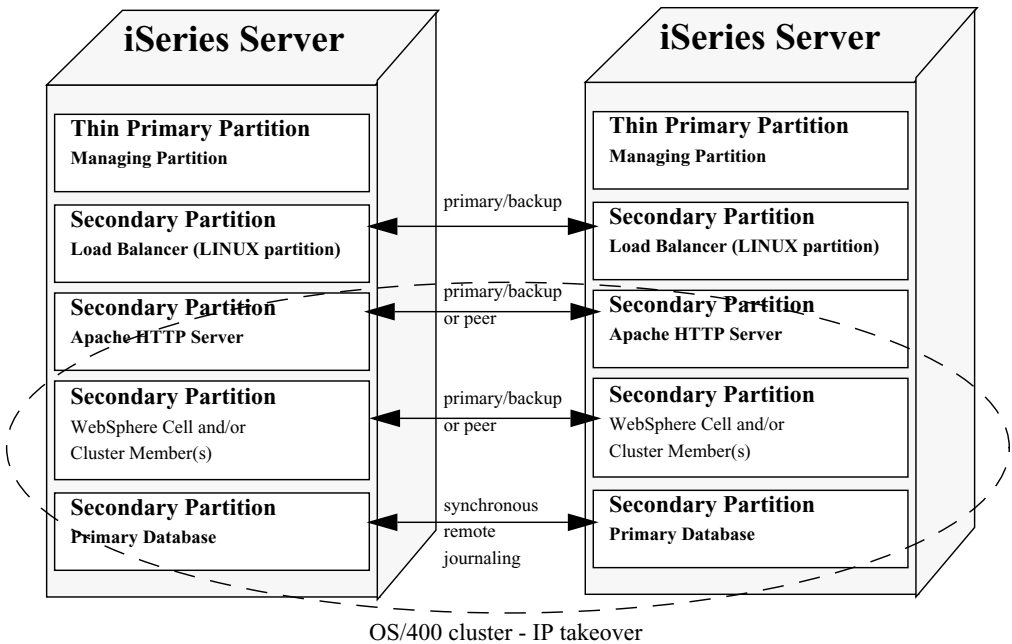
Before presenting the recommended topologies we shall briefly discuss LPAR and HA DB particulars which apply to all of the topologies

LPAR

LPAR is utilized in all of the recommended topologies. LPAR is becoming a standard amongst the OS/400 customer community. LPAR facilitates server consolidation

thereby cutting costs and administration for processors, memory, buses, I/O processors (IOPs), and I/O adapters (IOAs). Figure 15.1 illustrates the general LPAR configuration used in all of our topologies.

Figure 15.1 General LPAR configuration used in HA topologies



LPAR provides both data and process isolation yet allows centralized administration of system resources.

HA Database Topologies

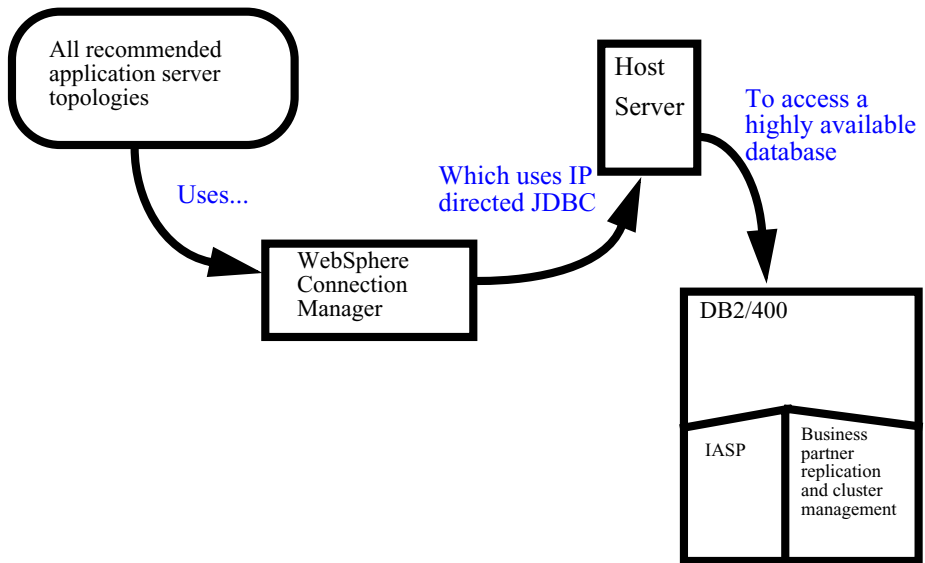
Data management is a critical part of any e-business topology. The availability of this data directly impacts WebSphere application server availability. From an OS/400 perspective there are three high availability database topologies one may choose from. These are:

- 1 HABP (High Availability Business Partner) products that provide transactional replication
- 2 IASP (Independent ASP) usage involving switched disks for local failover
- 3 HABP replication of IASP data to a remote system, which may also include local failover capabilities via the OS/400 clustering support
- 4 System-level replication utilizing the ESS (IBM Enterprise Storage Server) synchronous PPRC function between two ESS servers
- 5 A hybrid solution involving two or more of the aforementioned technologies

Disaster recovery can be critical for an enterprise. Disaster recovery here is assured when two active copies of data always exist. Data replication assures two active copies of data. Taking this into account, all but option 2 provide for disaster recovery protection.

In all of the recommended topologies, the relationship between the WebSphere application server and the HA DB scheme is the same. The WebSphere connection manager semantics hide the HA DB implementation from the WebSphere application. This is pointed out in the Figure 15.2. The application does have a responsibility of handling the exceptions which may be raised by the connection manager when the database is not available. How to handle these exceptions is explained in “Application Considerations for StaleConnectionException” on page 192.

Figure 15.2 WebSphere and high available database access



Recommended Topologies (Data Center)

Three topologies will be discussed here. In later chapters detailed implementation instructions will follow. The three topologies are as follows:

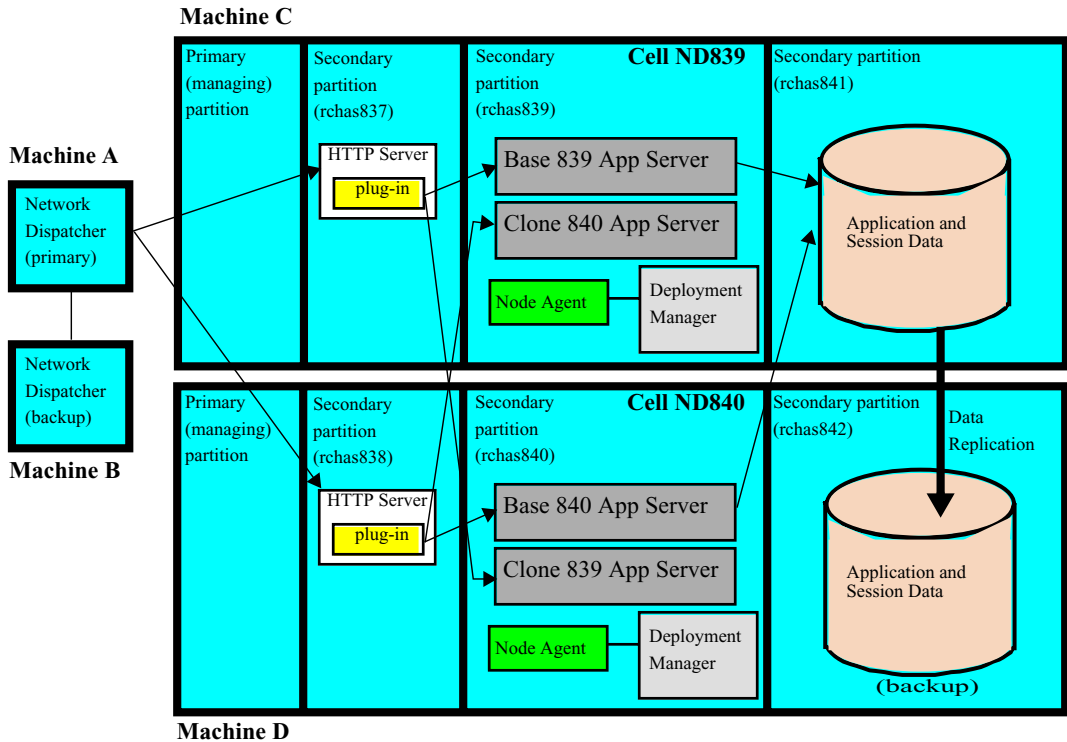
- Peer Cells using Network Dispatcher
- Primary/Backup Cells using Network Dispatcher
- Primary/Backup Cells using HA Apache

The topologies are local to the data center, and does not cover disasters (see “Recommended Topologies (Disaster Recovery)” on page 240 for more information on disaster recovery).

Peer Cells using Network Dispatcher

Figure 15.3 on page 236 below will be carried forward to following chapters where step by step configuration will be illustrated.

Figure 15.3 WebSphere Application Server dual cell with Network Dispatcher



The physical topology and middleware used on each machine and partition is described below.

Machine A and Machine B

WebSphere Network Dispatcher with the hot backup on Machine B. The Network dispatcher “high availability” option assures the network dispatcher is not a single point of failure. Machines A and B could be:

- iSeries Linux partition
- AIX (pSeries), in near future can be AIX in iSeries partition
- Sun Solaris
- Windows 2000

The forwarding method for the dispatcher could be MAC based or Kernel Content Based Routing (cbr). Depending on the forwarding method used the server affinity method could be “stickyness” to IP or passive cookie. The network dispatcher provides

clustering of HTTP web servers which allows unlimited scalability and failover. We recommend the use of a custom advisor to test Web server reach ability for both SSL and non-SSL traffic. The advisor should also check the health of the associated WebSphere application servers. Please see the Chapter 9, “Load balancer Considerations for High Availability” on page 137 for more detail.

Machine C

Machine C is a partitioned iSeries with a thin managing primary partition. Remember if the primary partition terminates then the secondary partitions will follow. Run a limited set of function in the managing partition.

Secondary partition rchas837 houses one or more Apache Web servers. External firewalls can be used to implement a demilitarized zone. The Apache Web server points to the plug-in generated from Network Deployment Cell ND839. WebSphere workload management code will route client requests to cell cluster members Base839 and Clone839. The workload management plug-in code will also provide server affinity and failover for client requests.

Secondary partition rchas839 houses the ND839 cell directory and base cluster member Base839 and Clone840. The cell Network deployment manager provides administration of the cell, associated cluster members and all resources in support of the deployed applications. The cell contains one horizontal cluster containing two cluster members (Base839 and Clone839). Clone839 actually resides on Machine D in secondary partition rchas840. Database access is via JDBC provider “Java Toolbox for iSeries”. The exact same application has been deployed into the clusters for cells ND839 and ND840. This allows the network dispatcher to failover to another HTTP server and in essence cell if need be.

Secondary partition rchas841 houses the application and the session (shopping cart) data. The database HA mechanism here is data replication using a HABP solution along with OS/400 clustering. All data is replicated to Machine D secondary partition rchas842. Persistent database sessions are used. Any one of the three HA DB mechanisms described above could be used on this tier. Keep in mind that the middleware configuration in partitions rchas837 and rchas839 stays the same ill regardless of what HA DB scheme is used.

Machine D

Machine D is also a partitioned iSeries with a thin managing primary partition.

Secondary partition rchas838 houses one or more Apache Web servers. The Apache Web server points to the plug-in generated from Network Deployment Cell ND840. WebSphere workload management code will route client requests to cell cluster members Base840 and Clone840. Again Web Sphere workload management plug-in code provides server affinity and failover for client requests.

Secondary partition rchas840 houses the ND840 cell directory and base cluster member Base840 and Clone839. The cell Network deployment manager provides administration of the cell, associated cluster members and all resources in support of the deployed applications. The cell contains one horizontal cluster containing two cluster members (Base840 and Clone840). Clone840 actually resides on Machine D in secondary partition rchas839. Database access is via JDBC provider “Java Toolbox for iSeries”. The exact same application has been deployed into the clusters for cells ND839 and ND840. This allows the network dispatcher to failover to another HTTP server and in essence cell if need be.

Secondary partition rchas842 houses the hot backup application and the session (shopping cart) data. The redirection mechanism here for failover is IP takeover managed by OS/400 clustering. The primary/backup roles between rchas841 and rchas842 can be manually or automatically switched.

In order to eliminate a single point of failure the configuration above incorporates two of everything. The configuration is by no means limited to two; the number could be whatever is necessary. Typically on OS/400 multiple application servers as not needed for scalability. Just a single application server (JVM) scales quite well whether running on a 4 or 32 way system. Here horizontal cluster members have been included not so much for normal runtime operations, but to enable one cell to be taken offline and not affect the CPU capacity available to run the enterprise. We recommend one configures the cluster members but only activates one member per cell. This avoids needlessly consuming additional system resource with 4 members when 2 members would do fine. Before taking a cell offline, start a horizontal cluster member to take full advantage of available CPU and memory.

The dual cell topology really provide both hardware and software isolation from a WebSphere perspective. This can be very beneficial when:

- Performing Planned Maintenance
- Deploying New versions of WebSphere
- Applying a fix and testing it before going production
- Rolling out a new application or a revision of an existing application.

One cell can be taken offline to perform any of the functions mentioned above and your enterprise can continue to run. If for whatever reasons a cell directory becomes corrupted you always have a hot backup.

Multiple cells will require more administration for day-to-day operations, which can be mitigated through the use of scripts (wsadmin). Multiple cells also require multiple cell repositories, requiring multiple backup of data. It is best to always backup directories.

Primary/Backup Cells using Network Dispatcher

This configuration is just a variation of the configuration described by “Peer Cells using Network Dispatcher” on page 236. With Primary/Backup only the primary cell is ever active. The backup cell is used for:

- Performing planned maintenance
- Deploying new versions of WebSphere
- Applying a fix and testing it before going production
- Rolling out a new application or a revision of an existing application

The day-to-day maintenance is somewhat lessened due to the fact only one cell needs to be administered. On the other hand the backup cell should not get too far behind should it ever be needed for production. With this configuration the WebSphere horizontal clustering is a must for redundancy and failover. All the Apache Web servers would point to the plug-in for the primary cell. It is recommended that you create a test database and test data source for testing purposes. This will allow for more extensive pre-production testing.

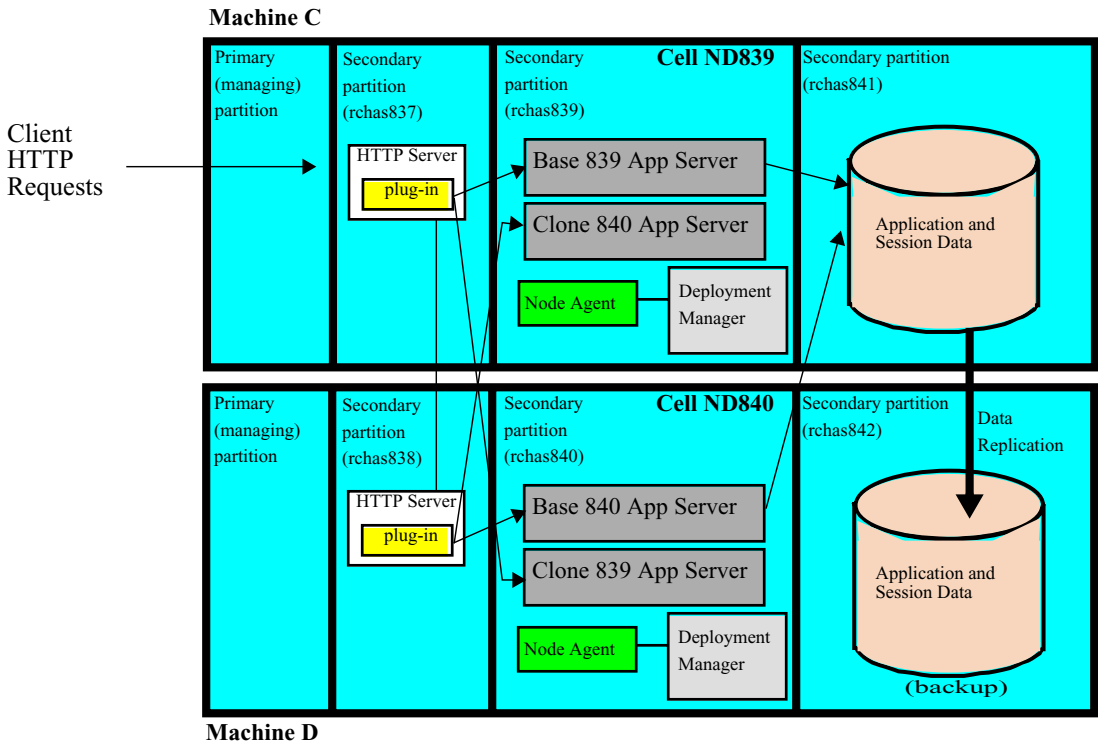
Primary/Backup Cells using HA Apache

This configuration is just a variation of the configuration described above in “Primary/Backup Cells using Network Dispatcher”. The only difference here is that the Apache HA option is used in place of the Network Dispatcher. This configuration is for a small to medium size company who wants a total OS/400 solution.

In Figure 15.4 on page 239 an Apache Web server instance runs in Machine C partition rchas837 and another in Machine D partition rchas840. Only the Apache instance in partition rchas837 is active and receiving client requests. The instance in rchas838 is a hot backup ready to take over should anything happen to rchas837.

This configuration uses the built-in HA capabilities of the OS/400 Apache Web server. Apache in turn uses OS/400 clustering to do both planned and unplanned role reversal. With this configuration the WebSphere horizontal clustering is a must for redundancy and failover. All the Apache Web servers would point to the plug-in for the primary cell.

Figure 15.4 WebSphere Application Server dual cell with HA HTTP Server



Recommended Topologies (Disaster Recovery)

The previous sections dealt with unplanned outages as a result of some sort of network or hardware failure. However, there are other categories of unplanned outage that IT administrators must take into consideration when designing fault tolerant e-business environments. More attention must be paid to disaster recovery (DR) strategies, where disaster recovery refers to how an e-business environment recovers from catastrophic site failures.

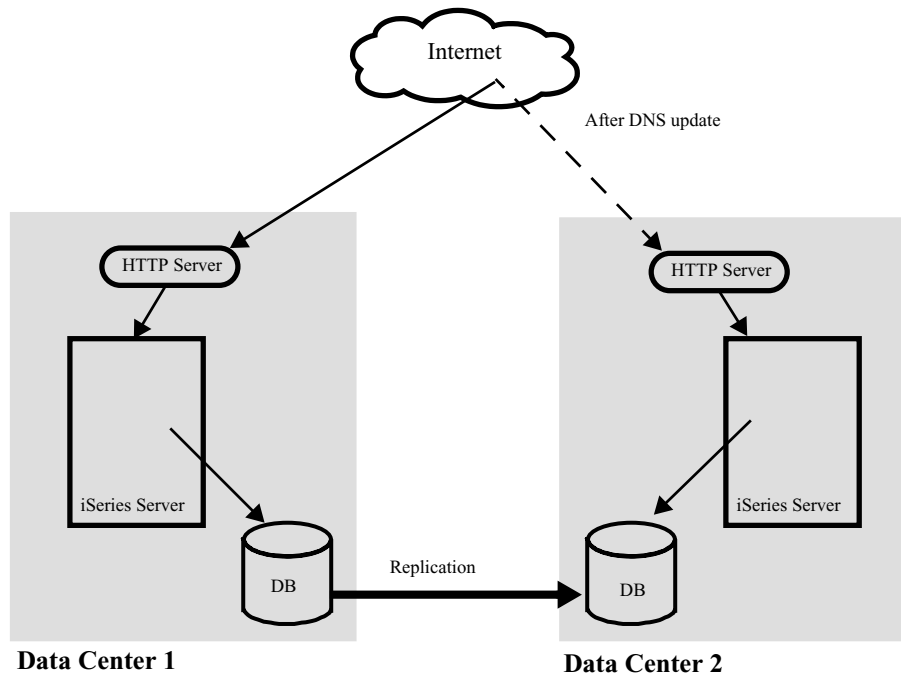
This section will go over some common approaches that one can take to safeguard their e-business environment from physical interruptions to a data center.

The topologies shown below build on the topologies described above for high availability. In addition, we chose to show topologies that use mirrored sites. Mirrored sites are fully redundant facilities with full, real-time information mirroring. Mirrored sites are identical to the primary site in all technical respects. These sites provide the highest degree of availability because the data is processed and stored at the primary and alternate site simultaneously.

Data Centers with Single Server

Figure 15.5 on page 240 depicts two data centers that have a single iSeries server in each of the data centers, and where replication is being used to copy the data from the active data center to the backup data center.

Figure 15.5 Disaster Recovery topology, single server

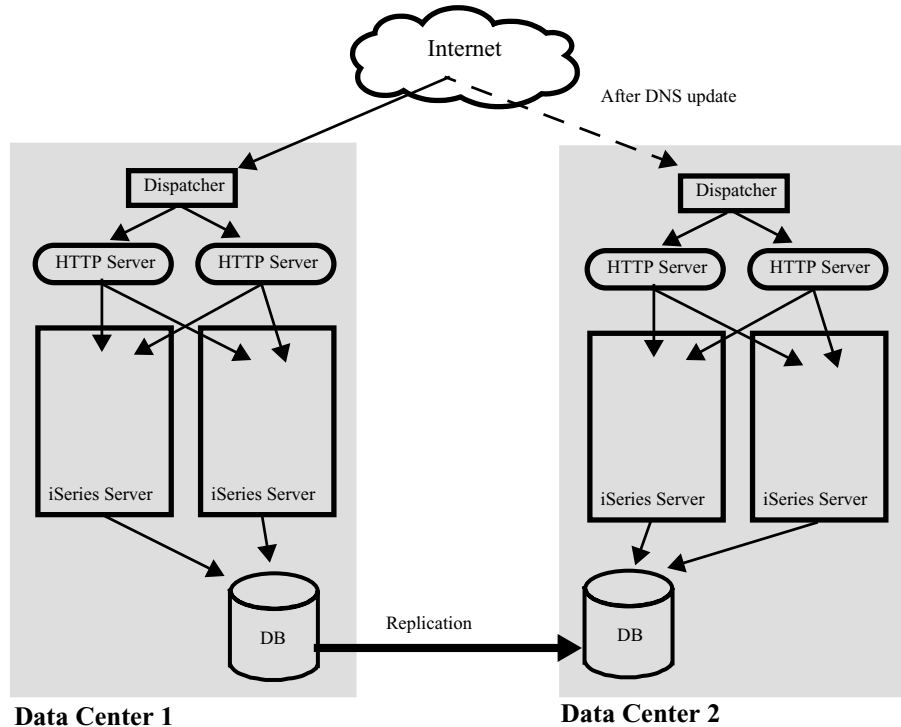


Requests are routed to Data Center 1. If some catastrophe befalls Data Center 1, requests are re-routed to Data Center 2. The re-routing only occurs after a DNS update has been done. The time a DNS update takes to complete may range from a few minutes to several hours.

Data Centers with Multiple Servers

Figure 15.6 on page 241 depicts two data centers that have two iSeries servers in each of the data centers, a dispatcher in each data center routing requests to multiple HTTP servers, and where replication is being used to copy the data from the active data center to the backup data center.

Figure 15.6 Disaster Recovery topology, multiple servers



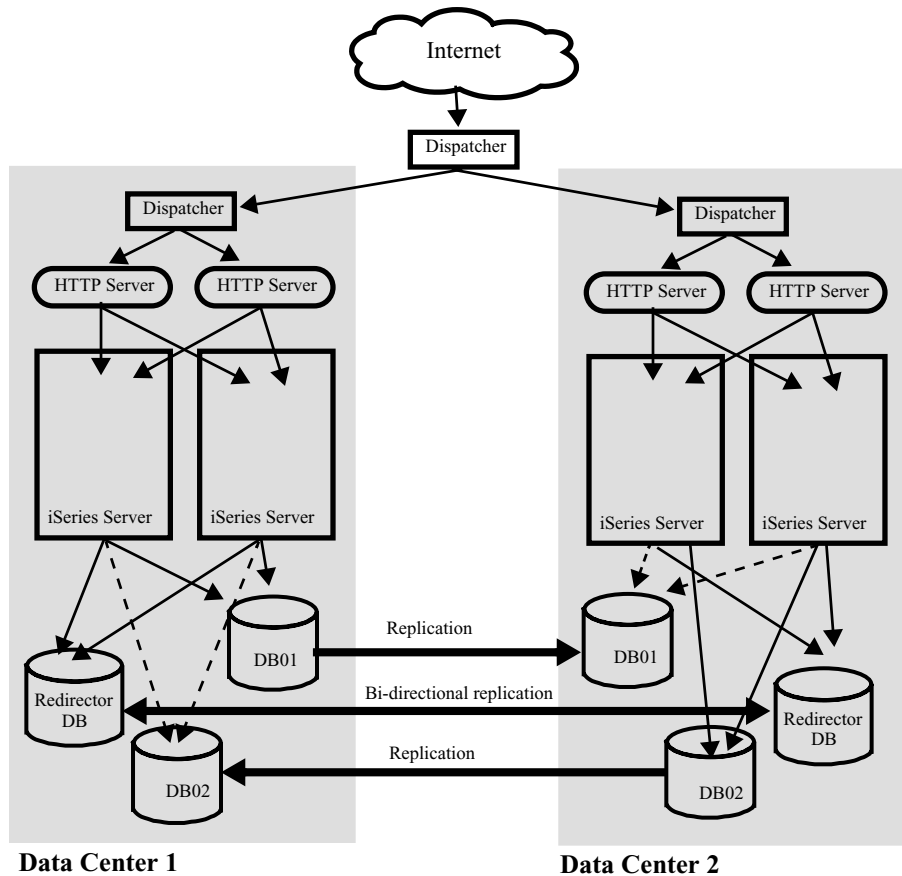
Requests are routed to Data Center 1. If some catastrophe befalls Data Center 1, requests are re-routed to Data Center 2. The re-routing only occurs after a DNS update has been done. The time a DNS update takes to complete may be in the range of a few minutes to several hours.

In this topology we are protected from a server failing due to having multiple servers in a data center. If a server fails, the other server is still functioning and a DNS update does not have to be done.

Data Centers with Multiple Servers and Intelligent Application Routing

Figure 15.7 on page 242 depicts two data centers that have two iSeries servers in each of the data centers, and where a top level dispatcher is being used to route requests to both data centers (i.e. both data centers are active). There are three databases in each data center. The DB01 database is used for clients that have their applications run in data center 1. The DB02 database is used for clients that have their applications run in data center 2. The Redirector database contains a mapping of client to data center. More on this later. All databases are being replicated.

Figure 15.7 Disaster Recovery topology, multiple servers, intelligent routing



The initial request from a client can be routed to either data center. The application handling the initial request (we will call this the sign-on application) must decide where the client application resides and redirect the client to the proper data center. The sign-on application does this by looking up the client in the Redirector database and then sending an HTTP redirect back to the client. The browser will automatically take the HTTP redirect URL and send the request to the specified data center. All subsequent

requests from the client will be handled by the data center (i.e. the top level dispatcher is by-passed).

Now assume that a catastrophe befalls Data Center 1. All requests are routed to Data Center 2. Note that no DNS update needs to be done. However, in order for clients that were serviced by Data Center 1 to be serviced by Data Center 2, a database update to the Redirector DB needs to be done to change the mapping for Data Center 1 clients to Data Center 2. This update is almost instantaneous (unlike a DNS update, where the time may range from minutes to hours). Clients that were running in Data Center 1 can now be serviced by Data Center 2.

CONFIGURING THE LOAD BALANCER

Edge components (formerly Edge Server) are now a part of the WebSphere Application Server offering. Edge components can be used in conjunction with WebSphere Application Server to control client access to Web servers and to enable business enterprises to provide better service to users who access Web-based content over the Internet or a corporate intranet. Using Edge components can reduce Web server congestion, increase content availability, and improve Web server performance. As the name indicates, Edge components usually run on machines that are close (in a network configuration sense) to the boundary between an enterprise's intranet and the Internet.

This section will go into how one would configure the Dispatcher for MAC-based forwarding and the Dispatcher cbr forwarding method.

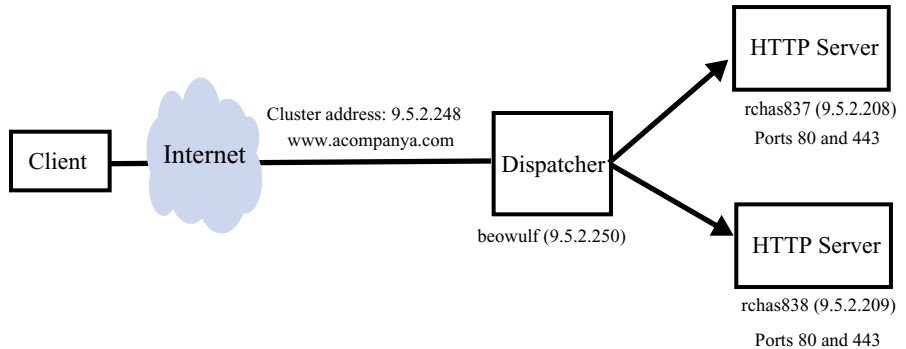
NOTE There are multiple methods of configuring the Dispatcher: command line, scripts, graphical user interface (GUI), and configuration wizard. Whichever method is chosen, the final configuration is stored in a configuration file. The file itself consists of a set of commands, when executed, create the desired configuration. When the configuration file is loaded into the GUI interface that is precisely what happens. We chose to use scripts to configure the Dispatcher.

Configuring Dispatcher for MAC-Based Forwarding Method

Using Dispatcher's MAC forwarding method (the default forwarding method), the Dispatcher load balances the incoming request to the selected server and the server returns the response directly to the client without any involvement of the Dispatcher. With this forwarding method, Dispatcher only looks at the inbound client-to-server flows. It does not need to see the outbound server-to-client flows. This significantly reduces its impact on the application and can result in improved network performance.

Figure 16.1 on page 246 shows a physical representation of the site using an Ethernet network configuration. The Dispatcher machine can be installed without making any physical changes to the network. After a client request is directed to the optimal server by the Dispatcher, the response is then sent directly from server to client with no involvement by the Dispatcher when using MAC forwarding method.

Figure 16.1 A simple Dispatcher configuration topology - MAC-based forwarding



The configuration that will be shown includes MAC forwarding method, “stickyness” to IP to assure Web server affinity, and custom advisor. The affinity method will assure that both non-SSL and SSL client requests for a particular client are sent to the same Web server, while the custom advisor is used to monitor the health of HTTP server and associated WebSphere application servers.

The steps to be taken to configure the Dispatcher are listed below:

- 1 Prepare the servers.
- 2 Create the Dispatcher configuration script.
- 3 Load Configuration script into Dispatcher.

Each of these steps are discussed below.

Step 1. Prepare the Servers

You will need three servers and four IP addresses. One server will be used as the Dispatcher (this can be a linux partition in an iSeries server); the other two servers (in our case these will be partitions on separate iSeries servers) will be used as Web servers. Each Web server requires one IP address. The Dispatcher server requires two addresses: the non-forwarding address (NFA - address of server used for administrative purposes such as remote configuration), and the cluster address (the address which will be load balanced) that you provide to clients to access your Web site.

Once you have the servers available, you will need to perform the following steps:

- 1 For this locally attached configuration example, set up your servers so that they are on the same LAN segment. Ensure that network traffic between the three machines does not have to pass through any routers or bridges.

- 2 Configure the network adapters of the three servers. For this example, we use the following network configuration:

Table 16.1 Server and IP address mappings

Server	Name	IP Address
beowulf	beowulf.acompanya.com	9.5.2.250
rchas837	rchas837.acompanya.com	9.5.2.208
rchas838	rchas838.acompanya.com	9.5.2.209
Subnet Mask: 255.255.255.0		

- 3 Ensure that beowulf.acompanya.com can ping both rchas837.acompanya.com and rchas838.acompanya.com.
- 4 Ensure that rchas837.acompanya.com and rchas838.acompanya.com can ping beowulf.acompanya.com.
- 5 Ensure that content is identical on the two Web servers (rchas837 and rchas838). For example, plug-in XML file have been deployed to the Web server systems so as to be used by the plug-in, application servers are operational and applications installed.
- 6 Ensure that Web servers on rchas837.acompanya.com and rchas838.acompanya.com are operational. Use a Web browser to request pages directly from http://rchas837.acompanya.com and http://rchas838.acompanya.com.
- 7 Obtain another valid IP address for this LAN segment. This is the address you will provide to clients who wish to access your site. For this example we use www.acompanya.com with an IP address of 9.5.2.248. This address is referred to as the cluster address.
- 8 Configure the two Web server systems to accept traffic for www.acompanya.com by running the following command on each of the Web server systems:


```
ADDTCPIFC INTNETADR('9.5.2.248') LIND(*VIRTUALIP)
SUBNETMASK('255.255.255.0')
```

 where the 9.5.2.248 address is the cluster address that is defined within the network dispatcher configuration.

You have now completed all configuration steps that are required on the two Web server servers.

Step 2. Create Configuration Script

The configuration file for our example is shown below. The Web cluster here encompasses rchas837 and rchas838 servers. Certain lines have been highlighted in bold and shall be discussed in more detail.

```

// Start of configuration
dscontrol set loglevel 5
dscontrol set logsize unlimited
dscontrol executor start

dscontrol cluster add www.acompanya.com address 9.5.2.248 primaryhost 9.5.2.250
dscontrol cluster set 9.5.2.248 stickytime 1800
dscontrol cluster set 9.5.2.248 proportions 49 50 1 0
dscontrol executor configure 9.5.2.248 en0 255.255.255.0

dscontrol port add 9.5.2.248:80 reset no

dscontrol server add 9.5.2.248:80:rchas838 address 9.5.2.209
dscontrol server set 9.5.2.248:80:rchas838 fixedweight y

dscontrol server add 9.5.2.248:80:rchas837 address 9.5.2.208
dscontrol server set 9.5.2.248:80:rchas837 fixedweight y

dscontrol port add 9.5.2.248:443 reset no

dscontrol server add 9.5.2.248:443:rchas838 address 9.5.2.209
dscontrol server set 9.5.2.248:443:rchas838 fixedweight y

dscontrol server add 9.5.2.248:443:rchas837 address 9.5.2.208
dscontrol server set 9.5.2.248:443:rchas837 fixedweight y

dscontrol manager start manager.log 10004

dscontrol advisor start wcinst 80 wcinst_80.log
dscontrol advisor interval wcinst 80 20
dscontrol advisor connecttimeout wcinst 9.5.2.248:80 10
dscontrol advisor receivetimeout wcinst 9.5.2.248:80 10

dscontrol advisor start wcinst 443 wcinst_443.log
dscontrol advisor loglevel wcinst 443 5
dscontrol advisor interval wcinst 443 20
dscontrol advisor connecttimeout wcinst 9.5.2.248:443 10
dscontrol advisor receivetimeout wcinst 9.5.2.248:443 10
// End of configuration

```

What follows is an explanation of the lines highlighted in bold above.

dscontrol cluster add www.acompanya.com address 9.5.2.248 primaryhost 9.5.2.250

This command defines a cluster name and address to which all clients will connect to. The address of the cluster is 9.5.2.248, which should map to the “host” www.acompanya.com. The primaryhost is the network address of the machine hosting the Dispatcher. The client URL would include the cluster IP address (or host name) and port.

dscontrol cluster set 9.5.2.248 stickytime 1800

The default stickytime for ports to be created under this cluster. This may be overridden for individual ports using port stickytime. The default value of stickytime is 0. Remember configuring a cluster or port to be sticky informs the Dispatcher to direct

unique client requests to the same Web server. For MAC based forwarding this applies to both non-SSL and SSL requests. The unit of measure for stickytime is seconds.

dscontrol port add 9.5.2.248:80 reset no

Now we must start to define the servers which shall be load balanced. The command above associates port 80 with the cluster 9.5.2.248.

dscontrol server add 9.5.2.248:80:rhas838 address 9.5.2.209

dscontrol server add 9.5.2.248:80:rhas837 address 9.5.2.208

Now that a port 80 has been added we proceed to add the associated Web servers, rhas838 and rhas837 at 9.5.2.209 and 9.5.2.208, respectively.

dscontrol server set 9.5.2.248:80:rhas838 fixedweight y

The weights are constant and we instruct the Dispatcher Manager function to not alter them. This will give us a round robin effect for new client requests.

dscontrol port add 9.5.2.248:443 reset no

Now define the servers which shall be load balanced for SSL client requests. The command above associates port 443 with the cluster 9.5.2.248.

dscontrol server add 9.5.2.248:443:rhas838 address 9.5.2.209

dscontrol server add 9.5.2.248:443:rhas837 address 9.5.2.208

Now that a port 443 has been added we proceed to add the associated Web servers, rhas838 and rhas837 at 9.5.2.209 and 9.5.2.208, respectively.

dscontrol advisor start wcinst 80 wcinst_80.log

dscontrol advisor interval wcinst 80 20

This example uses a custom advisor named wcinst. The command above starts wcinst for all servers defined on port 80. The interval states that every 20 seconds the advisor will query all the Web servers on port 80. The advisor is called by the Dispatcher. The advisor query will test the health of the Web servers and also the associated WebSphere application servers.

See “Custom Advisors” on page 255 for an example of a custom advisor.

dscontrol advisor connecttimeout wcinst 9.5.2.248:80 10

The Dispatcher establishes a connection to a clustered Web server and then calls the corresponding custom advisor supplying the connection. The command above, instructs the Dispatcher to wait no longer than 10 seconds for connect to complete.

If the connect does not complete in 10 seconds then the Web server will be marked as down. A definition like this is needed for both port 80 and 443.

dscontrol advisor receivetimeout wcinst 9.5.2.248:80 10

The custom advisor will use the connection supplied by the Dispatcher to perform health checks on the Web server. A request will be sent to the Web server and the advisor will wait for a response. The receivetimeout is the amount of time the advisor will wait before it will mark the Web server as down. A definition like this is needed for both port 80 and 443.

dscontrol advisor start wcinst 443 wcinst_443.log

The command above starts custom advisor wcinst for all Web servers defined on port 443.

Step 3. Load Configuration Script into Dispatcher

To load a Dispatcher configuration file, store the created file (we called it `beowulf.cfg`) in the install directory. For unix-type platforms that are supported by the Load Balancer, the directory would be:

```
/opt/ibm/edge/lb/servers/configurations/dispatcher
```

For the windows platform, the directory is:

```
C:\Program Files\IBM\edge\lb\servers\configurations\dispatcher
```

To load the configuration file using the GUI, perform the following steps:

- 1 Ensure `dsserver` is running
 - For AIX, HP-UX, Linux, or Solaris, run the following as root: `dsserver`
 - For Windows, `dsserver` runs as a service that starts automatically
- 2 Next, do one of the following:
 - For AIX, HP-UX, Linux, or Solaris: type `lbadmin`
 - For Windows: click **Start** => **Programs** => **IBM WebSphere** => **Edge Components** => **IBM Load Balancer** => **Load Balancer**
- 3 Right click on the **Dispatcher** and select **Connect to Host**.
- 4 The Dispatcher Login panel will be displayed. Click on OK.
- 5 The Dispatcher Host should be listed under the Dispatcher. Right click on the **Host**: and select **Load New Configuration**.
- 6 The Load new configuration panel should be displayed with a list of configuration scripts to choose from. Select the desired configuration and click on OK.

NOTE You can also load the script via a Dispatcher command.

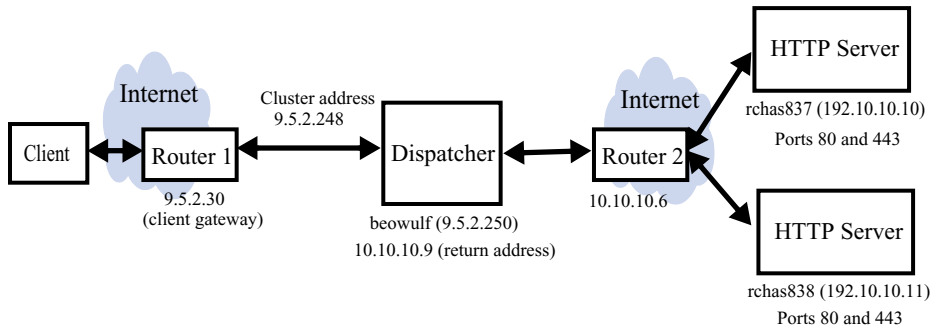
Configuring Dispatcher for cbr Forwarding Method

The Dispatcher component allows you to perform content-based routing for HTTP (using the “content” type rule) and HTTPS (using SSL session ID affinity) without having to use Caching Proxy.

The following example uses cbr forwarding with passive cookie affinity method for non-SSL and SSL session ID affinity for SSL client requests, and custom advisor. The affinity methods will assure both non-SSL and SSL client requests for a particular client are sent to the same Web server, while the custom advisor is used to monitor the health of HTTP server and associated WebSphere application servers.

Figure 16.2 on page 251 shows a physical representation of the site using an Ethernet network configuration. After a client request is directed to the optimal server by the Dispatcher, the response is then sent back to the dispatcher which then forwards the response to the client via the clientgateway.

Figure 16.2 A simple Dispatcher configuration topology - cbr forwarding method



If your subnet does not have a local router (router 1 in Figure 16.2 on page 251), then you must configure a machine to do IP forwarding and use that as the client gateway. Consult your operating system documentation to determine how to enable IP forwarding. You can also get by with not having a backend router (router 2) if the HTTP servers are on the same subnet as the Dispatcher.

The steps to be taken to configure the Dispatcher are listed below:

- 1 Prepare the servers.
- 2 Create the Dispatcher configuration script.
- 3 Load Configuration script into Dispatcher.

Each of these steps are discussed below.

Step 1. Prepare the Servers

You will need three servers and five IP addresses. One server will be used as the Dispatcher (this can be a linux partition in an iSeries server); the other two servers (in our case these will be partitions on separate iSeries servers) will be used as Web servers. Each Web server requires one IP address. The Dispatcher server requires three addresses: the non-forwarding address (NFA - address of server used for administrative purposes such as remote configuration); the cluster address; and the return address used by the Dispatcher when returning replies to the client.

Once you have the servers available, you will need to perform the following steps:

- 1 For this locally attached configuration example, set up your servers so that they are on the same LAN segment. Ensure that network traffic between the three machines does not have to pass through any routers or bridges.

- 2 Configure the network adapters of the three servers. For this example, we use the following network configuration:

Table 16.2 Server and IP address mappings

Server	Name	IP Address
beowulf	beowulf.acompanya.com	9.5.2.250
rchas837	rchas837.acompanya.com	192.10.10.10
rchas838	rchas838.acompanya.com	192.10.10.11
Subnet Mask: 255.255.255.0		

- 3 Ensure that beowulf.acompanya.com can ping both rchas837 and rchas838.
- 4 Ensure that rchas837 and rchas838 can ping beowulf.
- 5 Ensure that content is identical on the two Web servers (rchas837 and rchas838). For example, plug-in XML file have been deployed to the Web server systems so as to be used by the plug-in, application servers are operational and applications installed.
- 6 Ensure that Web servers on rchas837 and rchas838 are operational. Use a Web browser to request pages directly from http://rchas837 and http://rchas838.
- 7 Obtain another valid IP address for this LAN segment. This is the address you will provide to clients who wish to access your site. For this example we use www.acompanya.com with an IP address of 9.5.2.248. This address is referred to as the cluster address.
- 8 Obtain the return IP address. The return address is a unique address or host name that you configure on the Dispatcher machine. Dispatcher uses the return address as its source address when load balancing the client's request to the server. This ensures that the server returns the packet to the Dispatcher machine rather than sending the packet directly to the client.

You have now completed all configuration steps that are required on the two Web server servers.

Step 2. Create Configuration Script

The configuration file for our example is shown below. The Web cluster here encompasses rchas837 and rchas838 servers. Certain lines have been highlighted in bold and shall be discussed in more detail.

NOTE Some lines are shown as two lines in order to fit on the page. Lines that do not start with dscontrol are part of the previous line.

```
// Start of configuration
dscontrol set loglevel 1
dscontrol executor start
dscontrol executor set clientgateway 9.5.2.30

dscontrol cluster add 9.5.2.248 address 9.5.2.248 primaryhost 9.5.2.250
dscontrol cluster set 9.5.2.248 proportions 49 50 1 0
dscontrol executor configure 9.5.2.248 en0 255.255.255.0
```



```

dscontrol port add 9.5.2.248:80 method cbr protocol http reset no
dscontrol port set 9.5.2.248:80 porttype tcp

dscontrol server add 9.5.2.248:80:rchas837 address 192.10.10.10 router 10.10.10.6
returnaddress 10.10.10.9

dscontrol server add 9.5.2.248:80:rchas838 address 192.10.10.11 router 10.10.10.6
returnaddress 10.10.10.9
dscontrol executor configure 10.10.10.9 en0 255.255.255.0
dscontrol server set 9.5.2.248:80: rchas838 weight 9

dscontrol rule add 9.5.2.248:80:always type true priority 1 affinity passivcookie cookienam
JSESSIONID
dscontrol rule useserver 9.5.2.248:80:always rchas837
dscontrol rule useserver 9.5.2.248:80:always rchas838

dscontrol port add 9.5.2.248:443 method cbr protocol ssl reset no
dscontrol port set 9.5.2.248:443 stickytime 1800
dscontrol port set 9.5.2.248:443 porttype tcp

dscontrol server add 9.5.2.248:443:rchas837 address 192.10.10.10 router 10.10.10.6
returnaddress 10.10.10.9

dscontrol server add 9.5.2.248:443:rchas838 address 192.10.10.11 router 10.10.10.6
returnaddress 10.10.10.9

dscontrol rule add 9.5.2.248:443:always type true priority 1
dscontrol rule useserver 9.5.2.248:443:always rchas837
dscontrol rule useserver 9.5.2.248:443:always rchas838

dscontrol manager start manager.log 10004

dscontrol advisor start wcinst 80 wcinst_80.log
dscontrol advisor interval wcinst 80 20
dscontrol advisor connecttimeout wcinst 9.5.2.248:80 10
dscontrol advisor receivetimeout wcinst 9.5.2.248:80 10

dscontrol advisor start wcinst 443 wcinst_443.log
dscontrol advisor connecttimeout wcinst 9.5.2.248:443 10
dscontrol advisor receivetimeout wcinst 9.5.2.248:443 10
dscontrol advisor loglevel wcinst 443 5
dscontrol advisor interval wcinst 443 20
// End of configuration

```

What follows is an explanation of the lines highlighted in bold above.

dscontrol executor set clientgateway 9.5.2.30

Clientgateway is an IP address used for NAT/NAPT or Dispatcher's content-based routing. It is the router address through which traffic in the return direction is forwarded from Dispatcher to clients. Clientgateway must be set to a nonzero value before adding a port with a forwarding method of NAT/NAPT or Dispatcher's content-based routing. If you are unsure of the clientgateway or router 2 address, you can use a traceroute program with the client (or server) address to determine the router address. If your

subnet does not have a local router, you must configure a machine to do IP forwarding and use that as the clientgateway.

dscontrol cluster add 9.5.2.248 address 9.5.2.248 primaryhost 9.5.2.250

This command defines a cluster name and address to which all clients will connect to. In this case both the name and address are 9.5.2.248. The primaryhost is the network address of the machine hosting the Dispatcher.

dscontrol port add 9.5.2.248:80 method cbr protocol http reset no

Start to define the servers which shall be load balanced. In this example we shall load balance both non-SSL and SSL traffic on port 80 and 443. The command above associates port 80 with the cluster 9.5.2.248. Also all traffic on this port shall use the cbr forwarding method. The command also indicates what forwarding method to use. If you want NAT forwarding method, all you would have to do is change this line to the following:

```
dscontrol port add 9.5.2.248:80 method nat protocol http reset no
```

dscontrol server add 9.5.2.248:80: rchas837 address 192.10.10.10 router 10.10.10.6 returnaddress 10.10.10.9

dscontrol server add 9.5.2.248:80: rchas838 address 192.10.10.11 router 10.10.10.6 returnaddress 10.10.10.9

Now that a port has been added we proceed to add the associated servers in this case rchas837 and rchas838. The address of the servers are 192.10.10.10 and 192.10.10.11, respectively. The router address is 10.10.10.6. This allows the Dispatcher and clustered Web servers to reside on different networks. If the servers are on the same subnet as Dispatcher, type the server address as the router address.

Note the “returnaddress 10.10.10.9” is a unique IP address or host name on the Dispatcher machine that Dispatcher uses as its source address when load balancing the client’s request to the server. This ensures that the Web server will return the packet to the Dispatcher machine so that content of the request can be interrogated, rather than sending the packet directly to the client. (Dispatcher will then forward the IP packet on to the client.) You must specify the return address value when the server is added. Return address cannot be changed unless you remove the server and add it again. The return address cannot be the same as the cluster, server, or NFA (non-forwarding address) of the Dispatcher machine. Why is this necessary? Keep in mind we plan on using passive cookie as our affinity mechanism. On the very first request from a client it is possible that a HTTP session and cookie will be obtained at the application server level in which case the cookie is contained within the client HTTP request object. The location is identified by cookie name JSESSIONID. The cookie associated with this server must be obtained by the Dispatcher and put into its affinity record table to assure subsequent requests are routed to the same server. This is why the server response must flow back to the Dispatcher.

dscontrol rule affinityrule add 9.5.2.248:80:always type true priority 1 affinity passivecookie cookie name JSESSIONID

This command defines affinity for the cluster on port 80. In this case the affinity is passive cookie and the cookie name within the client data stream is JSESSIONID. Notice the rule name here is affinityrule. This rule has been created as “always true”. Such a rule will always be selected, unless all the servers associated with it are down. Keep in mind if you define multiple rules each rule can have a priority level.

dscontrol rule affinityrule useserver 9.5.2.248:80:always rchas837

dscontrol rule affinityrule useserver 9.5.2.248:80:always rchas838

The rule affinityrule is simply associated with each server in the cluster. At this point in time the cluster has been defined for non-SSL traffic.

```
dscontrol port add 9.5.2.248:443 method cbr protocol ssl reset no  
dscontrol port set 9.5.2.248:443 stickytime 1800
```

These commands define port 443 to the cluster. The combination of parameters has the effect of enabling SSL session ID affinity using cbr forwarding. There is a stickytime identified of 30 minutes. This means if a affinity is established followed by an inactivity period of 30 minutes, then the Dispatcher is free to pick any server it wishes to.

```
dscontrol server add 9.5.2.248:443:rchas837 address 192.10.10.10 router 10.10.10.6  
returnaddress 10.10.10.9  
dscontrol server add 9.5.2.248:443:rchas838 address 192.10.10.11 router 10.10.10.6  
returnaddress 10.10.10.9
```

Both servers are associated with the cluster. Note here the returnaddress again allows the Dispatcher base function to obtain the SSL session ID.

```
dscontrol advisor start wcinst 80 wcinst_80.log  
dscontrol advisor interval wcinst 80 20
```

This example uses a custom advisor named wcinst. The command above starts wcinst for all servers defined on port 80. The interval states that every 20 seconds the advisor will query all the Web servers on port 80. The query will test the health of the Web servers and also the associated WebSphere application servers.

See “Custom Advisors” on page 255 for an example of a custom advisor.

```
dscontrol advisor connecttimeout wcinst 9.5.2.248:80 10
```

The Dispatcher base function establishes a connection to a clustered Web server and then calls the corresponding custom advisor supplying the connection. The command above, instructs the Dispatcher base function to wait no longer than 10 seconds for connect to complete.

If the connect does not complete in 10 seconds then the Web server will be marked as down. A definition like this is needed for both port 80 and 443.

```
dscontrol advisor receivetimeout wcinst 9.5.2.248:80 10
```

The custom advisor will use the connection supplied by the Dispatcher base function to perform health checks on the Web server. A request will be sent to the Web server and the advisor will wait for a response. The receivetimeout is the amount of time the advisor will wait before it will mark the Web server as down. A definition like this is needed for both port 80 and 443.

```
dscontrol advisor start wcinst 443 wcinst_443.log
```

The command above starts custom advisor wcinst for all Web servers defined on port 443.

Step 3. Load Configuration Script into Dispatcher

Load the Dispatcher configuration file as was done previously for MAC based forwarding method.

Custom Advisors

The custom (customizable) advisor is a small piece of Java code, which you provide as a class file, that gets called by the base code. The base code provides all administrative services, such as starting and stopping an instance of the custom advisor, providing status and reports, and recording history information in a log file. It also reports results

to the manager component. Periodically the base code will perform an advisor cycle, where it individually evaluates all servers in its configuration. It starts by opening a connection with a server machine. If the socket opens, the base code will call the “getLoad” method (function) in the custom advisor. The custom advisor then performs whatever steps are necessary to evaluate the health of the server. Typically, it will send a user-defined message to the server and then wait for a response. (Access to the open socket is provided to the custom advisor.) The base code then closes the socket with the server and reports the load information to the Manager.

In normal mode, the custom advisor exchanges data with the server, and the base advisor code times the exchange and calculates the load value. The base code then reports this load value to the manager. The custom advisor needs only return a zero (on success) or negative one (on error). To specify normal mode, the replace flag in the constructor is set to false. The base code and custom advisor can operate in either normal or replace mode. Choice of the mode of operation is specified in the custom advisor file as a parameter in the constructor method.

In replace mode, the base code does not perform any timing measurements. The custom advisor code performs whatever operations are desired for its unique requirements, and then returns an actual load number. The base code will accept the number and report it to the manager. For best results, normalize your load number between 10 and 1000, with 10 representing a fast server, and 1000 representing a slow server. To specify replace mode, the replace flag in the constructor is set to true.

With this feature, you can write your own advisors that will provide the precise information about servers that you need. A sample custom advisor, ADV_sample.java, is provided with the Load Balancer product. After installing Load Balancer, you may find the sample code in <install directory>/servers/samples/CustomAdvisors install directory.

Custom advisors must be written in the Java language. The following files are referenced during compilation:

- The custom advisor file
- The base classes file, ibmnd.jar, which is found in the <install_path>/servers/lib directory

Your classpath environment variable must point to both the custom advisor file and the base classes file during the compilation. A compile command might have the following format:

```
javac -classpath /opt/ibm/edge/lb/servers/lib/ibmnd.jar ADV_name.java
```

The output of the compilation is a class file, for example, ADV_name.class. Before starting the advisor, copy the class file to the <install_path>/servers/lib/CustomAdvisors/ directory.

NOTE If you add a custom advisor to Dispatcher, or any other applicable Load Balancer component, you must stop and then restart dsserver (or the service for Windows) to enable the Java process to read the new custom advisor class files. The custom advisor class files are loaded only at startup. It is not necessary to stop the executor. The executor continues to run even when dsserver, or the service, has been stopped.

Sample Code for Custom Advisor

The following is a custom advisor we have created for WebSphere Commerce and consider a good example of what an advisor should do. This advisor issues both SSL and Non-SSL health checks.

```
// Start of Custom Advisor
package CustomAdvisors;

import java.io.*;
import java.net.*;
import java.util.*;
import java.util.Date;
import com.ibm.internet.nd.advisors.*;
import com.ibm.internet.nd.common.*;
import com.ibm.internet.nd.manager.*;
import com.ibm.internet.nd.server.SRV_ConfigServer;

//-----
//Define the table element for the hash tables used in this custom advisor
class ADV_nte implements Cloneable {
private String sCluster;
private int iPort;
private String sServer;
private int iLoad;
private Date dTimestamp;

//-----
//constructor
public ADV_nte(String sClusterIn,int iPortIn,String sServerIn,int iLoadIn)
{
    sCluster =sClusterIn;
    iPort =iPortIn;
    sServer =sServerIn;
    iLoad =iLoadIn;
    dTimestamp =new Date();
}

//-----
//check whether this element is current or expired
public boolean isCurrent(ADV_wcinst oThis)
{
    boolean bCurrent;
    int iLifetimeMs =3*1000 *oThis.getInterval();//set lifetime as 3 advisor cycles
    Date dNow =new Date();
    Date dExpires =new Date(dTimestamp.getTime()+iLifetimeMs);
    if (dNow.after(dExpires)){
        bCurrent =false;
    }else {
        bCurrent =true;
    }
    return bCurrent;
}
}
```

```

//-----
//value accessor(s)
public int getLoadValue(){return iLoad;}

//-----
//clone (avoids corruption between threads)
public synchronized Object Clone()
{
try {
return super.clone();
}catch (CloneNotSupportedException e){
return null;
}
}
}

//-----
//define the custom advisor
public class ADV_wcinst extends ADV_Base implements
ADV_MethodInterface,ADV_AdvisorVersionInterface
{
static final int ADV_WCINST_PORT_HTTP =80;
static final int ADV_WCINST_PORT_SSL =443;
//-----
//define tables to hold port-specific history information
static Hashtable htWCINSTHTTP =new Hashtable();
static Hashtable htWCINSTSSL =new Hashtable();
static final String ADV_WCINST_NAME ="wcinst";
static final int ADV_WCINST_DEF_ADV_ON_PORT =80;
static final int ADV_WCINST_DEF_INTERVAL =20;
static final String HTTP_GOOD = "HTTP/1.1 200 OK";
static final String HTTP_GOOD2 = "HTTP/1.1 200 ok";
static final String HTTP_ERROR = "404 Not Found";
static final String SERVLET_KEYWORD = "ServletInfo";
static final String SERVLET_LOCATION = "/webapp/wcs/stores/servlet?DEBUG=1";

//static final String ADV_HTTP_REQUEST_STRING =
//"HEAD /HTTP/1.0 \r \nAccept:*/*\r \nUser-Agent:"+
//"IBM_LB_Custom_Advisor \r \n \r \n";
//-----
//create byte array with SSL client hello message
public static final byte []abClientHello ={
(byte)0x80, (byte)0x1c,
(byte)0x01, //client hello
(byte)0x03, (byte)0x00, //SSL version
(byte)0x00, (byte)0x03, //cipher spec len (bytes)
(byte)0x00, (byte)0x00, //session ID len (bytes)
(byte)0x00, (byte)0x10, //challenge data len (bytes)
(byte)0x00, (byte)0x00, (byte)0x03, //cipher spec
(byte)0x1A, (byte)0xFC, (byte)0xE5, (byte)0x20, //challenge data
(byte)0xFD, (byte)0x3A, (byte)0x3C, (byte)0x18,
(byte)0xAB, (byte)0x67, (byte)0xB0, (byte)0x52,
(byte)0xB1, (byte)0x1D, (byte)0x55, (byte)0x44, (byte)0x0D, (byte)0x0A };

```

```

//-----
//constructor
public ADV_wcinst()
{
    super(ADV_WCINST_NAME,VERSION,ADV_WCINST_DEF_ADV_ON_PORT,
          ADV_WCINST_DEF_INTERVAL,"",
          true);//false =load balancer times the response
    setAdvisor (this );
}

//-----
//ADV_AdvisorInitialize
public void ADV_AdvisorInitialize() { return; }

//-----
//synchronized PUT and GET access routines for the hash tables
synchronized ADV_nte getNte(Hashtable ht,String sName,String sHashKey)
{
    ADV_nte nte =(ADV_nte) (ht.get(sHashKey));
    if (null !=nte){
        nte =(ADV_nte)nte.Clone();
    }
    return nte;
}

synchronized void putNte(Hashtable ht,String sName,String sHashKey,ADV_nte nte)
{
    ht.put(sHashKey,nte);
    return;
}

//-----
//ADV_findStatus() Parse the received response packet and try to find out the
// status of the WebSphere Commerce Application Server
public int ADV_findStatus(String inline) {
    int rc = ADV_HOST_INACCESSIBLE;

    try {
        BufferedReader reader = new BufferedReader(new StringReader(inline));
        String line = reader.readLine();
        while (null != line) {
            if (-1 < line.indexOf(SERVLET_KEYWORD)) {
                // if the SERVLET_KEYWORD is found in line, then Commerce
                // is up and working.
                // the SERVLET_KEYWORD that we check for Commerce is 'ServletInfo'.
                ADVLOG(3, "String content holds String " + SERVLET_KEYWORD + "\n");
                ADVLOG(3, "Line: " + line + "\n");

                rc = 200; //return 1 to ND to signify that Commerce is up

                line = null;
            } // if found NDAdvisor on this line
            // keep reading through the packet lines
        } else {

```

```

        line = reader.readLine();
    }
    } // search through packet for response
} catch (IOException ignore) {
    ADVLOG(1, "ERROR: IOException in ADV_findStatus()\n");
}

return rc;
} // ADV_findStatus()

//-----
//getLoadHTTP -determine HTTP load based on server response
int getLoadHTTP(int iConnectTime,ADV_Thread caller)
{
    int rc = 0;
    int returnValue = ADV_HOST_INACCESSIBLE;
    String getString = "GET " + SERVLET_LOCATION + " HTTP/1.0\r\n" +
        "Connection: Keep-Alive\r\n" +
        "User-Agent: IBM_Network_Dispatcher_wcinst_Advisor\r\n" +
        "Pragma: no-cache\r\n" +
        "Host: " + caller.getCurrentCluster() + "\r\n" +
        "Accept: */*\r\n" +
        "Accept-Encoding: gzip\r\n" +
        "Accept-Language: en\r\n" +
        "Accept-Charset: iso-8859-1,*,utf-8\r\n\r\n";

    rc = caller.send(getString);
    if (0 < rc) { // successfully sent, now get the results
        StringBuffer returnedData = new StringBuffer("");
        rc = caller.receive(returnedData);
        // Look for the error string or the good string
        if (0 < rc && returnedData != null) {
            String line = returnedData.toString();
            if (-1 < line.indexOf(HTTP_ERROR)) {
                ADVLOG(3, "Packet indicates error: " + HTTP_ERROR + "\n");
            } else if (-1 < line.indexOf(HTTP_GOOD)) {

                // We're looking for HTTP/1.1 200 ok in this first packet. If
                // this is here then we should be getting a servlet response
                // That response will either be in this packet or could be in a
                // second TCP packet from the server depending on timing, etc
                ADVLOG(3, "Packet indicates good response\n");
                ADVLOG(4, "Packet was : \n" + line);
                if (-1 == line.indexOf(SERVLET_KEYWORD)) {
                    rc = caller.receive(returnedData);
                    if (0 < rc && returnedData != null) {
                        line = returnedData.toString();
                        if (-1 == line.indexOf(SERVLET_KEYWORD)) {
                            // Servlet keywords not found in second packet
                            ADVLOG(3, "ERROR: Packet did not contain servlet response\n"
                                + "Packet #2 was :\n" + line);
                            line = null;
                        }
                    }
                }
            }
        }
    }
}

```



```

        } // need to check a second packet
        if (null != line) {
            returnValue = ADV_findStatus(line);
        }
    } // if first packet indicates HTTP 200 message
    else {
        ADVLOG(3, "Packet was not HTTP 404 or 200, it was :\n" + line +
            "\nResponse:\n" + returnedData);
    }
} // if received a packet
} // endif caller.send

return returnValue;
}
//-----
//getLoadSSL()-determine SSL load based on server response
int getLoadSSL(int iConnectTime,ADV_Thread caller)
{
    int iLoad =ADV_HOST_INACCESSIBLE;
    int iSocket =caller.getAdvisorSocket();//send hex request to server
    CMNByteArrayWrapper cbawClientHello =new CMNByteArrayWrapper(abClientHello);
    int iRc =SRV_ConfigServer.socketapi.sendBytes(iSocket,cbawClientHello);
    if (0 <=iRc){//did the request return a failure?
        StringBuffer sbReceiveData =new StringBuffer("");//buffer for response
        iRc =caller.receive(sbReceiveData);//get a response from the server
        if (0 <=iRc){//did the receive return a failure?
            if (0 <sbReceiveData.length()){//is data there?
                iLoad = 200;//ignore retrieved data and return success code
            }
        }
    }
    return iLoad;
}

//-----
//getLoad -merge results from the HTTP and SSL methods
public int getLoad(int iConnectTime,ADV_Thread caller)
{
    int iLoadHTTP;
    int iLoadSSL;
    int iLoad;
    int iRc;
    String sCluster =caller.getCurrentCluster();//current cluster address
    int iPort =getAdviseOnPort();
    String sServer =caller.getCurrentServer();
    String sHashKey =sCluster +":"+sServer;//hash table key
    if (ADV_WCINST_PORT_HTTP ==iPort){//handle an HTTP server
        iLoadHTTP =getLoadHTTP(iConnectTime,caller);//get the load for HTTP
        ADV_nte nteHTTP =new ADV_nte(sCluster,iPort,sServer,iLoadHTTP);
        putNte(htWCINSTHTTP,"HTTP",sHashKey,nteHTTP);//save HTTP load information
        ADV_nte nteSSL =getNte(htWCINSTSSL,"SSL",sHashKey);//get SSL information
        if (null !=nteSSL){
            if (true ==nteSSL.isCurrent(this)){//check the time stamp
                if (ADV_HOST_INACCESSIBLE !=nteSSL.getLoadValue()){//is SSL working?

```

```

        iLoad =iLoadHTTP;
    }else {//SSL is not working,so mark the HTTP server down
        iLoad=ADV_HOST_INACCESSIBLE;
    }
}else {//SSL information is expired,so mark the

    //HTTP server down
    iLoad =ADV_HOST_INACCESSIBLE;
}
}else {//no load information about SSL,report
    //getLoadHTTP()results
    iLoad =iLoadHTTP;
}
}
else if (ADV_WCINST_PORT_SSL ==iPort){//handle an SSL server
    iLoadSSL =getLoadSSL(iConnectTime,caller);//get load for SSL
    ADV_nte nteSSL =new ADV_nte(sCluster,iPort,sServer,iLoadSSL);
    putNte(htWCINSTSSL, "SSL",sHashKey,nteSSL);//save SSL load info.
    ADV_nte nteHTTP =getNte(htWCINSTHTTP, "HTTP",sHashKey);//get HTTP
    //information
    if (null !=nteHTTP){
        if (true ==nteHTTP.isCurrent(this)){//check the timestamp
            if (ADV_HOST_INACCESSIBLE !=nteHTTP.getLoadValue()){//is HTTP working?
                iLoad =iLoadSSL;
            }else {//HTTP server is not working,so mark SSL down
                iLoad =ADV_HOST_INACCESSIBLE;
            }
        }else {//expired information from HTTP,so mark SSL down
            iLoad =ADV_HOST_INACCESSIBLE;
        }
    }else {//no load information about HTTP,report
        //getLoadSSL()results
        iLoad =iLoadSSL;
    }
}
}
//-----
//error handler
else {
    iLoad =ADV_HOST_INACCESSIBLE;
}
return iLoad;
}
}
// End of Custom Advisor

```

What follows is an explanation of the lines highlighted in bold above.

```
static final String ADV_WCINST_NAME ="wcinst";
```

Custom Advisor name is wcinst.

```
static Hashtable htWCINSTHTTP =new Hashtable(); // track server/port 80
heath checks
```

```
static Hashtable htWCINSTSSL =new Hashtable(); // track server/port 443  
health checks
```

Keep in mind each of the clustered Web servers can receive client requests on port 80 or 443. The custom advisor in turn will perform health checks for these two ports. If either port is not operational then the custom advisor must inform the Dispatcher base function that the server should be considered unavailable. The nature of the calls from base Dispatcher to the custom advisor follow this type of pattern:

```
Base dispatcher code establishes connection to Web server "wc55host1" port 80  
Call Custom Advisor passing connection  
Interpret results from Custom Advisor (may mark server "wc55host1 down)
```

```
Base dispatcher code establishes connection to Web server "wc55host1" port 443  
Call Custom Advisor passing connection  
Interpret results from Custom Advisor (may mark server "wc55host1 down)
```

```
Base dispatcher code establishes connection to Web server "wc55host2" port 80  
Call Custom Advisor passing connection  
Interpret results from Custom Advisor (may mark server "wc55host2 down)
```

```
Base dispatcher code establishes connection to Web server "wc55host2" port 443  
Call Custom Advisor passing connection  
Interpret results from Custom Advisor (may mark server "wc55host2 down)
```

Repeat pattern ...

Given a server, in between advisor calls there must be a method for the custom advisor to track whether a preceding call to port 80 or 443 was successful or not. The method to keep track of this is a static hash table for server and ports 80 and 443.

```
public int getLoad(int iConnectTime,ADV_Thread caller)
```

This is the routine within the Custom Advisor that the base dispatcher code calls. Remember that the base Dispatcher code has established a connection to one of the clustered Web servers on either port 80 or 443. The indication of server and port are also passed to the custom advisor. The logic within the `getLoad()` routine is fairly straight forward. The following pseudo code presents a simple logic flow.

```
Record server and port on which to advise on  
Use Cluster and server to build a hash key  
If (port == 80)  
    getLoadHTTP() - function sends health check to port 80  
    Record (Result of getLoadHTTP in the static hash table for server/port 80)  
    Get last health check for server port 443 from SSL hash table  
    If (SSL timestamp not expired and last SSL health check ok)  
        Return result for getLoadHTTP()  
    Else  
        Return host not accessible  
Else If (port == 443)  
    getLoadSSL() - function sends health check to port 443  
    Record (Result of getLoadSSL in the static hash table for server/port 443)  
    Get last health check for server/port 80 from server/port 80 hash table  
    If (port80 timestamp not expired and last port 80 health check ok)  
        Return result for getLoadSSL()  
    Else  
        Return host not accessible
```

End

So what is the timestamp check all about? This is simply a check to make sure that the information contained within a hash table is not stale. Stale here is defined as three advisor cycles which would be 60 seconds or $2 * 20$. This should not happen. For more information on this see the `isCurrent()` function above.

`int getLoadHTTP(int iConnectTime,ADV_Thread caller)`

This routine is called to perform health checks on port 80. Here is where you can customize your check as you wish. In this example a servlet request is sent to the external HTTP server port 80 and the response is checked. This tests not only the Web server but also the corresponding application server. Once a request is sent the thread of execution will only wait “receivetimeout” number of seconds for a response. The “receivetimeout” was part of the configuration file described above.

`int getLoadSSL(int iConnectTime,ADV_Thread caller)`

This routine is called to perform health checks on port 443. Again you may customize as you wish. In this example a ClientHello is sent to the Web server port 443. We assume this should be adequate since the port 80 check handles the application server’s health status. Again, once a request is sent the thread of execution will only wait “receivetimeout” number of seconds for a response. The “receivetimeout” was part of the configuration file described above.

CHAPTER 17

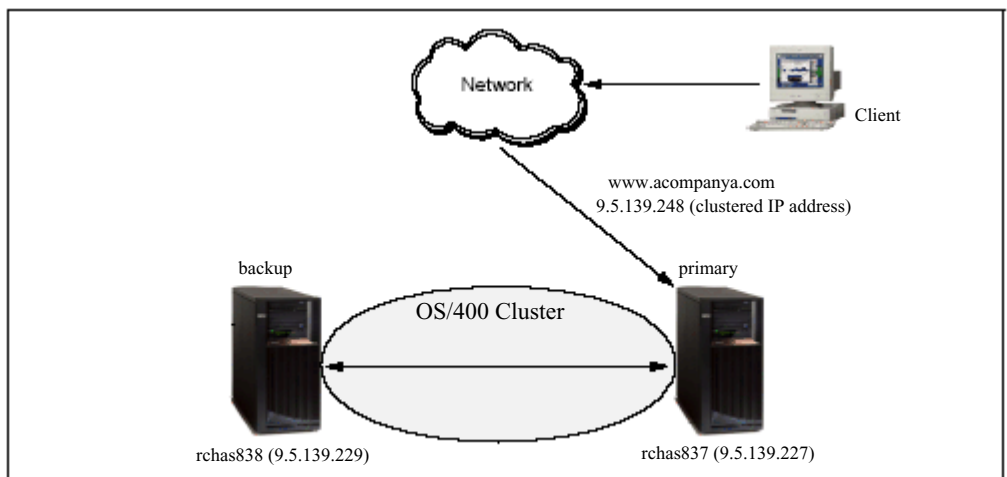
CONFIGURING HTTP SERVER

This chapter will show you how to configure a highly available HTTP server.

Highly Available HTTP Server Using Primary/Backup Model

The section will discuss the steps that are necessary to get a simple, highly available Web server running between two iSeries servers. On each server is an identical, yet separate, instance of the HTTP Server (powered by Apache), clustered together using iSeries clusters, where one server is the primary server and the second server is the backup server. The network environment is illustrated by Figure 17.1 on page 265.

Figure 17.1 Primary/backup with IP takeover



To configure the primary and backup servers, we need to perform the following tasks as explained in the sections that follow:

- 1 Prepare the iSeries servers
- 2 Create the cluster
- 3 Update the HTTP configuration files

TIP In addition the steps outlined in this section, you must review the Information Center's resources for clustering: <http://publib.boulder.ibm.com/html/as400/infocenter.html>. On this site, select your version and language, and click GO! Search for "clusters". The document that helps you the most in this step is entitled "Cluster configuration checklist".

Step 1. Prepare the iSeries Servers

Before you start, you need to validate and possibly modify some of the system and TCP/IP configuration settings on both iSeries servers. These instructions demonstrate what is needed on both systems by showing you what we did on rchas837. In your own environment, you must perform these steps for both the primary and backup servers.

- 1 Ensure that clustering is enabled for both iSeries servers by issuing the Change Network Attributes (CHGNETA) command on both systems:

```
CHGNETA ALWADDCLU (*ANY)
```

- 2 Create the same routable IP address on both iSeries servers to be used as the clustered IP address. This IP address is the "well-known" IP address at which your Web application is bound to via the Listen directive. That is, a Domain Name System (DNS) server must have an A record that maps the host name of `www.acompanya.com` (what the client would be as the host) to 9.5.139.248.

NOTE It may feel strange to create the same routable IP address on both iSeries. But, as long as both are not active at the same time this is OK. It is the IP address takeover feature of OS/400's HA clustering that automatically allows only one of the iSeries servers to have 9.5.139.248 active at one time. To be clear, you never manually make 9.5.139.248 active on either iSeries server. HA clustering's IP takeover does this for you.

- 3 Make sure that the loopback IP address 127.0.0.1 is configured and started on both iSeries servers.
- 4 Make sure that the Internet Daemon (INETD) server is started on both iSeries servers by issuing the Start TCP/IP Server (STRTCPSVR) command on both systems:

```
STRTCPSVR *INETD
```

If it is already started, you will get an error message.

TIP Because INETD is needed for the proper operation of HA clustering on your iSeries servers, we recommend that you change the properties of the INETD to Start when TCP/IP is started.

- 5 Make sure that HA processes used by OS/400 clustering can run unimpeded in the QBATCH subsystem on both iSeries servers. OS/400's HA clustering uses a batch job to slow poll the primary server to determine if this system is still available to the network. By default, this batch job is started in QBATCH. You must ensure that the job queue for QBATCH is large enough to always allow this job to run. You can use

the Change Job Queue Entry (CHGJOBQE) command to change the maximum active to no maximum:

```
CHGJOBQE SBSD(QBATCH) JOBQ(QBATCH) MAXACT(*NOMAX)
```

6 Verify that the basic TCP/IP configuration is correct.

Test the interconnectivity of all the servers and clients to ensure that the Step 2 has the best chance of succeeding.

- a Verify that primary rchas837 can verify TCP/IP Connection (PING) backup rchas838 at IP address 9.5.139.229.
- b Verify that backup rchas838 can PING primary rchas837 at IP address 9.5.139.227.
- c Conversely, make sure that a PING to clustered IP address 9.5.139.248 times out. That is, this IP address should not be active on any host in the subnetwork.
- d Verify that when the client PINGs the primary www.ahosta.com, the name is resolved to 9.5.139.248. However, this PING should time out.

Step 2. Create the Cluster

You can perform the following steps on either iSeries server rchas837 or rchas838. We created the HA cluster HATEST from rchas837.

NOTE Although we create the cluster using CL commands, you can also create the cluster by using the iSeries Navigator.

TIP The 5250 command GO CMDCLU provides a list of HA clustering commands that are available.

- 1** On iSeries server rchas837, use the Create Cluster (CRTCLU) command to create the cluster HATEST that includes node RCHAS837.

```
CRTCLU CLUSTER(HATEST) NODE((RCHAS837 ('9.5.139.227')))
```

The CRTCLU command should complete without errors.

- 2** Create the second cluster node RCHAS838 by entering the Add Cluster Node Entry (ADDCLUNODE) command (on iSeries server RCHAS837):

```
ADDCLUNODE CLUSTER(HATEST) NODE(RCHAS838 ('9.5.139.229'))
```

The ADDCLUNODE command should complete without errors.

- 3** To see the status of nodes RCHAS838 and RCHAS837 in cluster HATEST, enter the Display Cluster Information (DSPCLUINF) command:

```
DSPCLUINF CLUSTER(HATEST)
```

As shown in Figure 17.2 on page 268, nodes RCHAS838 and RCHAS837 have been added to cluster HATEST.

Figure 17.2 Display Cluster Information with two nodes

```

Display Cluster Information

Cluster . . . . . : HATEST
Consistent information in cluster . . . . . : *YES
Current cluster version . . . . . : 3
Current cluster modification level . . . . . : 0
Configuration tuning level . . . . . : *NORMAL
Number of cluster nodes . . . . . : 2
Detail . . . . . : *BASIC

Cluster Membership List

Potential
Node Mod Device
Vers Level Domain
-----Interface Address
Node      Status
RCHAS837  Active      3      0  *NONE      9.5.139.227
RCHAS838  Active      3      0  *NONE      9.5.139.229

```

Step 3. Update the HTTP Configuration Files

Now that you established a cluster between the two iSeries servers RCHAS837 and RCHAS838, it is time to add to this cluster a Cluster Resource Group (CRG) that is used by the HTTP Server (powered by Apache) to enable it to perform switchovers/failovers to the backup server when it detects that the primary server is down or not responding. You do this by creating a basic HTTP Server configuration on iSeries server RCHAS837 and then adding the necessary HA server directives. Then you make an identical copy of this HTTP Server configuration and Web site on iSeries server RCHAS838.

- 1 Using the Administration GUI (Admin GUI) create on iSeries server rchas837 an HTTP Server (powered by Apache) server with the attributes shown in Table 17.1.

Table 17.1 Create New HTTP Server wizard required parameters

Create HTTP Server wizard parameter	Value
Server name	HAHTTPSVR ¹
Server root	/www/hahttpsvr
Document root	/www/hahttpsvr/htdocs
On which IP address and TCP port would you like your new server to listen?	IP Address: 9.5.139.248 ² Port: 80
Do you want your new server to use an access log?	Yes

¹This server name also becomes the name of the Cluster Resource Group (CRG).
²This is the address of the clustered IP address shown in Figure 17.1 on page 265.

- 2 Add the HA server directives to the HAHTTPSVR server on rchas837:
 - a Select the **Manage** tab.
 - b For Server, select **HAHTTPSVR**. For Server area, select **Global configuration**.
 - c In the left-hand panel, click **System Resources**.
 - d Select the **Highly Available Server** tab.
 - e Select **Enable HTTP server to be highly available**. This expands your options for this tab as shown in Figure 17.3 on page 270. Specify the following options:

- i. Select **Primary/backup with IP takeover**.
- ii. In the Liveness monitor settings area, enter the Liveness check URL:
`http://9.5.139.248:80/index.html`
- iii. For the remaining parameters, keep the defaults.

The Liveness check URL is used by the Liveness Monitor to perform liveness checks on the server. This URL is slow polled (as defined by the Time between liveness checks) by the backup server to determine if the primary has failed. We recommend that the target of the URL specified not be overly complex in addition to being responsive (i.e. response to Liveness check URL should be as fast as possible).

- f Click OK.

Figure 17.3 Enabling HA for the HAHTTPSVR server on RCHAS837

Highly Available Server
Denial of Service
Advanced

Specify one specific server IP address to listen on: ?

	IP address	Port	FRCA
<i>Example</i>	<i>All IP addresses</i>	<i>80</i>	<i>Disabled</i>
<input type="radio"/>	9.5.139.248	80	Disabled

Enable HTTP server to be highly available ?

- Primary/backup with IP takeover
- Primary/backup with network dispatcher
- Peer

Enable highly available CGI programs ?

Liveness monitor settings

Liveness check URL:

Backup server URL:

Time between liveness checks: Seconds or.

Maximum time to wait for responses: Minutes or.

Maximum number of recovery attempts: or... ?

Exit program

Library: ?

Program: ?

- 3** Display the httpd.conf configuration file to view three new directives that are added as shown in Figure 17.4 on page 271.

Figure 17.4 Three new directives to support HA in HAHTTPSVR

```
LoadModule ha_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
...
HAModel PrimaryBackupWithIPTakeover
LmURLCheck http://9.5.139.248/index.html
...
```

- 4** Create an identical HTTP Server configuration and Web application on your backup iSeries server rchas838. There are many ways to accomplish this. As a high-level overview, we recommend this method:
 - a** Use the Admin GUI to create another HAHTTPSVR HTTP Server (powered by Apache) configuration on rchas838 using the same information found in Table 17.1 on page 268.
 - b** Copy and paste the new HA directives from the HAHTTPSVR httpd.conf configuration file on rchas837 to rchas838.
 - c** Copy and paste the entire Web application's HTML, GIFs, and other collateral from the IFS on rchas837 to rchas838. Place these files in a similar directory structure.

CHAPTER 18

CONFIGURING WEBSPHERE APPLICATION SERVER

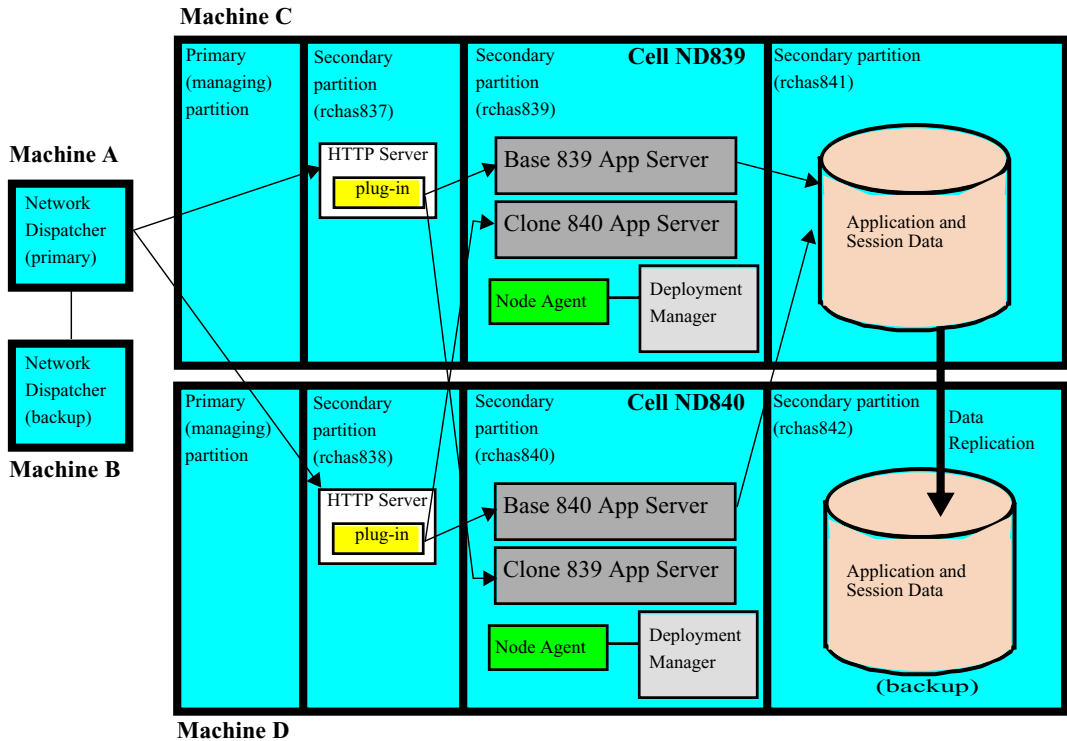
In this chapter the Dual Cell topology described in “Peer Cells using Network Dispatcher” on page 236 shall be implemented and a sample application installed. In Chapter 15, “Recommended Topologies” on page 233, the three recommended topologies where:

- 1** Peer cells with Network Dispatcher providing Web server clustering
- 2** Primary/Backup cells with Network Dispatcher providing Web server clustering
- 3** Primary/Backup cells with HA Apache Web server

The heart of each of these topologies is the dual cell implementation detailed in this chapter. One of the primary advantages of dual cell is the fact that one cell can be taken off-line and your enterprise is still operational. For configurations 2 and 3 WebSphere Horizontal clustering is a requirement to assure redundancy for application servers. For configuration 1 it is optional. The sample implementation below will include horizontal clustering. For more discussion on this configuration please reference the chapter mentioned above.

Dual Cell Topology Implementation

Figure 18.1 WebSphere Application Server dual cell with Network Dispatcher



In order to implement the WebSphere dual cell topology depicted in Figure 18.1, the following steps must be taken:

- 1 Install required products
- 2 Create Network Deployment instances
- 3 Create Base Application Server instances
- 4 Federate Base Application Server instances
- 5 Create horizontal clusters
- 6 Configure resources
- 7 Install applications
- 8 Enable session persistence
- 9 Generate plug-in file
- 10 Update TCP/IP configuration values

Step 1. Install Required Products

On partitions rchas839 and rchas840 install the WebSphere 5.x Base Application Server, Network Deployment and Apache Web server. The online installation instructions can be found at:

- Base Application Server – Navigate to **WebSphere Application Server home => Installation => Base installation**. This documentation also includes instruction for installing Apache Web server.
- Network Deployment -- Navigate to **WebSphere Application Server home => Installation => Network Deployment Installation**

At this point in time the relevant subsystems for WAS 5.x Base and Network deployment should be started. The relevant subsystems are QEJBAS5 for Base and QEJBASND5 for Network Deployment. If they are not started then issue the following set of commands on rchas839 and rchas840:

```
STRSBS SBSD(QEJBAS5/QEJBASND5) – Starts Network Deployment subsystem
```

```
STRSBS SBSD(QEJBAS5/QEJBAS5) – Starts Base application server subsystem
```

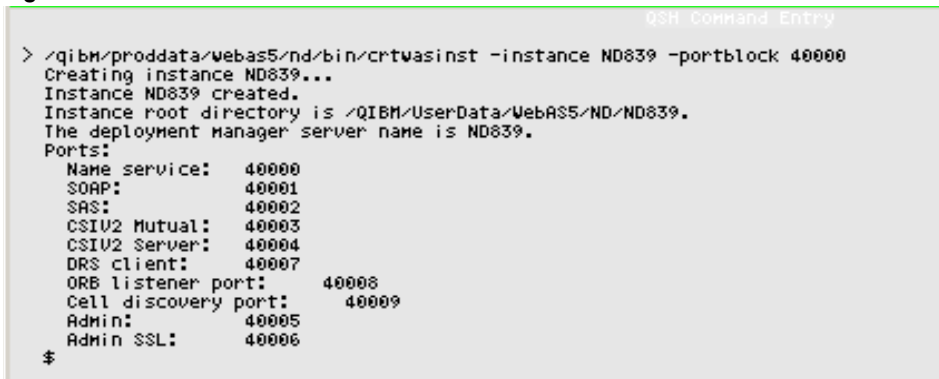
Step 2. Create Network Deployment Instances

Create Network deployment instances ND839 and ND840 on rchas839 and rchas840 respectively, using the WebSphere `crtwasinst qshell` command. On a command line on system rchas839, enter `strqsh` and issue the following command:

```
/qibm/proddata/webas5/nd/bin/crtwasinst -instance ND839 -portblock 40000
```

The use of parameter `portblock` will help avoid potential port conflicts. The results of the command is illustrated in Figure 18.2 on page 275.

Figure 18.2 Result of `crtwasinst` command on rchas839



```
QSH Command Entry
> /qibm/proddata/webas5/nd/bin/crtwasinst -instance ND839 -portblock 40000
Creating instance ND839...
Instance ND839 created.
Instance root directory is /QIBM/UserData/WebAS5/ND/ND839.
The deployment manager server name is ND839.
Ports:
  Name service: 40000
  SOAP: 40001
  SAS: 40002
  CSIU2 Mutual: 40003
  CSIU2 Server: 40004
  DR$ client: 40007
  ORB listener port: 40008
  Cell discovery port: 40009
  Admin: 40005
  Admin SSL: 40006
$
```

Make note of the Admin and the SOAP ports, these being 40005 and 40001 respectively. The Admin port will be used to bring up the administrative console and the SOAP port is needed to federate nodes/instances.

Proceed to create the Network Deployment Manager ND840 on rchas840 using the following command (again noting the Admin and SOAP ports):

```
/qibm/proddata/webas5/nd/bin/crtwasinst -instance ND840 -portblock 40000
```

Now that we have created the Network Deployment instances, we need to start the Network Deployment Manager on rchas839 and rchas840 using the WebSphere startManager qshell command. On rchas839, issue the following qshell command: /qibm/proddata/webas5/nd/bin/startManager -instance ND839

The results of the command is illustrated in Figure 18.3 on page 276.

Figure 18.3 Result of startManager command on rchas839

```

QSH Command Entry
> /qibm/proddata/webas5/nd/bin/startManager -instance ND839
CPC1221: Job 502967/QEJBSUR/ND839 submitted to job queue QEJBNDJOBQ in
library QEJBAS5.
EJB6123: Application server started.
Cause . . . . . : Application server ND839 in ND instance ND839 has
started and is ready to accept connections on admin port 40005.
$

```

Figure 18.4 on page 276 illustrates the fact the ND839 process is started within the QEJBASND5 subsystem.

Figure 18.4 Network Deployment Manager ND839 running on rchas839

```

WORK WITH ACTIVE JOBS
CPU %: 3.0 Elapsed time: 03:57:44 Active jobs: 188 12/31/03 16:35:33
Type options, press Enter.
2=Change 3=Hold 4=End 5=Work with 6=Release 7=Display message
8=Work with spooled files 13=Disconnect ...

Opt Subsystem/Job User Type CPU % Function Status
---
--- QSYSSCD QPGMR BCH .0 PGM-QEZSCNEP EUTW
--- QEJBASND5 QSYS SBS .0
--- ND839 QEJBSUR BCH .2 PGM-QEJBSRSTRSUR JUAW
--- QEJBAS5 QSYS SBS .0
--- SERVER1 QEJBSUR ASJ .0 PGM-QEJBSRSTRSUR JUAW
--- QNTTOSUP QSYS SBS .0

```

Now start Network Deployment manager ND840 on rchas840 by issuing the following qshell command:

```
/qibm/proddata/webas5/nd/bin/startManager -instance ND840
```

Verify node manager ND840 is running in subsystem QEJBASND5.

The Network Deployment managers allows one to administer the cell via the browser-based Administration Consoles. Proceed to bring up the admin consoles for cells ND839 and ND840 by bringing up separate browsers for the following URLs:

- <http://rchas839:40005/admin/>
- <http://rchas840:40005/admin/>

NOTE Port 40005 is the Admin port which was obtained when the Network Deployment instances were created.

Step 3. Create Base Application Server Instances

We shall proceed to create WAS Base Application server instances on rchas839 and rchas840. Four base application server instances shall be created with two residing on rchas839 and two on rchas840 (later on these instances shall be federated into the two Network Deployment cells created previously). The instances names, functional description and indication of federated cell follows:

- Base instances which shall be federated into Cell ND839
 - WebSphere base instance BASE839 to be created on rchas839. This instance shall house the base application server running on rchas839.
 - WebSphere base instance CLONE839 to be created on rchas840. This instance shall house the clone of the application server running in instance BASE839.
- Base instances which shall be federated into Cell ND840
 - WebSphere base instance BASE840 to be created on rchas840. This instance shall house the base application server running on rchas840.
 - WebSphere base instance CLONE840 to be created on rchas839. This instance shall house the clone of the application server running in instance BASE840.

WebSphere qshell script `crtwasinst` is used to create the Base instances. To create the base instances on rchas839, enter the following sequence of commands from within the qshell environment:

```
1 /qibm/proddata/webas5/base/bin/crtwasinst -instance BASE839
   -portblock 30000 -noembeddedjms -nodefaultapps
2 /qibm/proddata/webas5/base/bin/crtwasinst -instance CLONE840
   -portblock 30100 -noembeddedjms -nodefaultapps
```

The use of parameter `noembeddedjms` is specified to insure an embedded Java Message Service (JMS) provider is not created for either instance. The application eventually deployed does not require JMS and in turn will not need the additional overhead. In addition, we avoid having the default application installed into these instances by specifying parameter `nodefaultapps`.

Now create the base instances on rchas840 by entering the following sequence of commands from within the qshell environment:

```
1 /qibm/proddata/webas5/base/bin/crtwasinst -instance BASE840
   -portblock 30000 -noembeddedjms -nodefaultapps
2 /qibm/proddata/webas5/base/bin/crtwasinst -instance CLONE839
   -portblock 30100 -noembeddedjms -nodefaultapps
```

Step 4. Federate Base Application Server Instances

We shall federate all four base instances created in the previous step into the appropriate cells. On rchas839, enter the following sequence of commands from within a qshell environment:

```
1 /qibm/proddata/webas5/base/bin/addnode rchas839 40001 -instance
   BASE839 -startingport 20500
2 /qibm/proddata/webas5/base/bin/addnode rchas840 40001 -instance
   CLONE840 -startingport 20600
```

What did we accomplish by issuing the `addnode` commands above? Base instances BASE839 was federated into cell ND839 and CLONE840 into cell ND840. Port 40001 is the SOAP port for both cells and is used to do the federation. The `startingport` parameter is used to avoid port conflicts.

On rchas840, enter the following sequence of commands:

- 1 /qibm/proddata/webas5/base/bin/addnode rchas840 40001 -instance BASE840 -startingport 20500
- 2 /qibm/proddata/webas5/base/bin/addnode rchas839 40001 -instance CLONE839 -startingport 20600

Issuing the above addnode commands resulted in base instance BASE840 being federated into cell ND840 and base instance CLONE839 into cell ND839. Port 40001 is the SOAP port for both cells and is used to do the federation.

Once all nodes are federated, then NODEAGENT processes should be running under subsystem QEJBAS5 on both rchas839 and rchas840. A NODEAGENT process enables administrative communication between the Network Deployment manager/cell and application servers deployed on federated nodes. Figure 18.5 on page 278 shows the two node agents running in subsystem QEJBAS5 on system rchas839.

Figure 18.5 Node agents running on system rchas839

Opt	Subsystem/Job	User	Type	CPU %	Function	Status
—	QSYSSCD	QPGMR	BCH	.0	PGM-QEZSCNEP	EVTU
—	QEJBASND5	QSYS	SBS	.0		DEQU
—	ND839	QEJBSUR	BCH	.0	PGM-QEJBSTRSUR	JUAW
—	QEJBAS5	QSYS	SBS	.0		DEQU
—	NODEAGENT	QEJBSUR	BCH	.0	PGM-QEJBSTRSUR	JUAW
—	NODEAGENT	QEJBSUR	BCH	.0	PGM-QEJBSTRSUR	RUN

You can determine which instance a NODEAGENT job is managing by looking at the job log of the NODEAGENT job. The job log contains the directory path to the instance that the NODEAGENT is managing. For example, the job log shown in Figure 18.6 on page 278 shows that NODEAGENT job 503522/QEJBSVR/NODEAGENT is managing base instance BASE839.

Figure 18.6 Job log of Node Agent running on rchas839

```

Job . . . : NODEAGENT      User . . . : QEJBSUR      Number . . . : 503522
>> CALL PGM(QEJBAS5/QEJBSTRSUR) PARM('-instance' '/QIBM/UserData/WebAS5/Base/
BASE839' '-server' 'nodeagent')

```

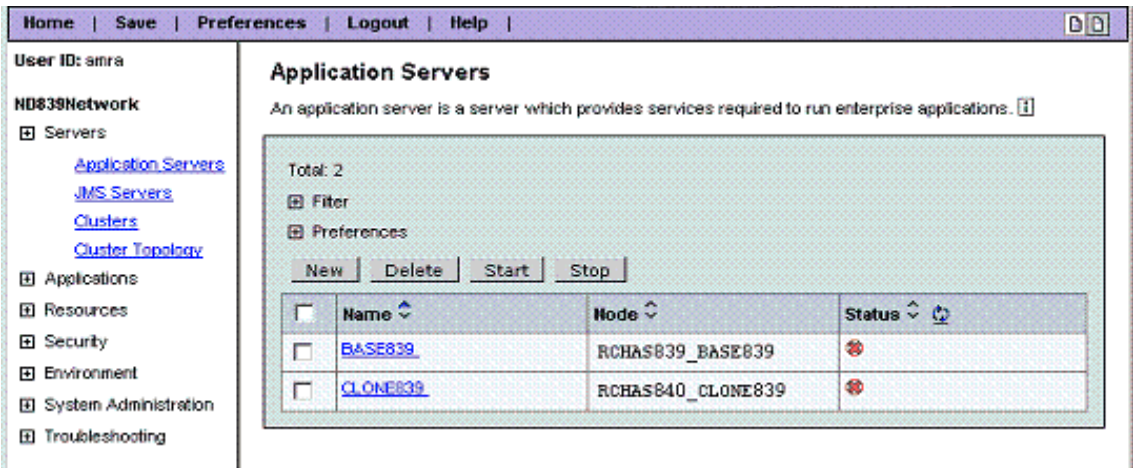
Let's briefly recap what has been accomplished. Two Network deployment cells (ND839 and ND840) have been created and the Network Deployment managers (Admin consoles) used to manage each have been started. Also four base application server instances (BASE839, CLONE840, BASE840 and CLONE839) have been created and federated into the appropriate cells. Node agents are started on each system which allows the deployment and administration of applications deployed in the base instances using the Network Deployment managers (Admin console). Notice we have not yet deployed any applications. Before we deploy applications we want to create WebSphere clusters using the Network Deployment managers. In addition, any resources which the application may need must also be configured before application deployment.

Step 5. Create Horizontal Clusters

We shall be creating clusters “cluster839” and “cluster840” in cells ND839 and ND840, respectively. We will proceed to create the cluster in cell ND839. Similar steps will need to be performed to create the cluster in cell ND840.

Let’s start with the admin console for cell ND839. From the navigation tree on the left side of the administration console, click on **Servers** => **Application Servers**. The screen shot below illustrates what you should see.

Figure 18.7 Application servers within cell ND839



Two application servers are displayed. It is easy to get application servers mixed up with a WebSphere instances and visa versa. Remember a WebSphere instance is a repository for one or more application servers and the applications deployed within those servers. An instance can have multiple application servers and each application server can have multiple applications deployed within it. When instance BASE839 and CLONE839 were federated into cell ND839, the base application servers are part of the instance and that is why they show up above.

Shortly we shall create a cluster to support horizontal cluster members. Each cell shall have one cluster which contains two cluster members. We shall use BASE839 application server as the base cluster member and create another cluster member in instance/node CLONE839 which resides on rchas840. Currently CLONE839 is a distinct separate application server that is not related to BASE839 – not a cluster member. So we want to delete CLONE839 now before we clone/cluster BASE839. Once deleted we should just see BASE839. Proceed to delete by selecting CLONE839 and clicking on **Delete**.

The next step is to create a server cluster. Remember a cluster is a logical collection of application server processes. You can use clusters to group servers for workload balancing. Application servers within a cluster are called cluster members. All cluster members in a single cluster must have identical application components deployed on them. Other than the applications configured to run on them, cluster members do not have to share any configuration data.

To create a cluster, click on **Servers** => **Clusters**. On the Server Cluster panel that is displayed, click on the **New** button and create cluster cluster839 as shown in Figure 18.8 on page 280 below.

Figure 18.8 Creating a cluster

The screenshot shows a web browser window with a navigation menu on the left and a main content area titled "Create New Cluster". The navigation menu includes "User ID: amra", "ND839Network", "Servers" (with sub-links for Application Servers, JMS Servers, Clusters, and Cluster Topology), "Applications", "Resources", "Security", "Environment", "System Administration", and "Troubleshooting". The main content area is titled "Create New Cluster" and contains a form for "Step 1: Enter Basic Cluster Information". The form has the following fields and options:

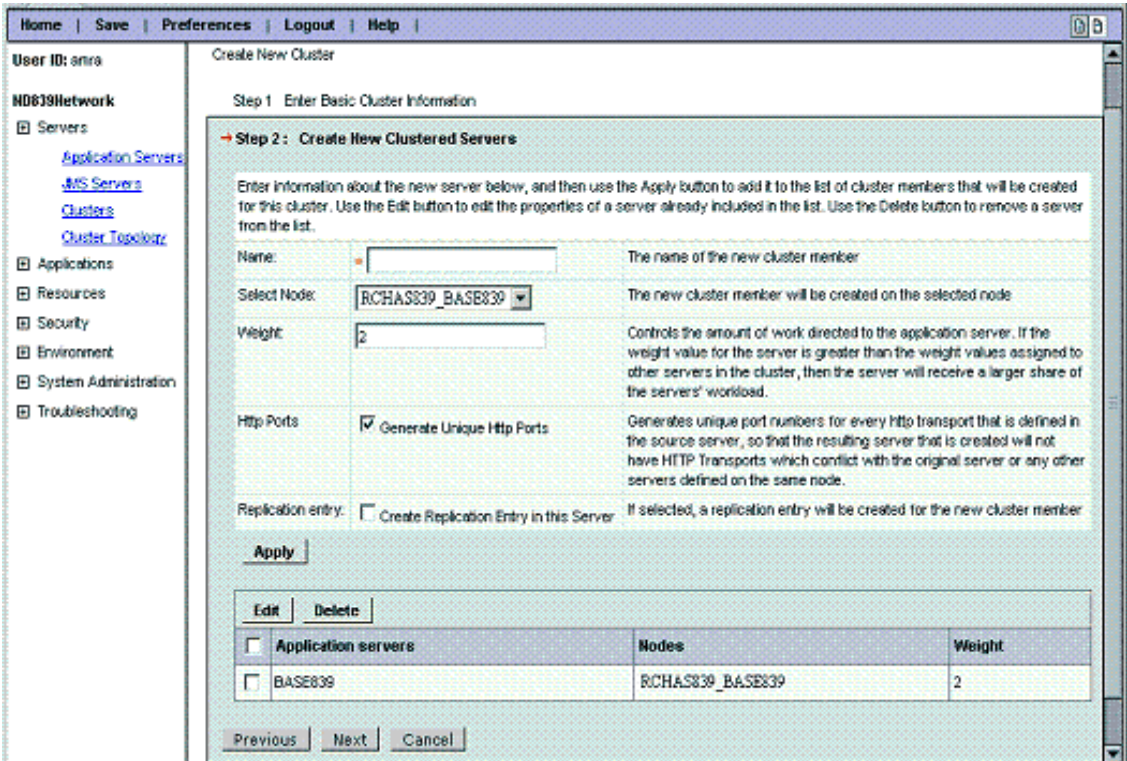
- Cluster name:** A text input field containing "cluster839".
- Prefer local:** A checked checkbox labeled "Prefer local enabled".
- Internal replication domain:** An unchecked checkbox labeled "Create Replication Domain for this cluster".
- Existing server:** Two radio buttons. The first is "Do not include an existing server in this cluster" (unchecked). The second is "Select an existing server to add to this cluster" (checked). Below this is a dropdown menu showing "ND839Network/RCHAS839_BASE839/BASE839" and a "Weight:" input field containing "2".

At the bottom of the form are "Next" and "Cancel" buttons.

The prefer local option only pertains to applications where the Enterprise Java Beans container runs in a process separate from the application Servlets and JSPs. For Web clients we recommend that Web and EJB containers run in the same process so this option does not pertain. We shall not use a replication domain to insure HTTP sessions are highly available, but instead we shall later configure persistent database. We shall pick application server BASE839 as the first cluster member. For now the server weight is left at 2, more on this later. Click on the **Next** button.

Notice in Figure 18.9 on page 281 that application server BASE839 is displayed as the only cluster member on node RCHAS839_BASE839.

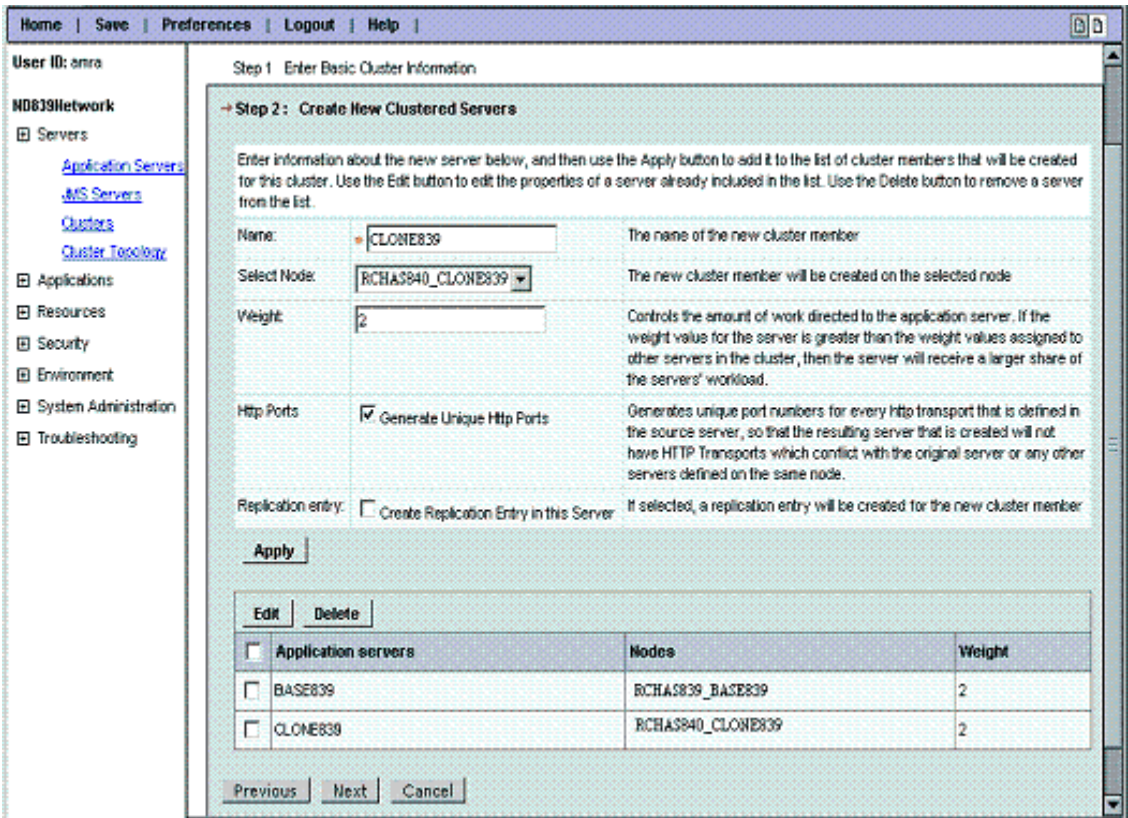
Figure 18.9 After creating a cluster



Now create a second cluster member called CLONE839 on node RCHAS840_BASE840 by setting the Name field to CLONE839 and selecting the RCHAS840_BASE840 value from the Select Node selection list. Click **Apply**.

Figure 18.10 on page 282 shows that CLONE839 has been created on node RCHAS840_CLONE839. The server actually resides in base instance CLONE839.

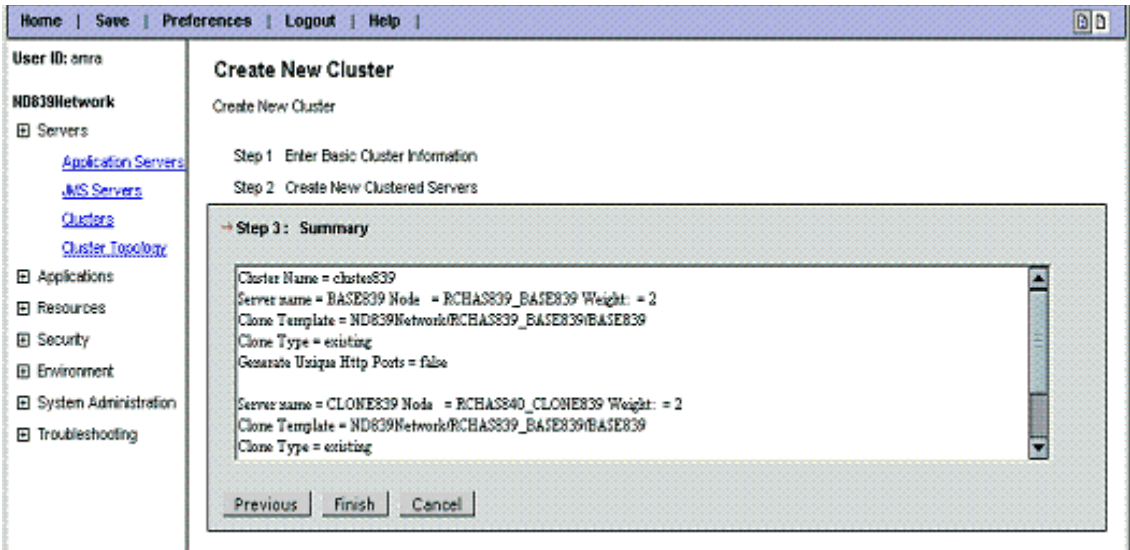
Figure 18.10 Creating a new cluster member



What we have created is referred to as a horizontal cluster since the members' span more than one node. A vertical cluster would contain only members residing on the same node. Eventually a Web server plug-in will get generated which shall contain the necessary information for WebSphere plug-in code to spray requests to the clustered application servers thereby providing both load balancing and failover.

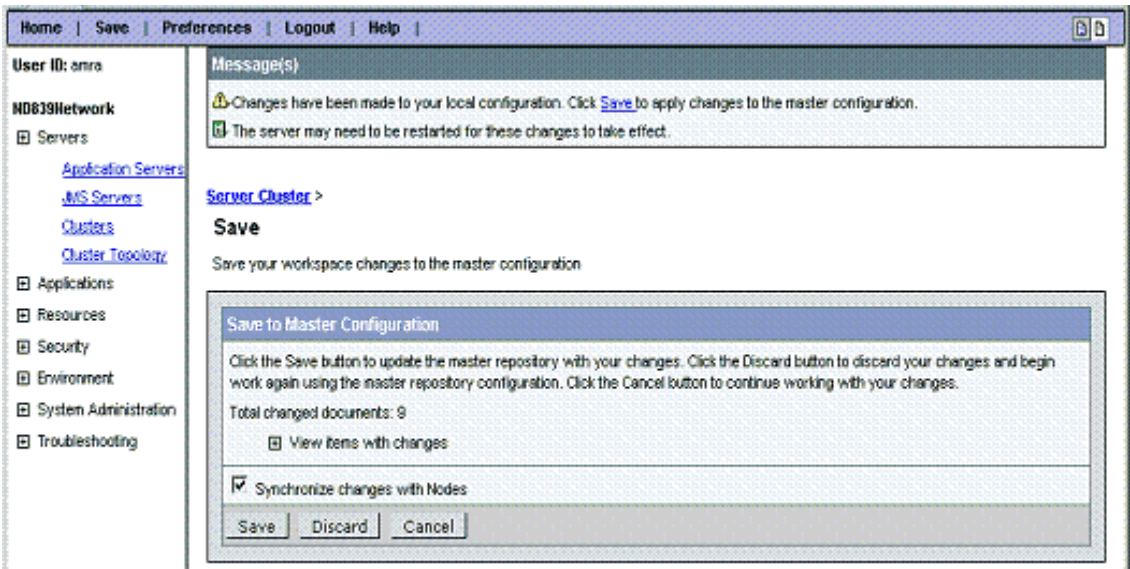
Click **Next**. You will see a summary of the cluster just created, as shown in Figure 18.11 on page 283.

Figure 18.11 Summary information for cluster that was created



Click **Finish**. The next panel will be WebSphere indicating that a save to the master configuration file needs to be performed. Save the changes by clicking **Save**, which will result in the panel shown in Figure 18.12 on page 283.

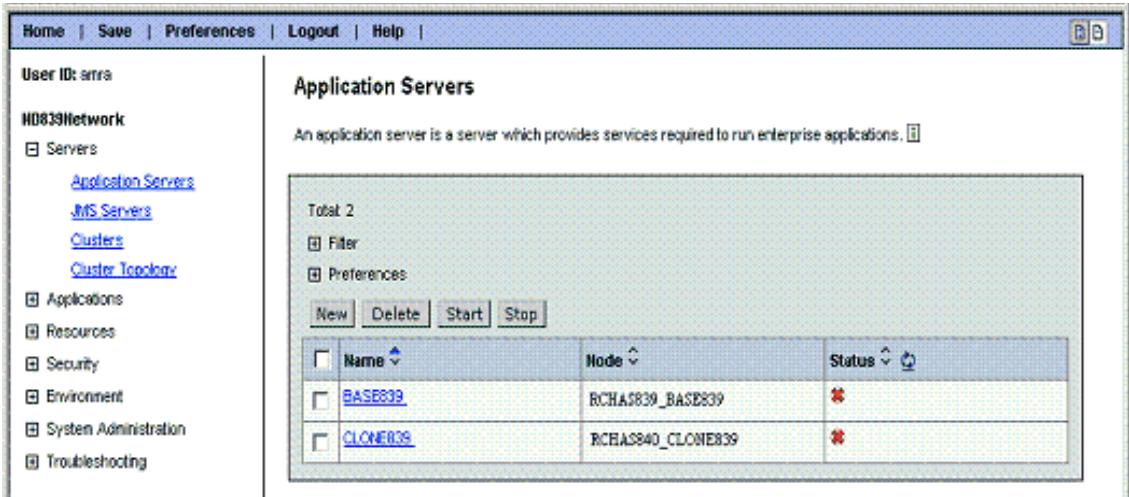
Figure 18.12 Saving changes to master configuration file



Ensure **Synchronize changes with Nodes** is selected and click the **Save** button.

From the navigation tree on the left side of the administration console, click on **Servers** => **Application Servers**. Notice that both **BASE839** and **CLONE839** in Figure 18.13 on page 284 show up as application servers.

Figure 18.13 Application servers - stopped

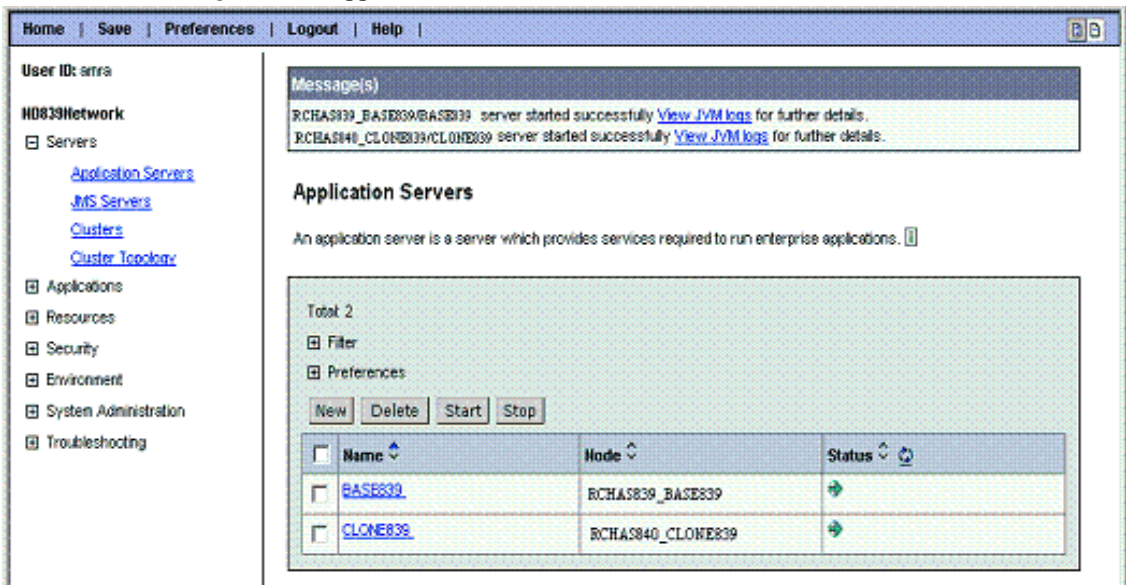


Start both application servers by selecting the application servers and clicking **Start**.

NOTE Control will not return until the application servers have started or failed in some manner. Please be patient.

The base cluster member BASE839 residing on node RCHAS839_BASE839 and the cluster member CLONE839 residing on RCHAS840_CLONE840 have now started as indicated by an arrow pointing to the right in the status column in Figure 18.14 on page 284.

Figure 18.14 Application servers - started



If the application server does not start then investigate system error and output files. The most likely cause is a port conflict which can be resolved by changing the portblock for

the application server having the problem. For example, suppose CLONE839 does not start due to a port conflict. To resolve this one would perform the following steps on system rchas840 from within the qshell environment:

```
/qibm/proddata/webas5/base/bin/chgwassvr -instance CLONE839 -server CLONE839 -portblock 27000.
```

Now try to start the application server again.

Note that no application has yet been installed. Before the installation of the application is performed, resources that the application needs must be defined and created.

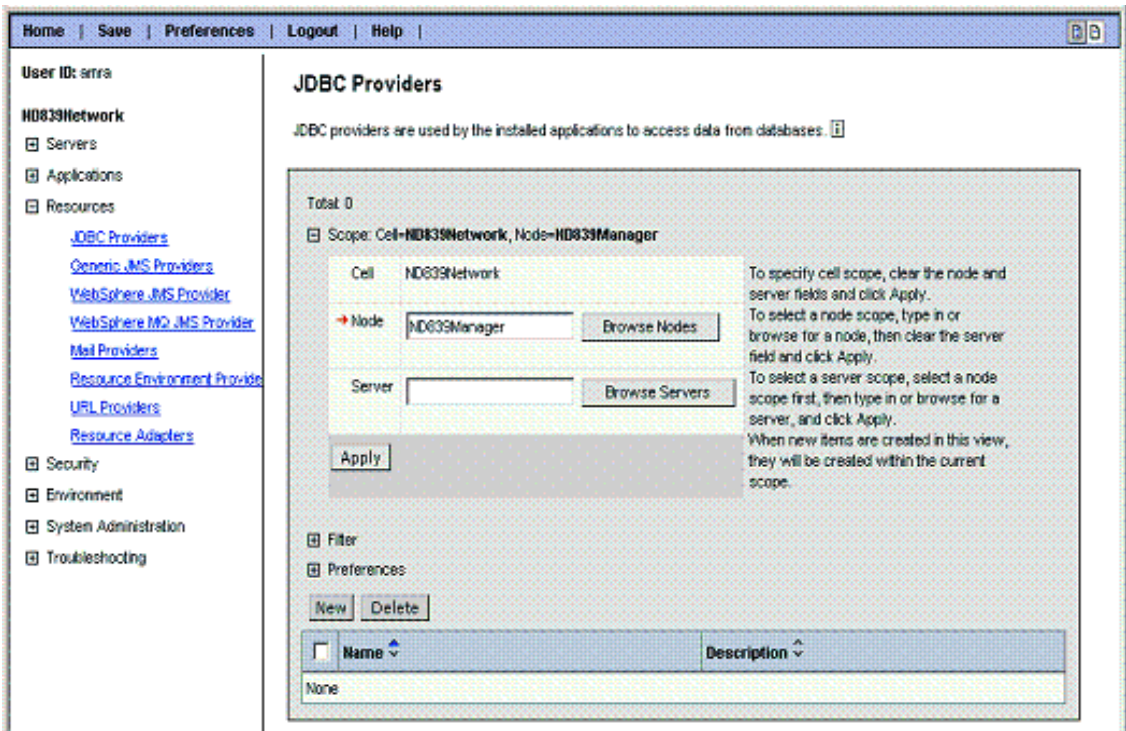
The same steps to create a cluster must also be performed for cell ND840 on rchas840.

Step 6. Configure Resources

The application needs access to the database residing on rchas841. We shall use a JDBC provider for this access. We will proceed to create the JDBC provider in cell ND839. Similar steps will need to be performed to create the JDBC provider in cell ND840 on rchas840.

From the navigation tree on the left side of the administration console, click **Resources** => **JDBC Providers**. Figure 18.15 on page 285 shows the JDBC Providers panel.

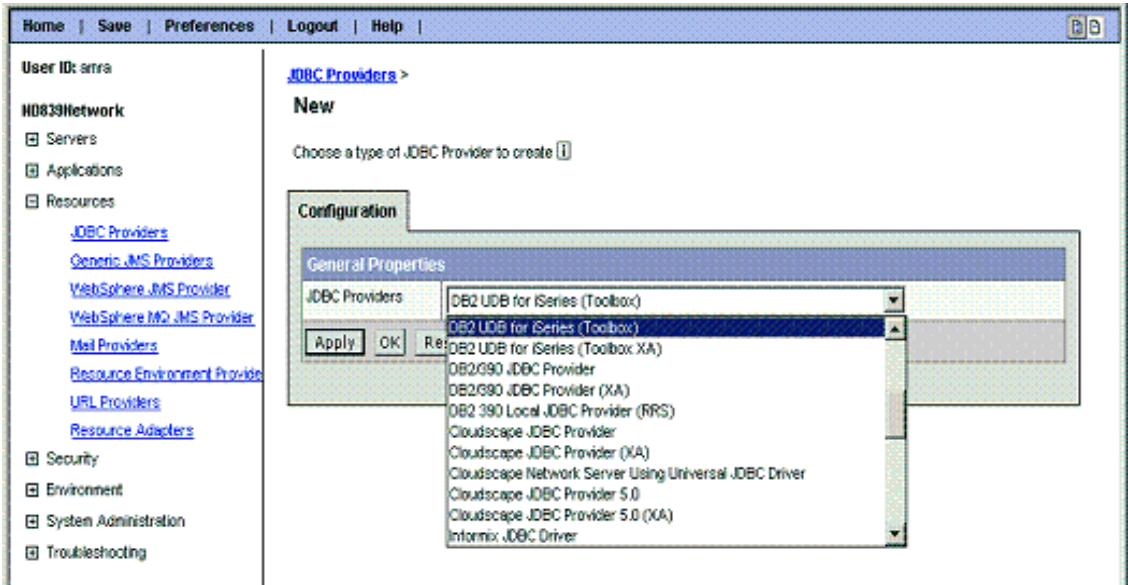
Figure 18.15 JDBC Provider panel - node scoped



Notice that the scope of the resource is node-scoped. We want the resources to be cell-scoped so that the resource can be used by all cluster members contained within the cell. To make the resource cell-scoped, blank out the Node field and click **Apply**.

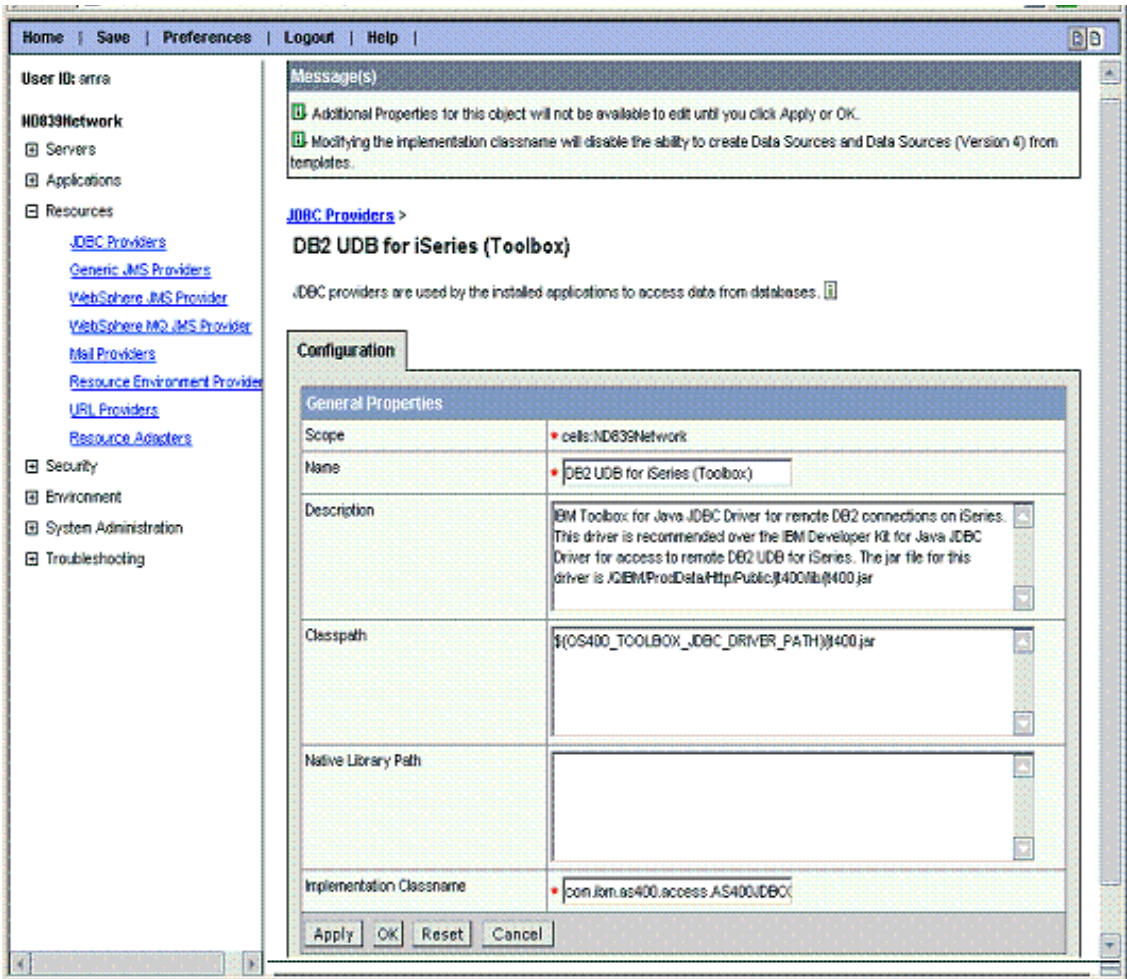
Now click **New** to define a cell scoped JDBC provider.

Figure 18.16 New JDBC provider



On iSeries we have several choices for JDBC providers, the provider we shall use is “DB2 UDB for iSeries (Toolbox)”. Click the drop-down button and click “DB2 UDB for iSeries (Toolbox)” and proceed to click **OK**. The next panel displayed is shown in Figure 18.17 on page 287.

Figure 18.17 DB2 UDB for iSeries JDBC provider configuration

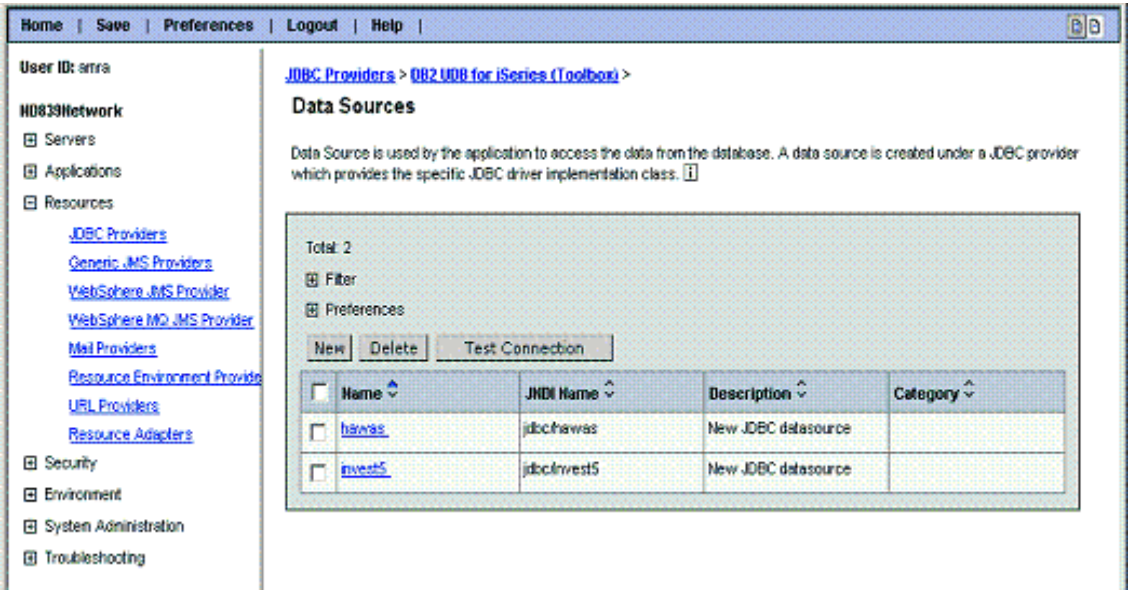


Click **OK** and save the changes to the master configuration as we did before, ensuring that the Synchronize changes with Nodes is checked prior to performing the save.

We are not going to step through all the displays for creating the JDBC provider and data source. Instead we shall just focus on the data source custom properties which are important when cluster members communicate with a highly available database.

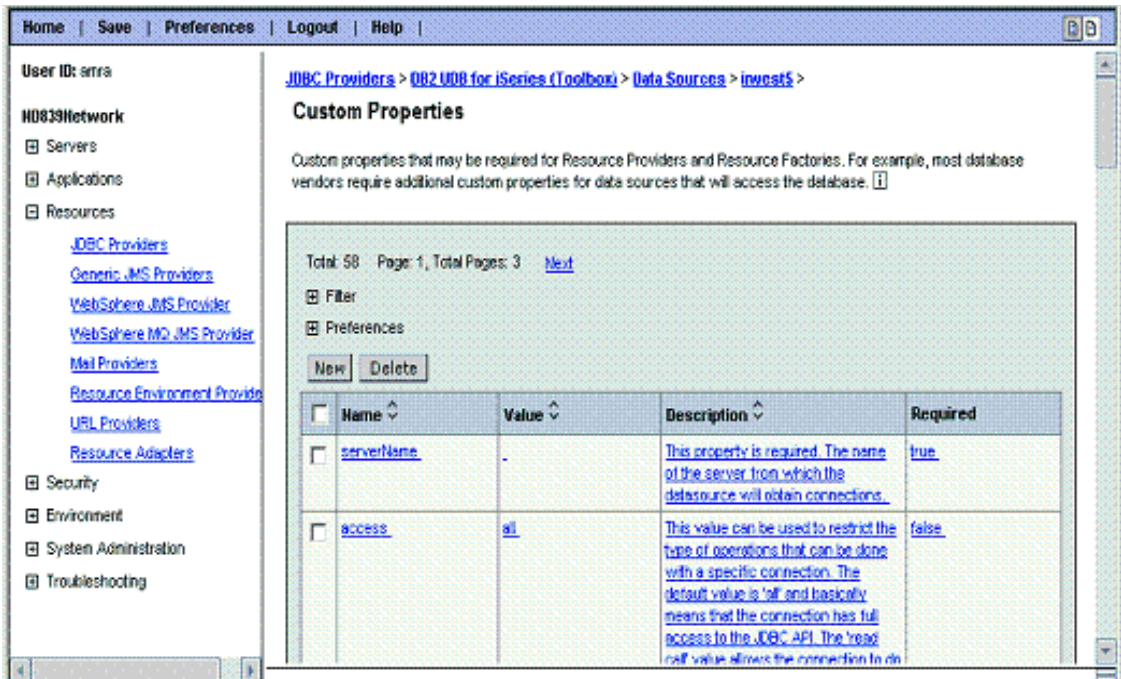
For our sample application we have created three data sources called invest4, invest5 and hawas. Invest4 and invest5 are version 4 and 5 data sources respectively. The applications which will be installed will need both. The hawas data source is a version 5 data source which will be used to persist HTTP sessions to the database on rchas841. The custom properties for the three are the same. The hawas and invest5 data sources are shown in Figure 18.18 on page 288. We shall illustrate the custom property changes for data source invest5.

Figure 18.18 The hawas and invest5 data sources



Click on **invest5**. On the panel that comes up, scroll to the bottom and click on **Custom Properties**. Figure 18.19 on page 288 shows the Custom Property panel for data source invest5.

Figure 18.19 Custom Properties for data source invest5



Three changes are required to data source Custom Property defaults. The changes and reasoning are as follows:

- **serverName**

Name of the database server to which the data source will connect to. In this scenario host name rchas841 which resolves to an IP address. This is the IP address used by HA database and iSeries clustering to provide a switchable resource. In this scenario the switchable resource is our enterprise database containing application data and persistent session data containing shopping cart information. The WebSphere connection manager will manage the availability of the pool of connections.

- **receiveBufferSize**

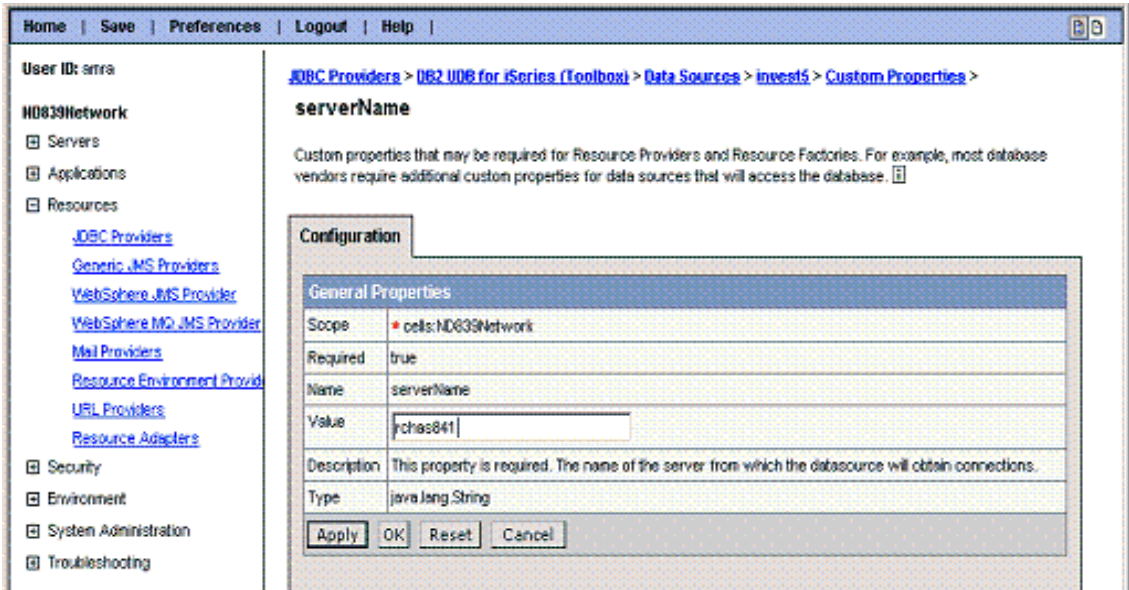
The transport mechanism between the Cluster Members and the host DB server jobs on rchas841 involves TCP/IP sockets. Typically database transfers from host DB servers (rchas841) and cluster members may be quite large. This can cause a socket with a default receive buffer size of 8192 to undergo extensive flow control iterations consuming undo system resource. To avoid this we recommend setting the receive buffer size to 1,000,000.

- **keepAlive**

If the system hosting the database (rchas841) goes down unexpectedly then it is possible that a socket reset may not flow from rchas841 and in turn not wake up or free the cluster member thread waiting for a response from the database server. Without the socket connection honoring the keepAlive timer, this job could sleep forever. The keepAlive property for the toolbox connection will cause a timer to be activated and a state set in the TCP/IP endpoint denoting the connection has been idle. If the idle timer pops and the connection were still idle, then a probe is sent out; otherwise the timer is reset. This will avoid any Web Sphere Application servers from hanging when the highly available database undergoes unplanned failover. The TCP/IP keepAlive timer also needs to be set using CFGTCP. The TCP/IP changes are covered later.

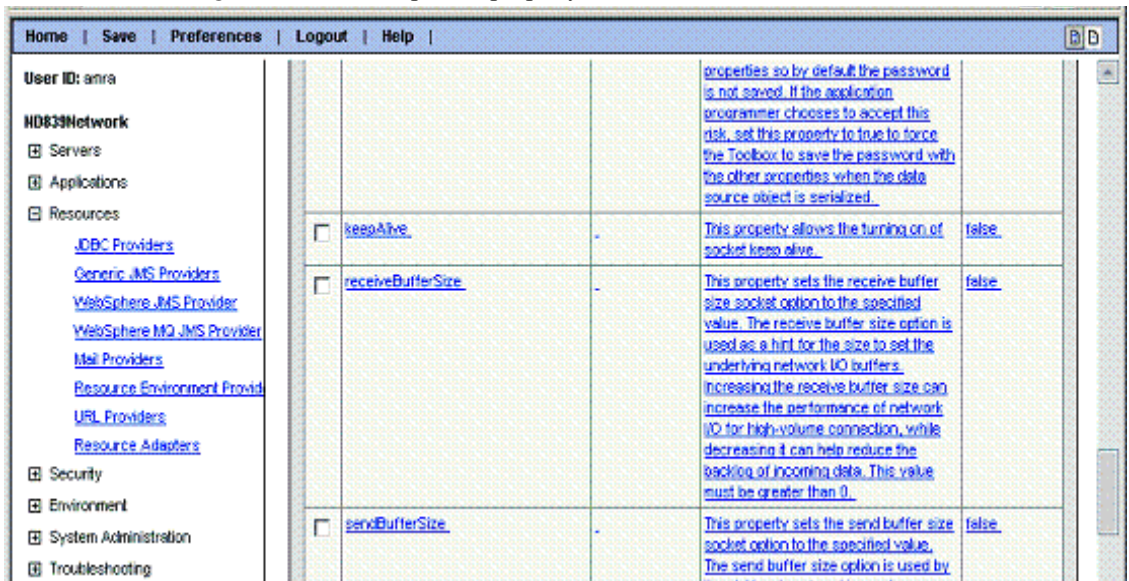
Click on **serverName** and in the configuration panel that comes up enter “rchas841” for serverName value. See Figure 18.20 on page 290.

Figure 18.20 Setting serverName data source property



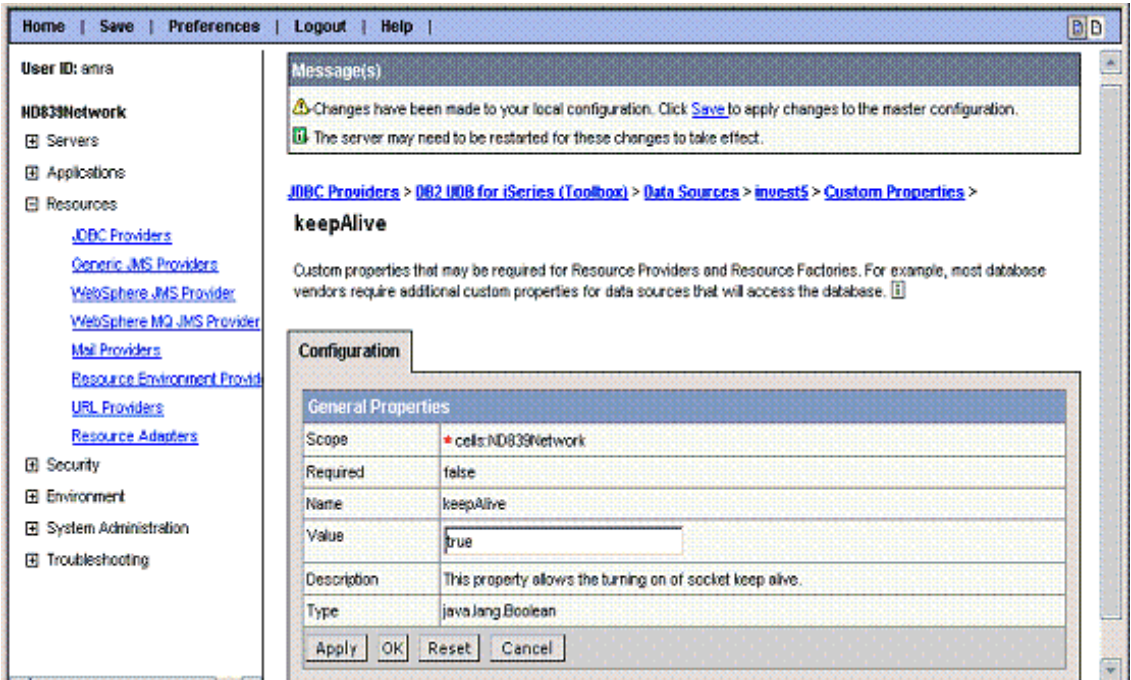
Click **OK**. This will bring you back to the initial Custom Properties panel. Navigate to the page (custom properties may span multiple pages- use the **Next** link to go to different pages) that contains the **keepAlive** property. The property is shown in Figure 18.21 on page 290.

Figure 18.21 The keepAlive property



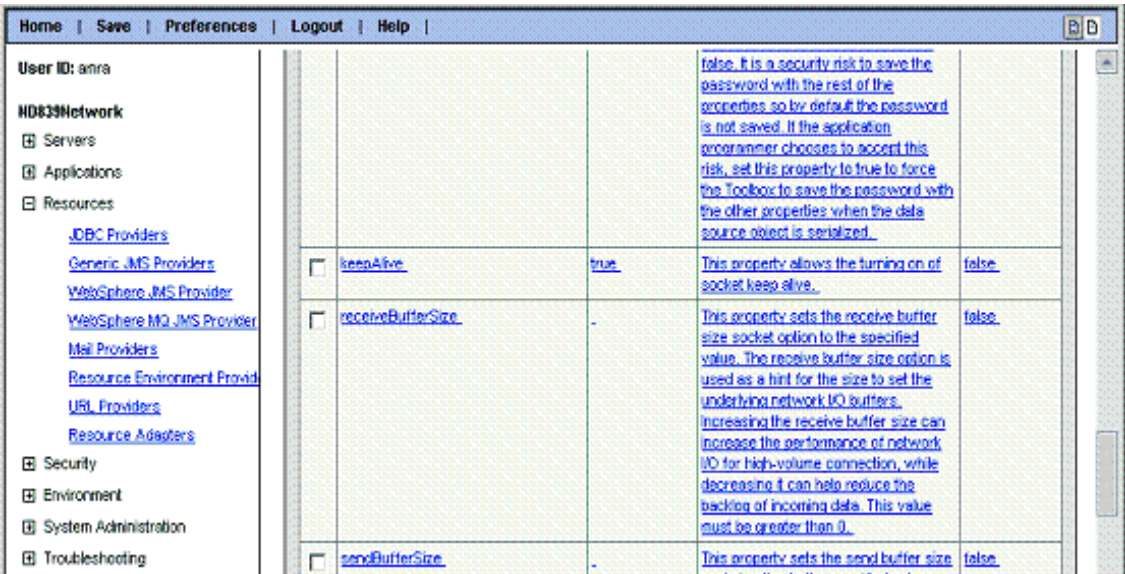
Click on the **keepAlive** property and in the configuration panel that comes up set the **keepAlive** property to **true**. See Figure 18.22 on page 291.

Figure 18.22 Setting keepAlive data source property



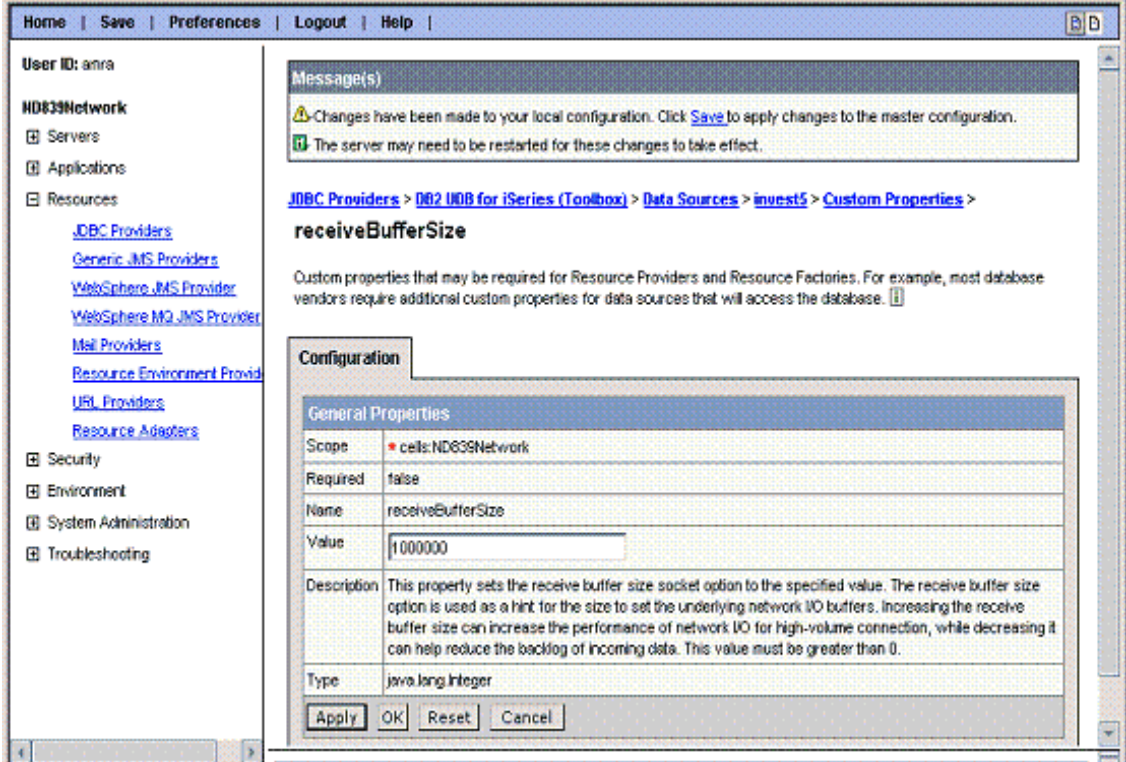
Click **OK**. As it did before, clicking **OK** will bring you back to the initial Custom Properties panel. Navigate to the page that contains the receiveBufferSize property. The property is shown in Figure 18.23 on page 291.

Figure 18.23 The receiveBufferSize property



Click on the **receiveBufferSize** property and in the configuration panel that comes up set the receiveBufferSize property to 1000000. See Figure 18.24 on page 292.

Figure 18.24 Setting receiveBufferSize data source property



Click **OK** and save the changes to the master configuration as we did before, ensuring that the Synchronize changes with Nodes is checked prior to performing the save.

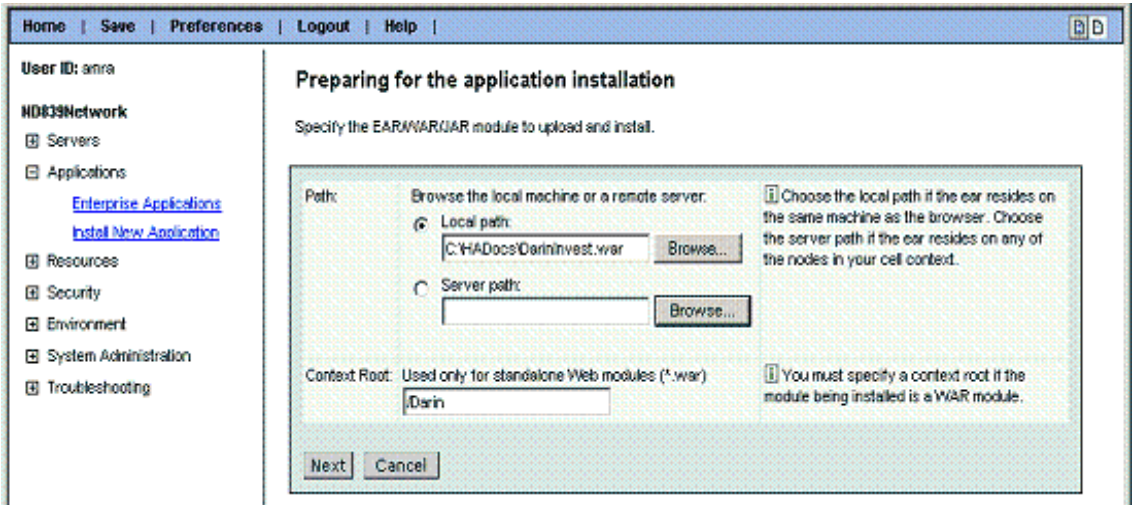
The same steps to create the JDBC provider must also be performed for cell ND840 on rchas840.

Step 7. Install Applications

We can now proceed to install the sample applications. The applications are packaged in DarinInvest.war and SteveInvest.ear files. The application will be deployed within cluster “cluster839” which contains cluster members BASE839 and CLONE839 on nodes rchas839 and rchas840 respectively. As part of the application install, WebSphere will automatically distribute the application to both federated base instances. We shall just cover the displays to install DarinInvest.war. Similar steps will need to be performed to install the applications in cell ND840.

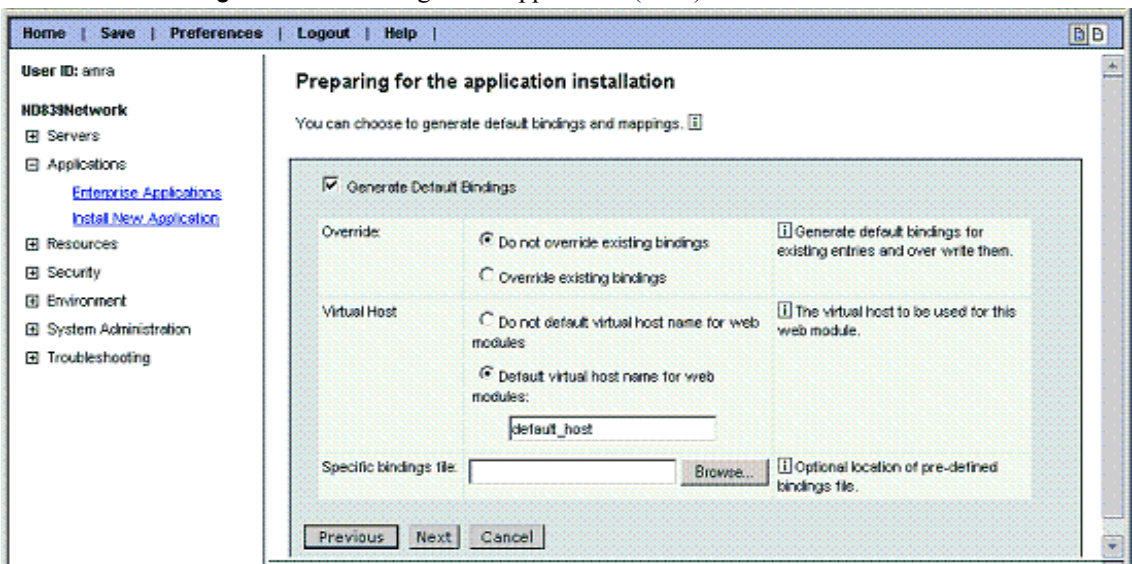
From the navigation tree on the left side of the administration console, click on **Applications => Install New Applications**. On the panel that is displayed, check Local Path, set the path to the application file, and set the Context Root to /Darin. See Figure 18.25 on page 293.

Figure 18.25 Installing a new application



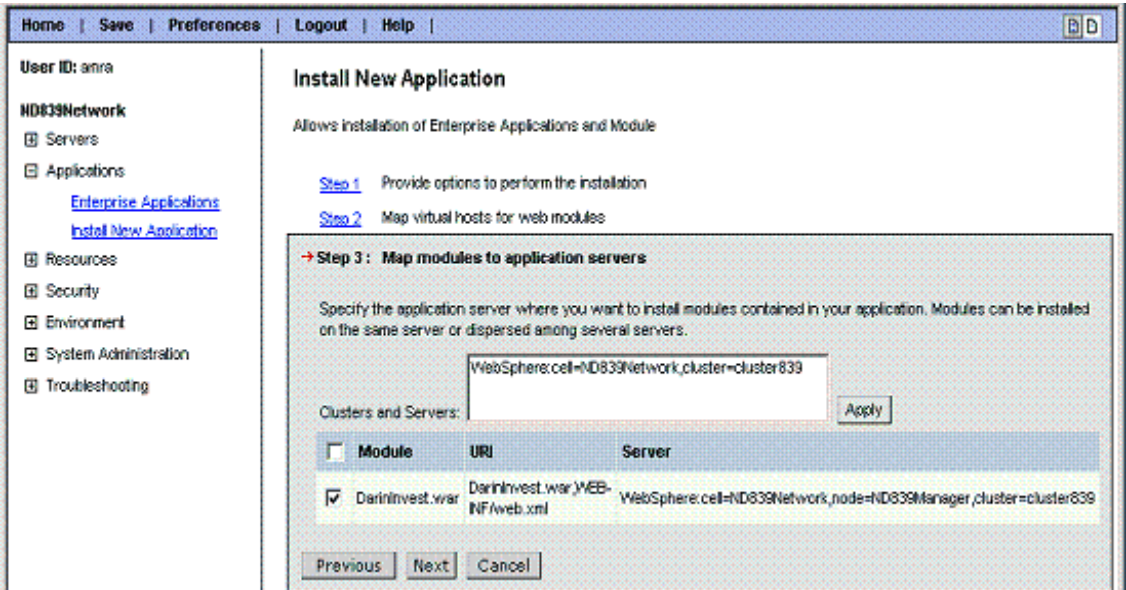
Click **Next**. On the panel shown in Figure 18.26 on page 293, check “Generate Default Bindings” and click **Next**.

Figure 18.26 Installing a new application (cont.)



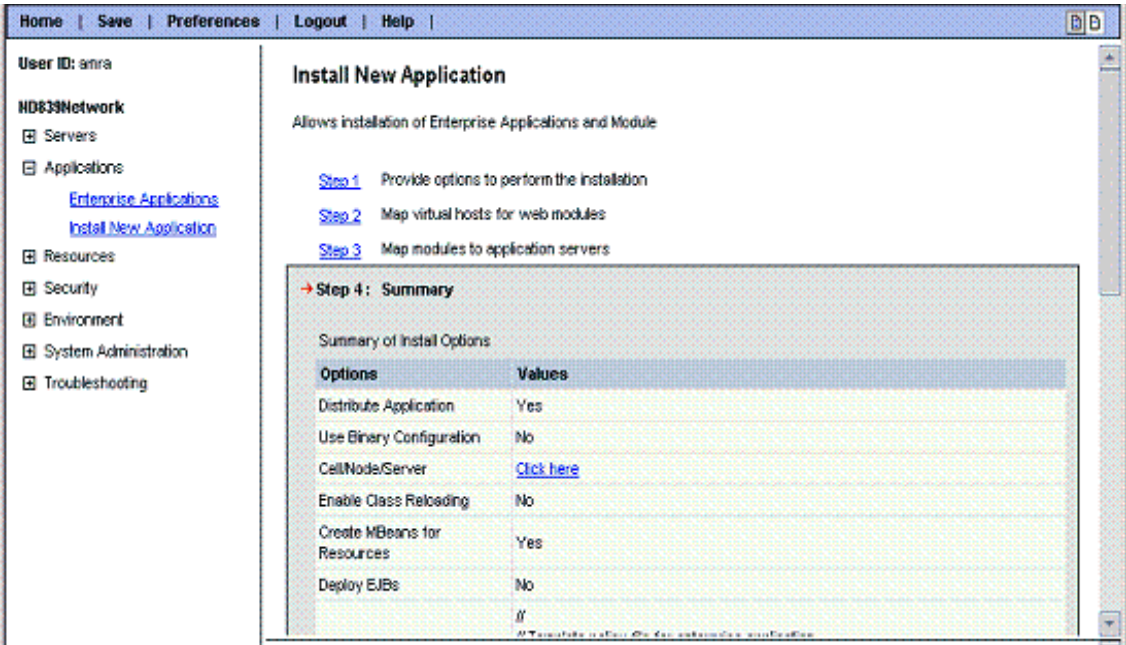
Click **Next** on the next two panels. On the Map modules to application servers panel shown in Figure 18.27 on page 294, select cluster Cluster839, and click **Next**.

Figure 18.27 Installing a new application (cont.)



Click **Finish** on the panel shown in Figure 18.28 on page 294.

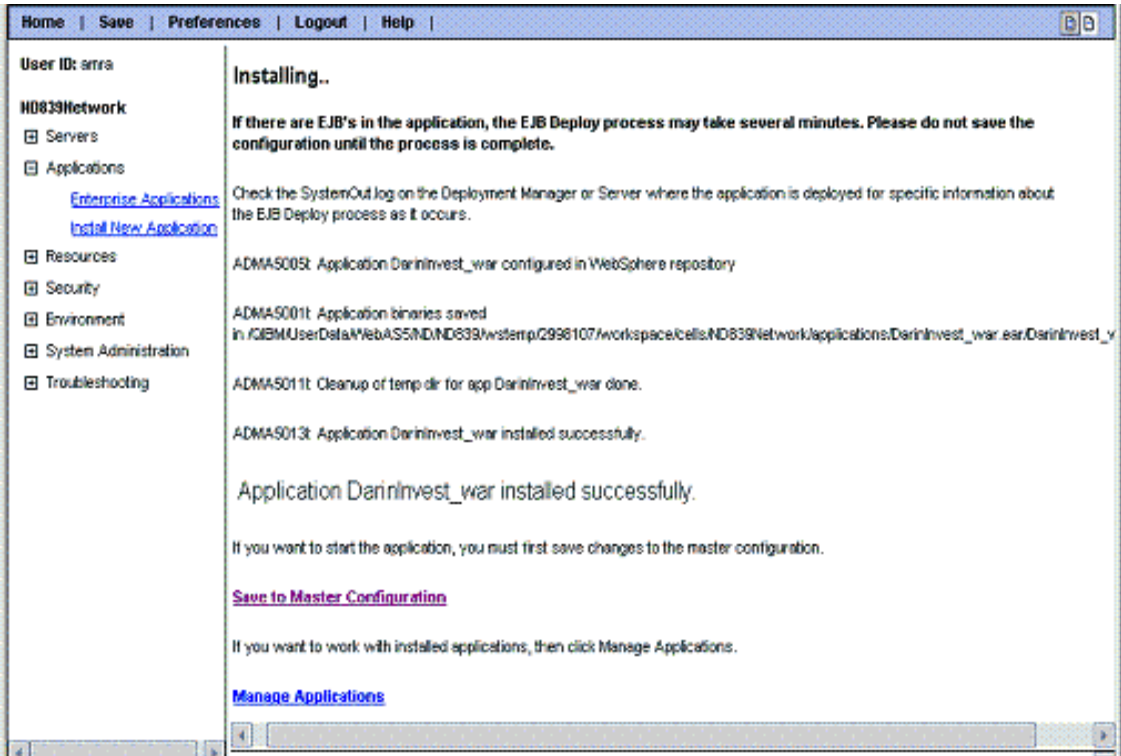
Figure 18.28 Installing a new application (cont.)



Almost done. You should see an indication that the application was installed successfully as shown in Figure 18.29 on page 295. Click **Save to Master Configuration**, which will take you to the Save panel that has been seen previously. On

that panel, ensure that the Synchronize changes with Nodes is checked and click on Save.

Figure 18.29 Installing a new application (cont.)



At this point, the application should have been distributed to all cluster members.

Application SteveInvest.ear was also installed although the install steps will not be shown here. Both applications were installed into the cluster which assures they are distributed to both instances federated into the cell.

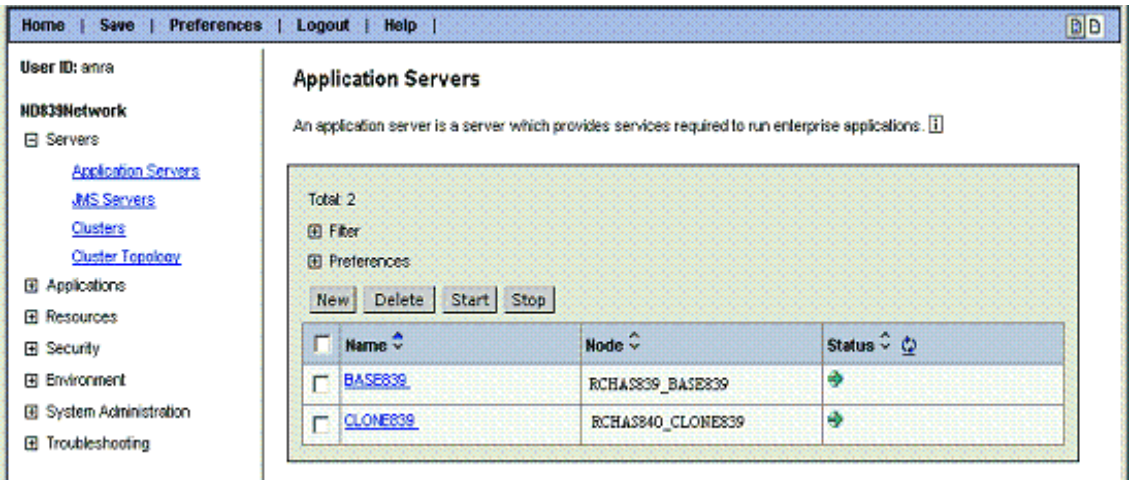
The same steps to install the application must also be performed for cluster members in cell ND840 on rchas840.

Step 8. Enable Session Persistence

Now that the applications have been installed we want to configure both cluster members to use persistent HTTP sessions (shopping cart) to the HA database located on rchas841. We have defined a data source “hawas” which can be used for the persistent session table. The data source already points to rchas841 and has the custom properties set as described above. We shall include the configuration screen shots for cluster member BASE839.

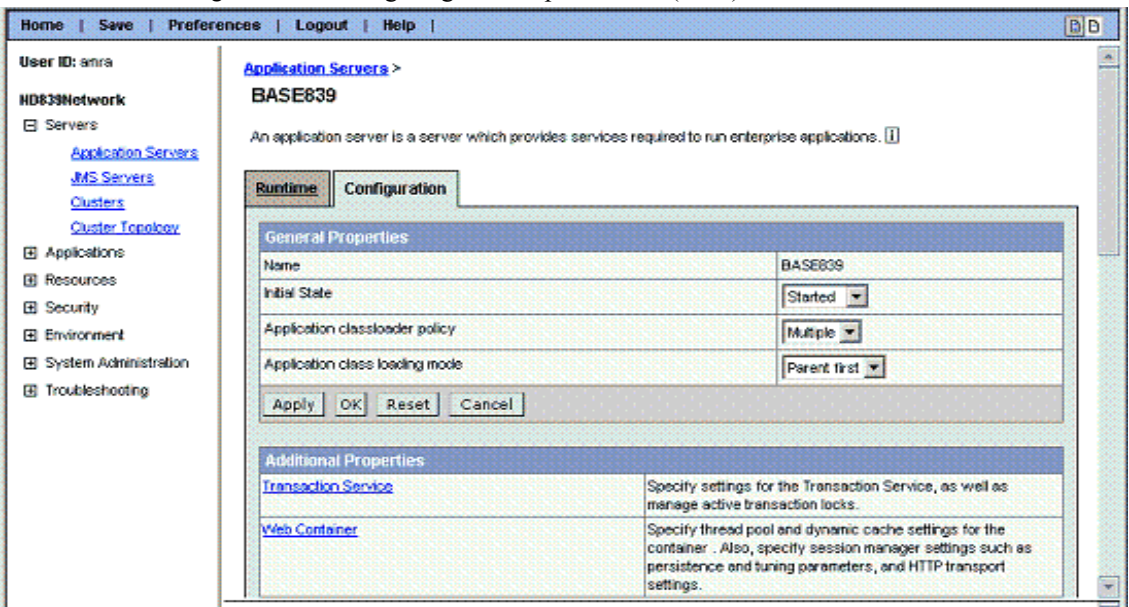
From the navigation tree on the left side of the administration console, click on **Servers** => **Application Servers**. You should see the application servers for cell ND839 as shown in Figure 18.30 on page 296. Click on cluster member **BASE839**.

Figure 18.30Configuring session persistence



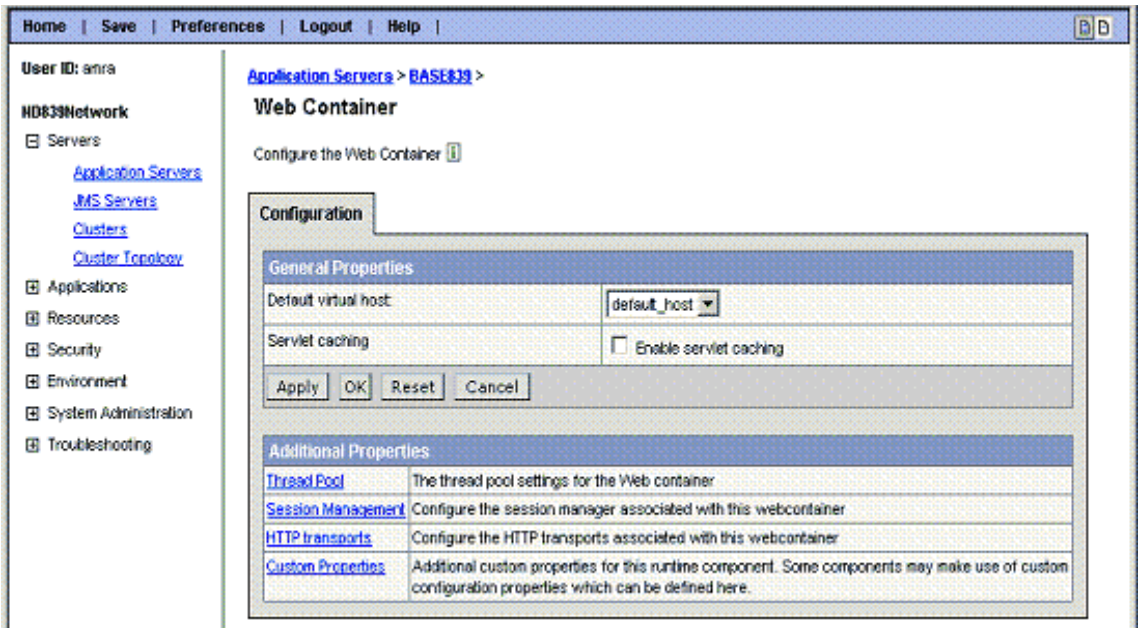
Click on **Web Container**.

Figure 18.31Configuring session persistence (cont.)



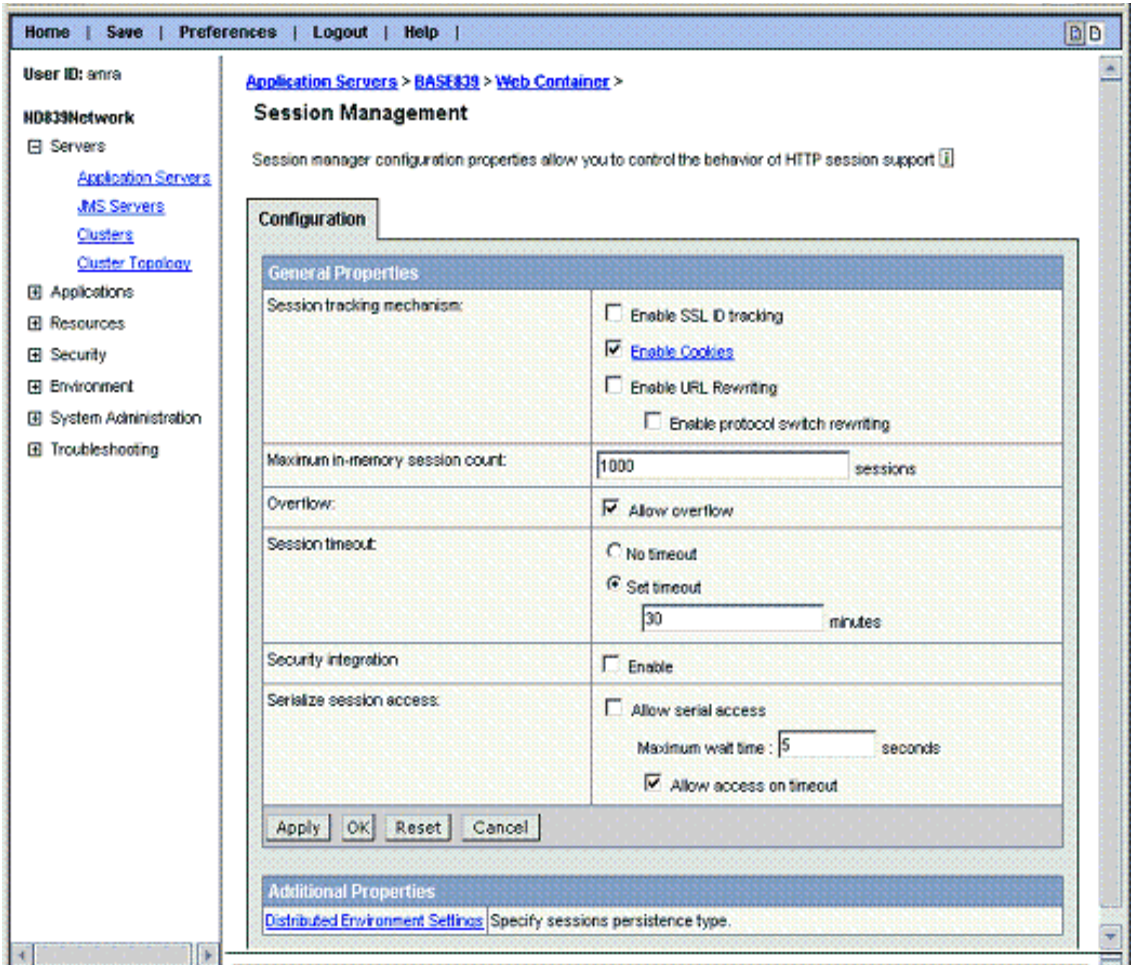
Click on **Session Management**.

Figure 18.32 Configuring session persistence (cont.)



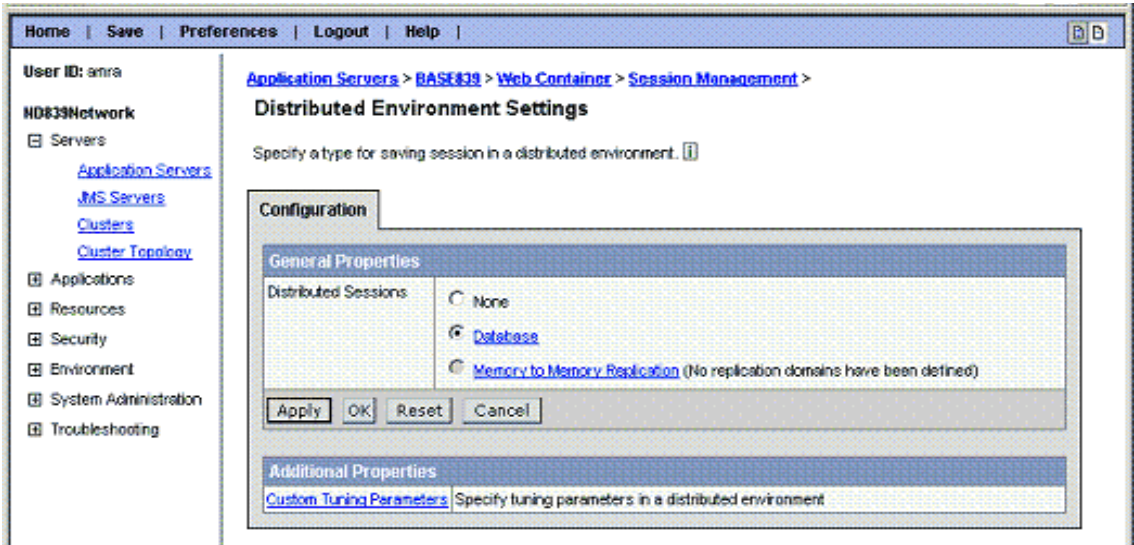
Ensure **Enable Cookies** is selected and accept other default values. Click on **Distributed Environment Settings**. See Figure 18.33 on page 298.

Figure 18.33 Configuring session persistence (cont.)



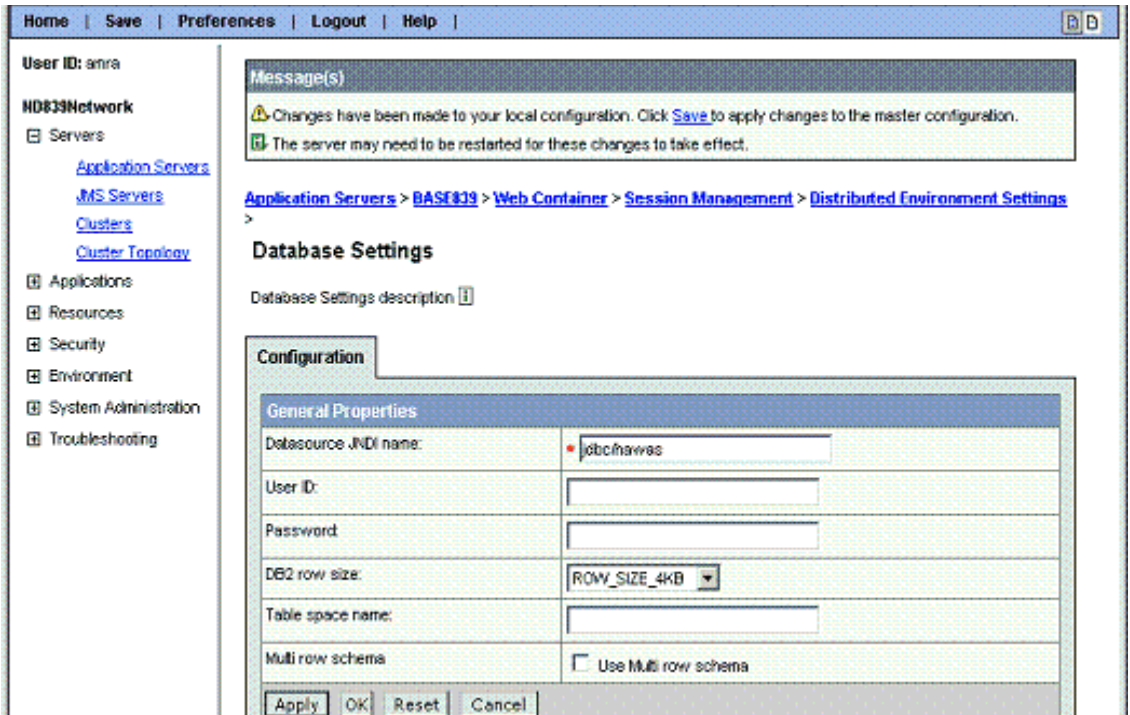
Select Database as the method to persist HTTP sessions and click on **Database**. See Figure 18.34 on page 299.

Figure 18.34 Configuring session persistence (cont.)



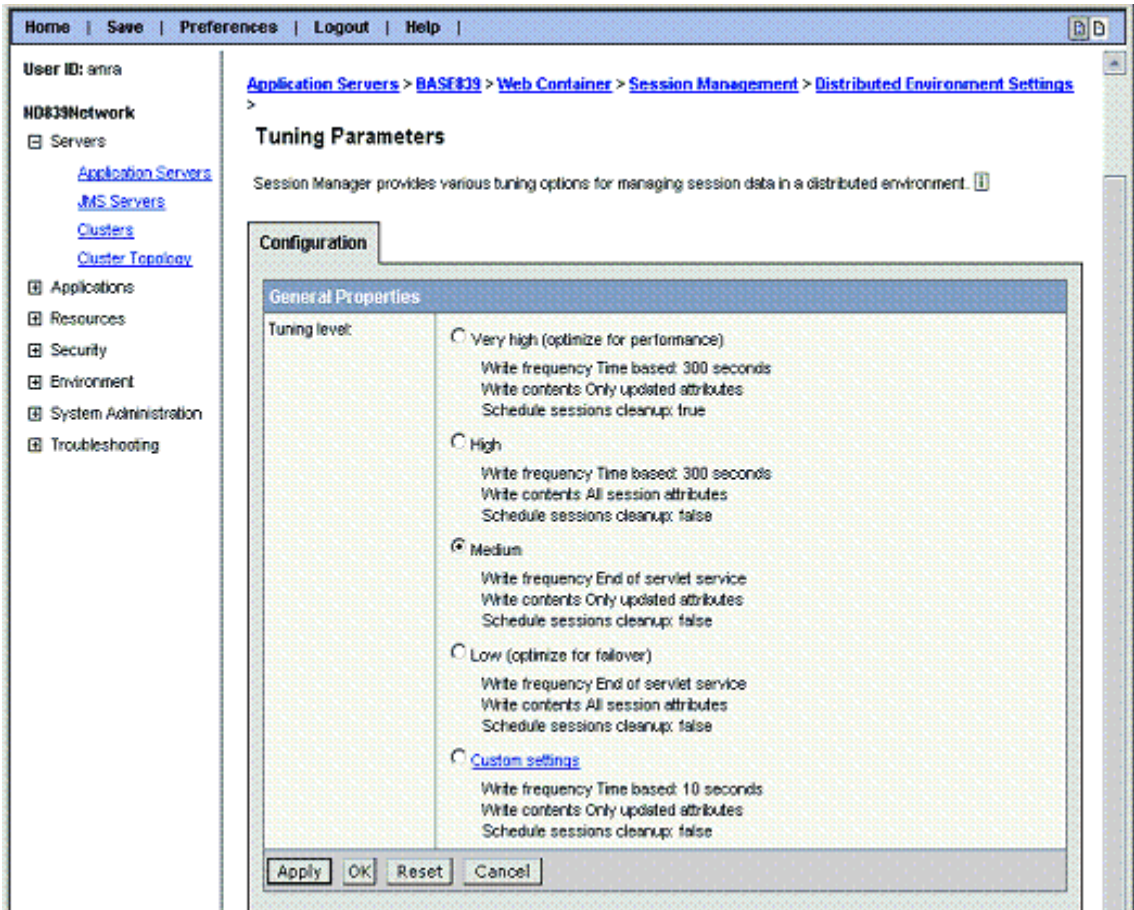
Specify hawas as the Datasource JNDI name to be used for persistence (see Figure 18.35 on page 299) and click **OK**. This will take you back to the Distributed Environment Settings panel shown in Figure 18.34 on page 299. Click on **Custom Tuning Parameters**.

Figure 18.35 Configuring session persistence (cont.)



The parameter selected here will determine how often the session data is written to the database. Select *Medium*. *Medium* assures in the case of failover the session data will be current. From a performance standpoint it is a better option than *Very High* or *High*. You must analyze your application to determine which option is best for your circumstance. Click **OK** and save the changes to the master configuration as we did before, ensuring that the *Synchronize changes with Nodes* is checked prior to performing the save.

Figure 18.36 Configuring session persistence (cont.)



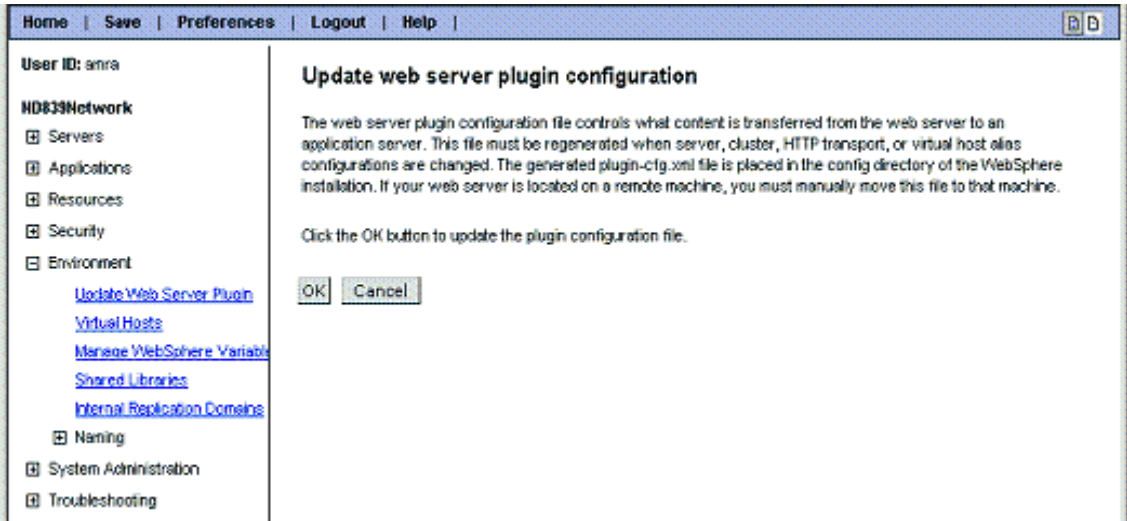
You must stop and start the cluster member in order to activate the change. Once started, a SQL collection name *hawas* will be created on *rchas841* with a table named *Sessions*. You will want to copy this library to *rchas842* before the HABP configures data replication. Do the same sequence of steps for cluster member *CLONE839* and cluster members in cell *ND840* on *rchas840*.

Step 9. Generate Plug-in File

Next we want to create the HTTP plug-in which is the link between the HTTP server and the cluster members contained within the cell *ND839*.

From the navigation tree on the left side of the administration console, click on **Environment => Update Web Server Plugin**. Click on **OK** to regenerate the plug-in.

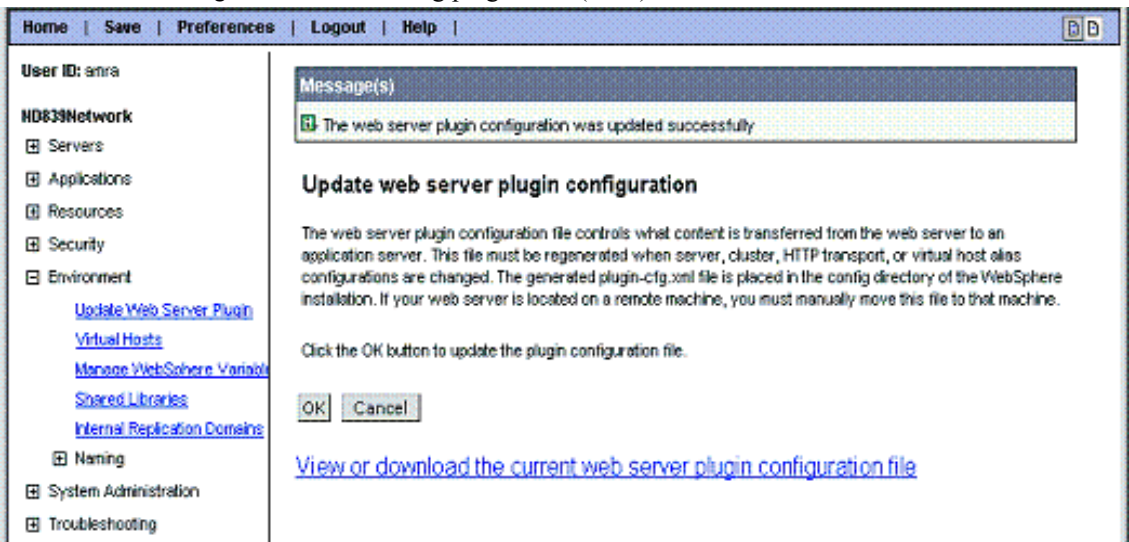
Figure 18.37Generating plug-in file



You may view the regenerated plug-in with the administration console, or you may view it in whole by navigating to:

`/QIBM/UserData/WebAS5/ND/ND839/config/cells/plugin-cfg.xml`

Figure 18.38Generating plug-in file (cont.)



The topology diagram indicates that the HTTP server will reside on a different partition than the application server. There is a term for this – Remote HTTP. In practice many customers run the Web and the base cluster member in the same LPAR partition. For completeness here we shall cover what should be done in both cases.

NOTE You will not be able to use the GenPluginCfg script to generate the plugin-cfg.xml file for the remote HTTP instance. Instead you will need to take the plugin-cfg.xml generated by the Network Deployment Manager and move that back to the partition containing the HTTP server instance.

When running in the same partition simply move the Network deployment level plug-in to one of the federated base directories. For example move /QIBM/UserData/WebAS5/ND/ND839/config/cells/plugin-cfg.xml. to directory /QIBM/UserData/WebAS5/Base/BASE839/config/cells/plugin-cfg.xml. Do not point your Web server at the /QIBM/UserData/WebAS5/ND/ND839/config/cells/plugin-cfg.xml since the deployment manager will periodically overlay this and would negate changes we shall make.

When running remote HTTP follow the online instructions, and instead of using GenPluginCfg just move the ND plug-in at /QIBM/UserData/WebAS5/ND/ND839/config/cells/plugin-cfg.xml to the Web instance created.

A copy of the plug-cfg.xml generated with the Network Deployment admin console has been captured and is included below. There are a few changes which are necessary. The changes can be made by editing the plug-cfg.xml file using the EDTF (Edit File) CL command. Pertinent information and changes required are highlighted in bold. A discussion of the changes follows.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Config ASDisableNagle="false" IISDisableNagle="false"
  IgnoreDNSFailures="false" RefreshInterval="60" ResponseChunkSize="64">
  <Log LogLevel="Error" Name="/QIBM/UserData/WebAS5/ND/ND839/logs/http_plugin.log"/>
  <Property Name="ESIEnable" Value="true"/>
  <Property Name="ESIMaxCacheSize" Value="1024"/>
  <Property Name="ESIInvalidationMonitor" Value="false"/>
  <VirtualHostGroup Name="default_host">
    <VirtualHost Name="*:9080"/>
    <VirtualHost Name="*:80"/>
  </VirtualHostGroup>
  <ServerCluster CloneSeparatorChange="false" LoadBalance="Round Robin" Name="cluster839"
    PostSizeLimit="1000000" RemoveSpecialHeaders="true" RetryInterval="60">
    <Server CloneID="v60aig0j" ConnectTimeout="0" ExtendedHandshake="false"
      LoadBalanceWeight="2" MaxConnections="0" Name="rchas839_BASE839_BASE839"
      WaitForContinue="false">
      <Transport Hostname="rchas839.RCHLAND.IBM.COM" Port="30009" Protocol="http"/>
    </Server>
    <Server CloneID="v60aiiv5" ConnectTimeout="0" ExtendedHandshake="false"
      LoadBalanceWeight="2" MaxConnections="0" Name="rchas840_CLONE839_CLONE839"
      WaitForContinue="false">
      <Transport Hostname="rchas840.RCHLAND.IBM.COM" Port="30010" Protocol="http"/>
    </Server>
    <PrimaryServers>
      <Server Name="rchas839_BASE839_BASE839"/>
      <Server Name="rchas840_CLONE839_CLONE839"/>
    </PrimaryServers>
  </ServerCluster>
  <ServerCluster CloneSeparatorChange="false"
    LoadBalance="Round Robin" Name="ND839_ND839Manager_Cluster"
    PostSizeLimit="1000000" RemoveSpecialHeaders="true" RetryInterval="60">
    <Server ConnectTimeout="0" ExtendedHandshake="false"
```

```

        MaxConnections="0" Name="ND839Manager_ND839" WaitForContinue="false"/>
    <PrimaryServers>
        <Server Name="ND839Manager_ND839"/>
    </PrimaryServers>
</ServerCluster>
<UriGroup Name="default_host_cluster839_URIs">
    <Uri AffinityCookie="JSESSIONID"
        AffinityURLIdentifier="jsessionid" Name="/Darin/*"/>
    <Uri AffinityCookie="JSESSIONID"
        AffinityURLIdentifier="jsessionid" Name="/Steve/*"/>
</UriGroup>
<Route ServerCluster="cluster839"
    UriGroup="default_host_cluster839_URIs" VirtualHostGroup="default_host"/>
<RequestMetrics armEnabled="false" newBehavior="false"
    rmEnabled="false" traceLevel="HOPS">
    <filters enable="false" type="URI">
        <filterValues enable="false" value="/servlet/snoop"/>
        <filterValues enable="false" value="/webapp/examples/HitCount"/>
    </filters>
    <filters enable="false" type="SOURCE_IP">
        <filterValues enable="false" value="255.255.255.255"/>
        <filterValues enable="false" value="254.254.254.254"/>
    </filters>
</RequestMetrics>
</Config>

```

The following information illustrates the changes necessary and also the reasoning for them.

```
<Log LogLevel="Error" Name="/QIBM/UserData/WebAS5/Base/BASE839/logs/http_plugin.log"/>
```

In the case where the Web server and cluster members are on the same system, the plugin-cfg.xml has been moved back to BASE839 directory and so path to the http log file must also be changed to reflect this.

For Remote HTTP modify this directory according to path where the plugin-cfg.xml is stored.

```
<ServerCluster CloneSeparatorChange="false" LoadBalance="Round Robin" Name="cluster839"
PostSizeLimit="1000000" RemoveSpecialHeaders="true" RetryInterval="1800">
```

The xml above defines attributes which are cluster scoped. Cluster scoped here denotes activated for all cluster members contained within cluster “cluster839”. What is RetryInterval? Once a cluster member is marked down, how long is optimal for the HTTP plug-in to wait before retrying the cluster member again? The amount of time before the plug-in will retry is configured by the RetryInterval attribute. The default value is 60 seconds which is just too small. Setting the RetryInterval to a small value will allow a cluster member, which becomes available, to quickly begin servicing requests. However, too small of a value can cause serious performance degradation, or even cause your plug-in to appear to stop serving requests. There is little point in setting this at a level so short that a cluster member is not likely to be revived in that time period. Many client requests could queue up in a short period of time with no chance of success. To avoid this problem, we recommend to set a more conservative RetryInterval, the 60 second default is just too short. In the example above we set it to 1800 seconds. You will need to decide what is appropriate for your business.

```
<Server CloneID="v60aig0j" ConnectTimeout="60" ExtendedHandshake="false"
LoadBalanceWeight="2"
MaxConnections="0" Name="rchas839_BASE839_BASE839" WaitForContinue="false">
  <Transport Hostname="rchas839.RCHLAND.IBM.COM" Port="30009" Protocol="http"/>
</Server>
```

The “ConnectTimeout” parameter controls how long the HTTP plug-in instance waits when attempting to establish a TCP connection with a cluster member. If the connection has not been made before this time-out expires, the plug-in marks the cluster member as ‘down’ and tries another one. This parameter is one of the principal determinants of the time taken for user requests to be switched from an inaccessible cluster member to another cluster member in the cluster. Reducing this time-out value reduces the delay in responding to users during a failover incident. Absence of this parameter means that the system default TCP connection time-out applies. On OS/400 the connection time-out for a blocking connect is two minutes. This may be too long for some customer environments and if it is, then the ConnectTimeout attribute is a method to resolve this. We recommend setting ConnectTimeout as low as possible without causing ‘false negatives’. In this case, false negatives are situations where the worst case for the time taken for a cluster member to respond to a connection request (that is, taking account of peak loads) is less than the ConnectTimeout setting. This leads plug-ins to mark cluster members down when they are not. Set ConnectTimeout high enough to avoid false negatives instead of setting RetryInterval artificially small. This may take some thought and experimentation to come up with the right value. In the example above we set the value to 60 seconds for cluster member BASE839.

```
<Server CloneID="v60aiiv5" ConnectTimeout="60"
ExtendedHandshake="false" LoadBalanceWeight="2"
MaxConnections="0" Name="rchas840_CLONE839_CLONE839" WaitForContinue="false">
  <Transport Hostname="rchas840.RCHLAND.IBM.COM" Port="30010" Protocol="http"/>
</Server>
```

Set ConnectTimeout attribute for cluster member CLONE839 to 60 seconds.

Within each cell one may run multiple HTTP servers for load balancing and failover purposes. Each one of these HTTP servers would use a copy of the modified plugin-cfg.xml. Also a Network Dispatcher would need to know about each of these Web servers.

Do the same sequence of steps for cell ND840 on rchas840.

Step 10. Update TCP/IP configuration values

Change four Configure TCP/IP configuration values on rchas839, rchas840, rchas841 and rchas842. This change will prevent hangs between the WebSphere Application server and the back-end database during database switch and failover. For more detailed explanation on why, see “” on page 178.

Use CHGTCPA command and change the following values on all four systems.

- TCP keep alive -- 2 minutes
- TCP R1 retransmission count – 3
- TCP R2 retransmission count – 9
- ARP cache timeout – 2

The command prompt for the CHGTCPA CL command is shown in Figure 18.39 on page 305 with the changed values (note there you need to scroll to the next page to set the ARP cache timeout value shown in Figure 18.40 on page 305):

Figure 18.39CHGTCPA command prompt - first screen

```

Change TCP/IP Attributes (CHGTCPA)

Type choices, press Enter.

TCP keep alive . . . . . 2 1-40320, *SAME, *DFT
TCP urgent pointer . . . . . *BSD *SAME, *BSD, *RFC
TCP receive buffer size . . . . . 8192 512-8388608, *SAME, *DFT
TCP send buffer size . . . . . 8192 512-8388608, *SAME, *DFT
TCP R1 retransmission count . . . . . 3 1-15, *SAME, *DFT
TCP R2 retransmission count . . . . . 9 2-16, *SAME, *DFT
TCP minimum retransmit time . . . . . 250 100-1000, *SAME, *DFT
TCP closed timewait timeout . . . . . 120 0-14400, *SAME, *DFT
TCP close connection message . . . . . *THRESHOLD *SAME, *THRESHOLD, *ALL...
UDP checksum . . . . . *YES *SAME, *YES, *NO
Path MTU discovery:
  Enablement . . . . . *YES *SAME, *DFT, *NO, *YES
  Interval . . . . . 10 5-40320, *ONCE
IP datagram forwarding . . . . . *NO *SAME, *YES, *NO
IP source routing . . . . . *YES *SAME, *YES, *NO
IP reassembly time-out . . . . . 10 5-120, *SAME, *DFT
More...

```

Figure 18.40CHGTCPA command prompt - next page

```

Change TCP/IP Attributes (CHGTCPA)

Type choices, press Enter.

IP time to live (hop limit) . . . . . 64 1-255, *SAME, *DFT
IP QoS enablement . . . . . *NO *SAME, *TOS, *YES, *NO
IP QoS datagram batching . . . . . *NORMAL *SAME, *NORMAL, *MINDELAY
IP QoS timer resolution . . . . . 100 5-5000, *SAME, *DFT
IP dead gateway detection:
  Enablement . . . . . *YES *SAME, *DFT, *NO, *YES
  Interval . . . . . 2 1-60
ARP cache timeout . . . . . 2 1-1440, *SAME, *DFT
Network file cache:
  Enablement . . . . . *YES *DFT, *CLEAR, *SAME, *YES, *NO
  Cached file timeout . . . . . 300 *NOMAX,30-604800 sec (1week)
  Cache size . . . . . 10 10-100000 megabytes
Log protocol errors . . . . . *NO *SAME, *YES, *NO

```

Only change the four values mentioned above.

CHAPTER 19

CONFIGURING TO MAKE DATA HIGHLY AVAILABLE

Making data residing on iSeries highly available may be accomplished through either replication or switched disk technologies. Replication solutions from High Availability Business Partner (HABP) companies can provide both local and remote site failover capabilities. Switched disk technologies in V5R2 can provide local failover capabilities when data is stored within one or more Independent ASPs. In addition, there are also storage area network (SAN) alternatives which involve IBM's Enterprise Storage Server (ESS).

This chapter will focus on the implementation details for an independent ASP (IASP) configuration to provide for the high availability of the database tier for an e-business application that was used in our internal testing environment. The terms 'independent disk pool' and 'independent auxiliary storage pool (ASP)' are synonymous, and they will be used interchangeably throughout this chapter. For more information on ESS alternatives or third-party database replication software solutions, please see Chapter 12, "Data Considerations for High Availability" on page 209.

Configuring Switched Disk (IASP)

An independent disk pool is a collection of disk units that can be brought online or taken offline independent of the rest of the storage on a system, including the system disk pool, traditional user disk pools, and other independent disk pools. An independent disk pool can be either switchable among multiple systems in a clustered environment or privately connected to a single system.

The additions to Independent ASPs in V5R2 provide several advantages. Library-capable objects such as database files and journal receivers can now be stored within an independent ASP. Up to 223 independent disk pools can be configured, including the ability to have disk pool groups. This gives one the capacity to have multiple databases for each iSeries system. But, database objects that are tightly coupled to other objects must be placed in the same IASP since a collection cannot span across IASP boundaries.

Applications and data may be switched between partitions or systems in the cluster to minimize downtime associated with both planned and unplanned outages. Even in an environment that does not utilize switched disks, using private IASPs on a single system can improve the overall availability of that system and its associated applications since the rest of the system may continue to run despite failures in one or more IASPs since each IASP is independent from the rest of the system. Lastly, IASPs may be ideal for some server consolidation scenarios since the same libraries can exist in multiple IASPs on the same OS/400 partition. This can be especially beneficial to those who provide application hosting services.

Since the goal for this particular internal test environment was to configure an IASP for maximum availability, a switched disk environment – versus just a private IASP – was the appropriate choice. Secondly, we decided to use just a primary IASPIASP as opposed to an IASP group with both a primary and secondary IASP for the following two reasons: simplification for illustrative purposes and because the WebSphere Commerce product normally places both the database files and the journal receivers into the same library. However, for most enterprise databases, it would be best to place the database files and journal port into a primary IASP and then to create a secondary IASP to contain the associated journal receivers within a separate library. This provides extra data isolation characteristics so that one does not lose both the database files and journal receivers in the event of a disk failure that is not recoverable. Please remember to configure either RAID protection or disk mirroring for the system ASP as well as any IASPs that store mission-critical application data.

Our emphasis in this chapter is on the usage of IASPs to achieve a higher-level of availability for just the database tier of mission-critical applications. In other words, this is an example of a typical distributed application environment in which the application programs and processes running within WebSphere are made highly available through various software clustering technologies provided by WebSphere middleware. But to achieve high availability for some other key components in the application path, one must use other means. OS/400 clustering provides an IP takeover addressing capability that works well with WebSphere data source definitions that utilize IP names or addresses for database access. One can easily designate a takeover address for the switchable hardware associated with an independent ASP that contains a database collection. Therefore, independent ASPs can provide a mechanism to help you meet your high-availability objectives when databases are in the application path.

The following high-level steps were taken to configure a switched disk environment for a remote database tier that is used by our WebSphere test application:

- 1** System hardware and software requirements planning
- 2** Cabling to complete an HSL loop containing the switched disk tower and the two nodes that will comprise the cluster
- 3** Through iSeries Navigator interfaces, I/O resource ownership changes needed to be done since secondary partitions were part of our cluster. Our database servers happened to reside on those secondary partitions.
- 4** Through Management Central options within iSeries Navigator, cluster creation was performed involving two secondary partitions.
- 5** Enablement of the disk tower to be switchable through iSeries Navigator
- 6** Creation of a switchable hardware group with a new disk pool (i.e. an IASP) called 'HADB' through iSeries Navigator
- 7** Migration of an enterprise database collection to the IASP

- 8 Creation of a library called 'HAWAS' in the IASP to be used for persistence of WebSphere servlet session data
- 9 Modification or creation of the applicable JDBC data source definitions via the WebSphere administrative console to allow access to both the enterprise database (INVEST collection) and WebSphere session data collections (HAWAS collection) that are stored within the IASP.
- 10 Switchover and failover tests to validate that the final IASP configuration provided the expected high-availability characteristics.

Step 1. System Hardware and Software Requirements Planning

You must have the following hardware to configure library-capable IASPs that are switchable in a multi-system clustered environment:

- 1 Two or more iSeries servers capable of running OS/400 V5R2 or one iSeries server capable of running OS/400 V5R2 configured with logical partitions (LPAR) and
- 2 One or more switchable devices. This could be:
 - One or more expansion units (towers) residing on an HSL loop
 - One or more input/output processors (IOP) in a logical partition

NOTE In an LPAR environment, you can switch the input/output processor (IOP) containing the independent disk pools between system partitions without having an expansion unit. The IOP must be on the bus shared by multiple partitions. All input/output adapters (IOAs) on the IOP will be switched.

In addition, you must satisfy the following physical planning requirements:

- High Speed Link (HSL) cables must be used to attach the expansion units (towers) to the servers in the cluster.
- The expansion unit must be physically adjacent in the HSL loop to the alternate system or expansion unit owned by the alternate system. You can include a maximum of two servers (cluster nodes) on each HSL loop, though each server can be connected to multiple HSL loops. You can include a maximum of four expansion units on each HSL loop, though a maximum of three expansion units can be included on each loop segment. On an HSL loop containing two servers, two segments exist, separated by the two servers. All expansions on one loop segment must be contained in the same device CRG.

The switchable expansion unit must be SPCN-cabled to the system unit that will serve as the primary node for the switchable hardware group (device CRG). The primary node could be a primary or secondary logical partition within the system unit. If using LPAR, the system buses in the intended tower must be owned dedicated by the partition involved in the cluster.

Depending upon how you plan to implement independent disk pools for library-capable objects, you must have the following software and licenses:

- OS/400 V5R2
- iSeries Navigator

iSeries Navigator is the graphical user interface for managing and administering your iSeries server from your Windows desktop. It is required to perform some of the disk

management tasks necessary to implement independent disk pools. The profile you use for Navigator access to IASP functions must have *IOSYSCFG and *ALLOBJ authorities. Before you can access disk management functions in iSeries Navigator, you must complete the following steps:

- 1 Install the Configuration and Service component
 - a From the File menu of iSeries Navigator select **Install Options => Selective Setup**.
 - b Follow the instructions on the resulting dialog to install the Configuration and Service component.
- 2 Enable the Disk Units folder
 - a In iSeries Navigator right-click the server connection and select **Application Administration**.
 - b On the resulting window, click **OK**.
 - c Click the **Host Applications** tab.
 - d Expand **Operating System/400 => Service**.
 - e Select **Disk Units** to have Default Access or All Object Access.
 - f Click **OK**.
 - g Restart iSeries Navigator.

The service tools server can be configured to be available when the server has been powered on to DST. If you use only the Operations Console with LAN connectivity to perform DST activities, the service tools server does not need to be reconfigured, as it is already available to you when the server has been powered on to DST.

You can enable the service tools server through DST by dedicating a network interface card to the service tools server. To enable the service tools server with its own network interface card, complete the following steps:

- 1 From the Use dedicated service tools (DST) display, select option 5 (**Work with DST environment**) and press **Enter**. The Work with DST Environment display appears.
- 2 From the Work with DST Environment display, select option 2 (**System devices**) and press **Enter**. The Work with System Devices display appears.
- 3 From the Work with System Devices display, select option 6 (**Console mode**) and press **Enter**. The Select Console Type display appears.
- 4 From the Select Console Type display, press **F11** (Configure). The Configure Service Tools Adapter display appears.
- 5 From the Configure Service Tools Adapter display, enter the LAN Adapter and TCP/IP information. Press **F1** (Help) for the type of information required in each field.
- 6 Press **F7** (Store) to save your changes.
- 7 Press **F14** (Activate) to activate the adapter.

The service tools server is ready to use with a valid service tools user ID.

You must add the service tools server to the service table in order to access service tools on OS/400 using TCP/IP and iSeries Navigator. The service tools server can be added prior to configuring your local area network (LAN). To add the service tools server to the service table, complete the following steps:

- 1 From any command line, type ADDSRVTBLE (Add Service Table Entry) and press **Enter**. The Add Service Table Entry display appears.
- 2 Enter the following information in the fields provided:
 - Service: as-sts
 - Port: 3000
 - Protocol: 'tcp' (this entry must appear lowercase and in single quotes)
 - Text description: 'Service Tools Server'

This field is optional, but you are strongly recommended to enter a description of the table entry.
- 3 Press **F10** (Additional Parameters).
- 4 Enter AS-STs in the Alias field. The Alias must be capitalized because some table searches are case-sensitive.
- 5 Press **Enter** to add the table entry.
- 6 TCP/IP must be ended and restarted for the service table entry to be use. If you cannot end TCP at this time, you will not be able to use the service tools server. Enter ENDTCP (End TCP) to end TCP/IP if this is possible in your environment.
- 7 Enter STRTCP (Start TCP). Verify that the service tools server is listening to port 3000 by entering NETSTAT OPTION(*CNN) from a 5250 session. Look for as-sts under the heading Local Port with a State value of Listen.

If you will be using iSeries Navigator to perform disk unit or logical partition configuration and management, you need to complete the following steps once per server:

- 1 From an iSeries Navigator session, right-click the server name under **My Connections** (for your environment you may use your own name for the connections function instead of the default My Connections).
- 2 Select **Application Administration**. Press **OK** until you get to a window with a **Host Applications** tab. Select the **Host Applications** tab, expand **Operating System/400**, and expand **Service**.
- 3 Select any of the service tools that you want to authorize: Disk Units, QIBM_QYTP_SERVICE_LPARGMT, or Service Trace. You can select more than one.
- 4 Press **OK**. These functions are now available to the iSeries Navigator user provided they have a service tools user ID.

Once the service tools server has been added to the service table, authorized users can access the logical partition (LPAR) and disk management service functions using iSeries Navigator and TCP/IP. Note that, as with all service tools user IDs, you can selectively grant or restrict a user to specific service functions using functional privileges.

Option 41 (OS/400 - Switchable Resources) must be installed. The option is a clustering requirement that gives you the capability to switch independent disk pools between systems. In order to switch an independent disk pool between servers, the servers must be members of a cluster and the independent disk pool must be associated with a switchable hardware group in that cluster. Option 41 also gives you the capability of using the IBM Simple Cluster Management interface in iSeries Navigator to define and manage a simple cluster that uses switchable resources.

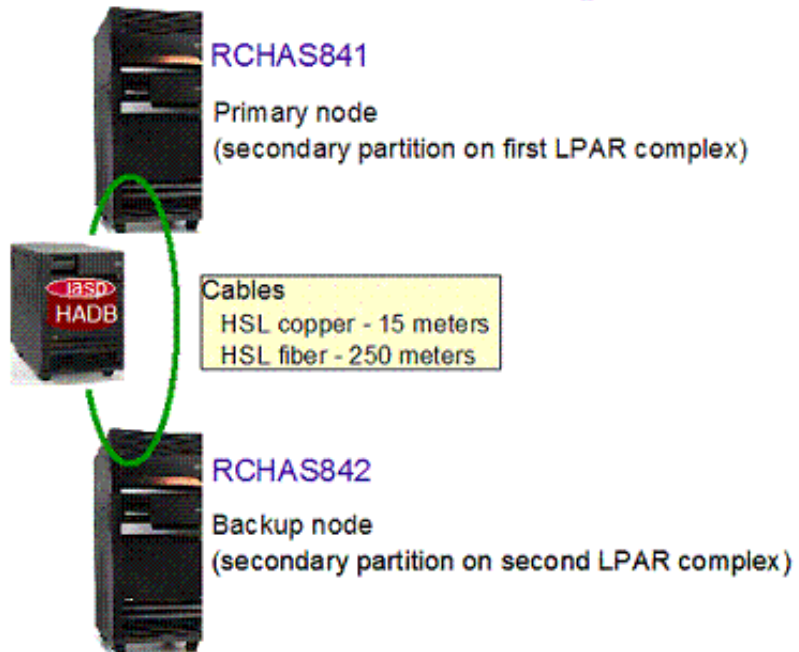
Finally, you must satisfy the following communications requirements: At least one TCP/IP communication interface between the servers in the cluster. For redundancy, it is recommended that there be two separate interfaces between the servers.

NOTE: It is not required that the HSL OptiConnect Loop interface between servers be used in a switchable expansion unit (tower) configuration. Also, it is not required that Virtual OptiConnect communication between LPAR partitions be used in a switchable IOP in a logical partition environment.

Step 2. Physical Cabling of HSL Loop

For our simple two-node cluster, a total of three HSL copper cables were utilized to complete the loop between the single switchable tower and the two physical iSeries servers that hosted our enterprise databases on secondary partitions RCHAS841 and RCHAS842. One LPAR complex hosted primary partition RCHAS837 and secondary partitions RCHAS839 and RCHAS841. The second LPAR complex hosted primary partition RCHAS838 and secondary partitions RCHAS840 and RCHAS842. Please refer to the diagram below:

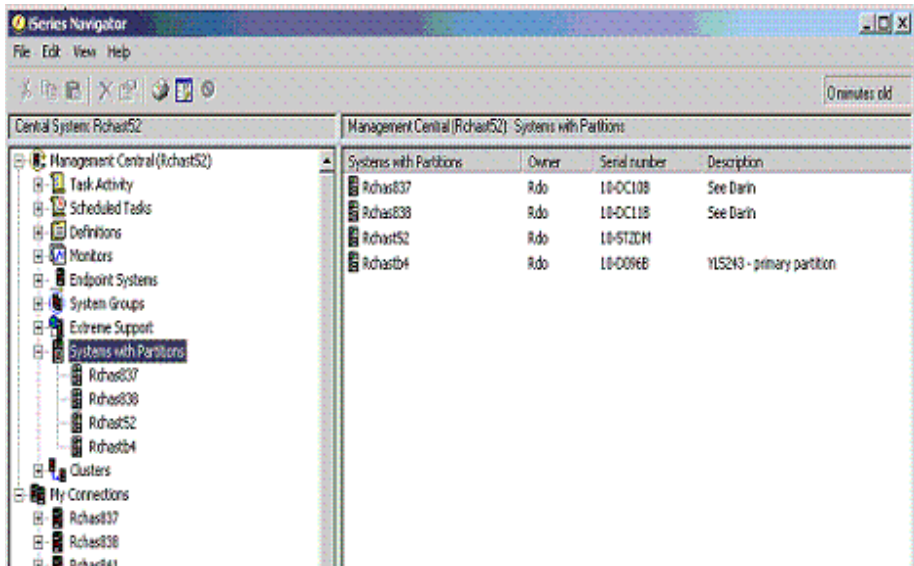
Figure 19.1 IASP hardware configuration



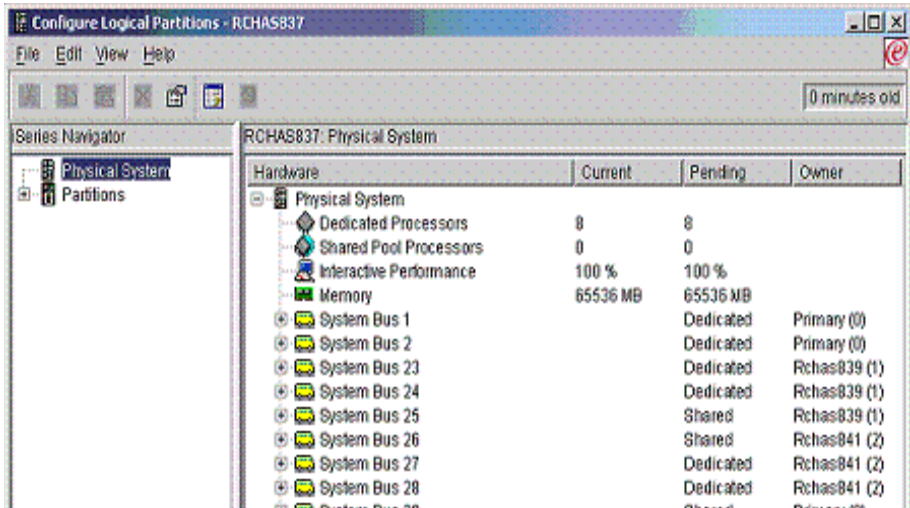
Step 3. I/O Change Resource Ownership for LPAR

Our database servers happened to reside on secondary logical partitions RCHAS841 and RCHAS842. There was shared ownership of the I/O resources with system buses 29 and 30 for the switchable tower we used in our configuration. Therefore, ownership of the switchable tower resources needed to be changed to dedicated ownership for secondary partition RCHAS841 since that is a requirement for switchable disk pool configurations. The following are the iSeries Navigator screen shots for that configuration change.

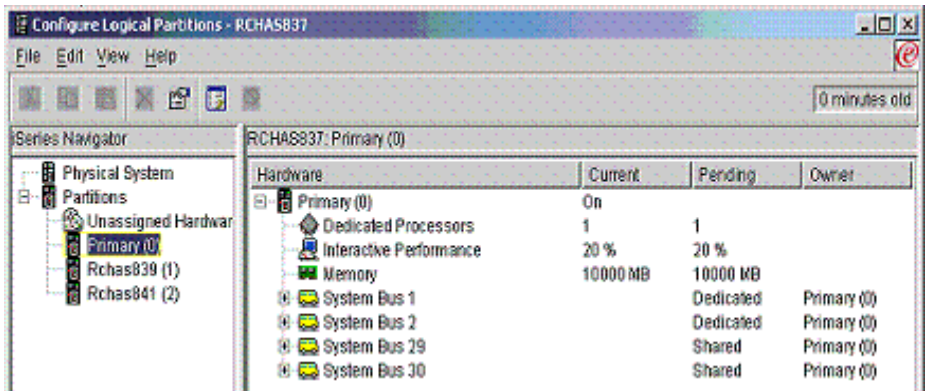
We expanded the **Systems with Partitions** category to locate primary partition RCHAS837.



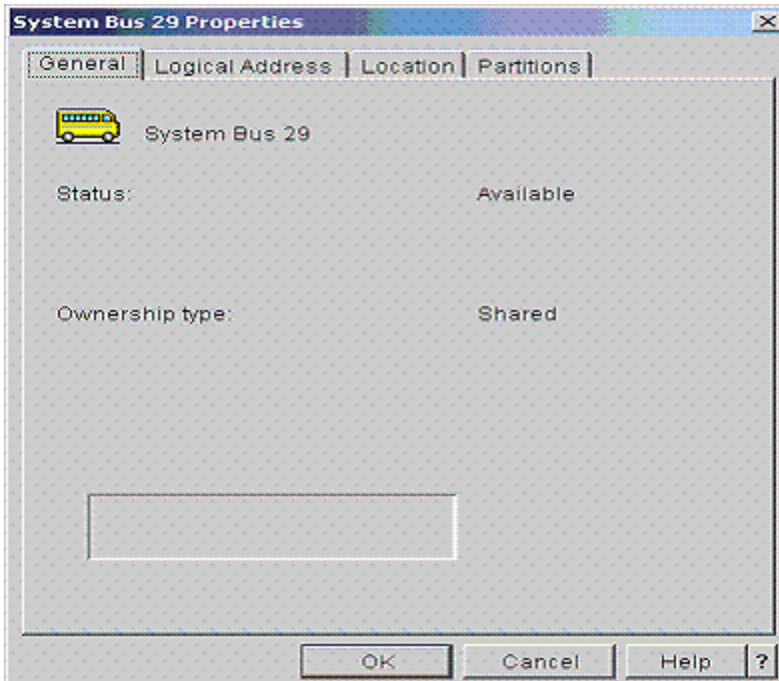
We right-clicked on Rchast837 and then selected **Configure Partitions** to display all the physical resources associated with the system.



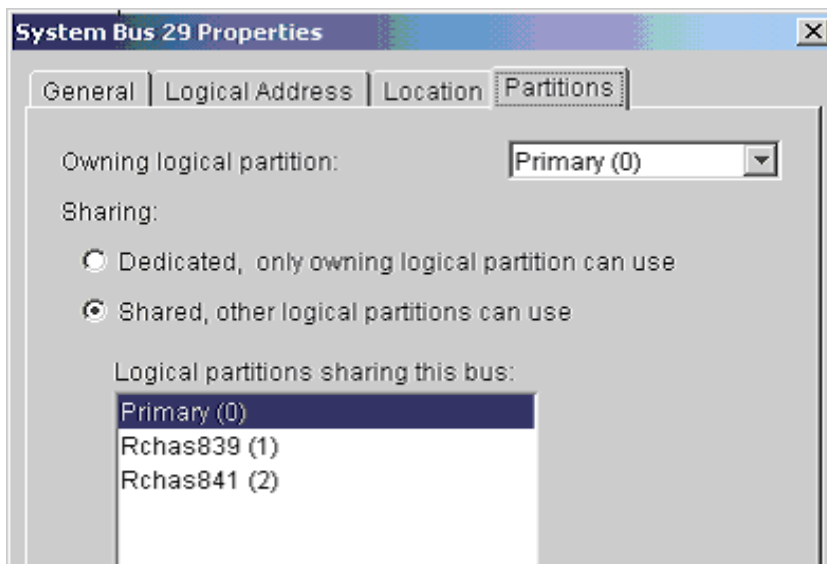
We expanded the information under **Partitions** and then selected **Primary(0)** to work with the shared buses.



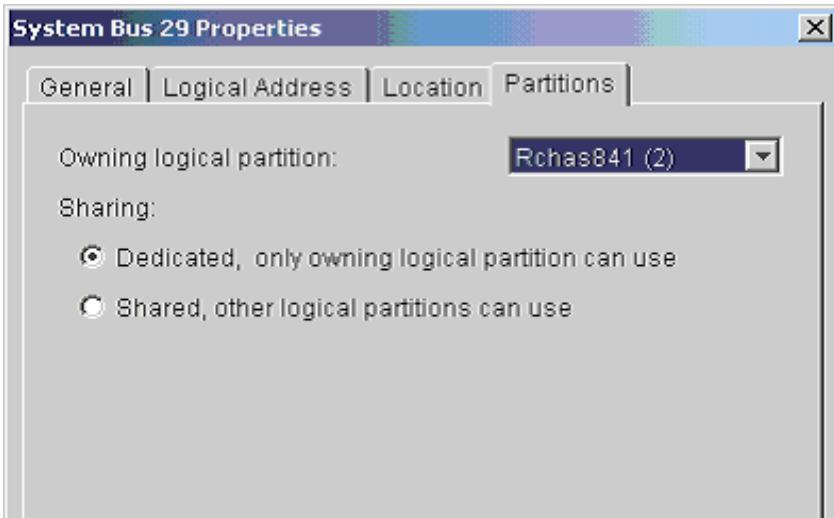
We clicked on the Properties context menu option for **System Bus 29** in the right pane to bring up the following screen:



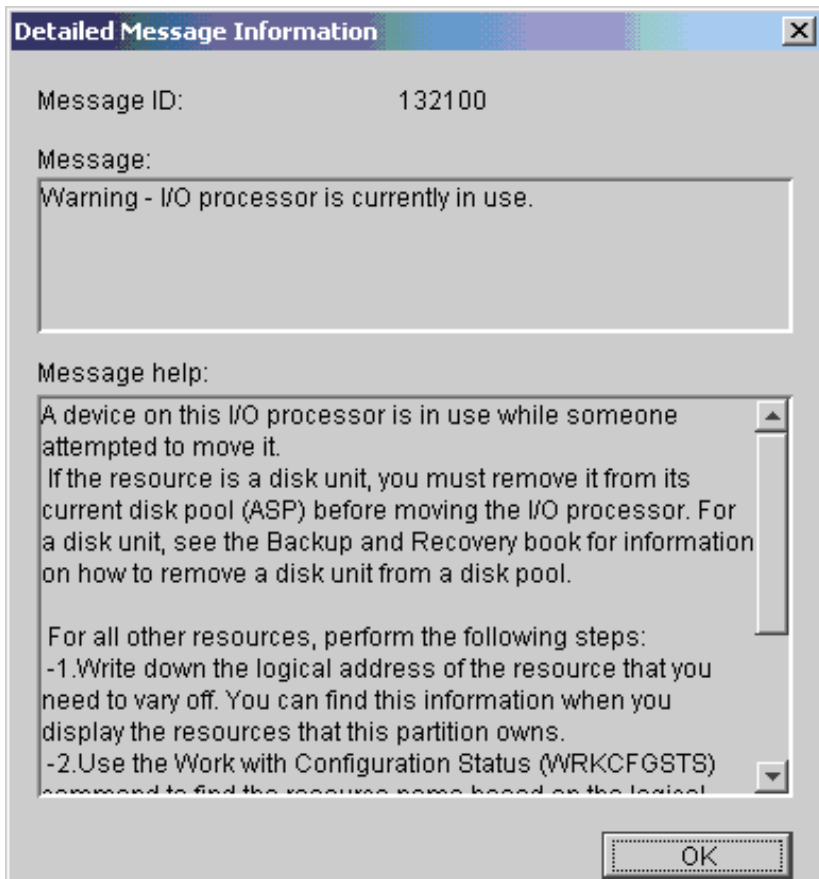
Then, we clicked on the **Partitions** tab to display the current information for system bus 29:



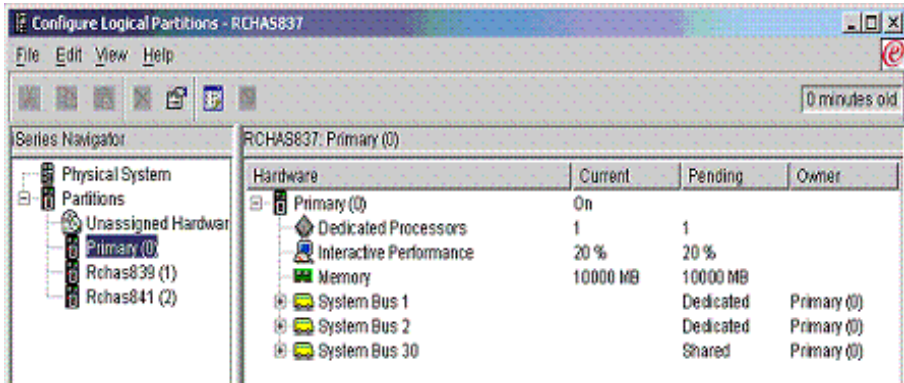
The **Dedicated** option was chosen and we selected RCHAS841 as the owning logical partition:



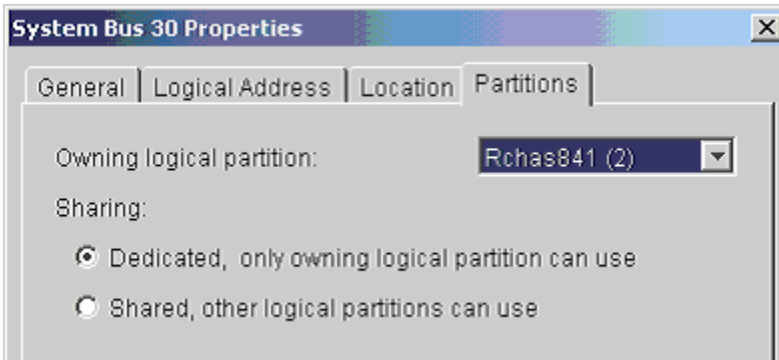
The following warning message may appear:



Since the disk units on system bus 29 were not part of any existing disk pool, we chose to click **OK**.



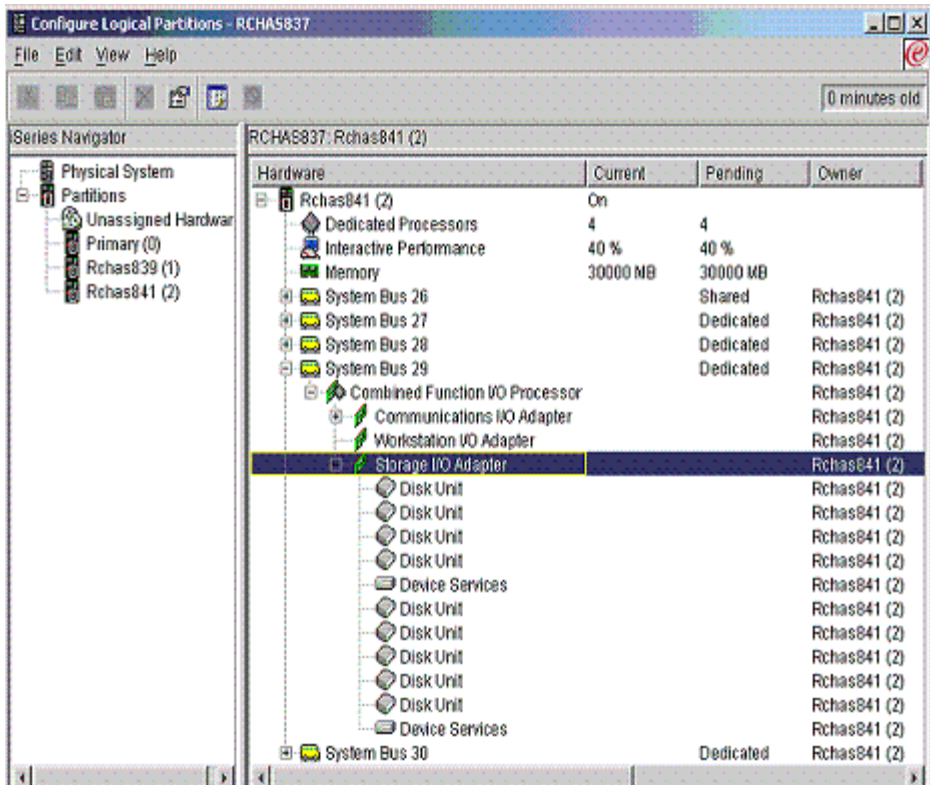
Notice in on page 318 that system bus 29 no longer appears under the primary partition. Next, click on the Properties context menu option for **System Bus 30** in the right pane to bring up the following screen:



We chose the Dedicated option and selected RCHAS841 as the owning logical partition.

Notice in on page 319 that system bus 30 is now owned in dedicated mode by secondary partition rchas841.

In order to view the disk units associated with system bus 29, we expanded the + sign to the left of **System Bus 29** in the right pane.



Step 4. Create the OS/400 Cluster

The next step was to create a two-node cluster involving systems RCHAS841 and RCHAS842.

NOTE Although we create the cluster using CL commands, you can also create the cluster by using the iSeries Navigator.

TIP The 5250 command GO CMDCLU provides a list of HA clustering commands that are available.

- 1 Ensure that clustering is enabled for both iSeries servers by issuing the Change Network Attributes (CHGNETA) command on both systems:

```
CHGNETA ALWADDCLU(*ANY)
```

- 2 Make sure that the Internet Daemon (INETD) server is started on both iSeries servers by issuing the Start TCP/IP Server (STRTCPSVR) command on both systems:

```
STRTCPSVR *INETD
```

If it is already started, you will get an error message.

TIP Because INETD is needed for the proper operation of HA clustering on your iSeries servers, we recommend that you change the properties of the INETD to Start when TCP/IP is started.

- 3** Make sure that HA processes used by OS/400 clustering can run unimpeded in the QBATCH subsystem on both iSeries servers. OS/400's HA clustering uses a batch job to slow poll the primary server to determine if this system is still available to the network. By default, this batch job is started in QBATCH. You must ensure that the job queue for QBATCH is large enough to always allow this job to run. You can use the Change Job Queue Entry (CHGJOBQE) command to change the maximum active to no maximum:

```
CHGJOBQE SBSD(QBATCH) JOBQ(QBATCH) MAXACT(*NOMAX)
```

- 4** On iSeries server RCHAS841, use the Create Cluster (CRTCLU) command to create the cluster COMMERCE that includes node RCHAS841.

```
CRTCLU CLUSTER(COMMERCE) NODE((RCHAS841 ('9.5.2.170')))
```

The CRTCLU command should complete without errors.

- 5** Create the second cluster node RCHAS842 by entering the Add Cluster Node Entry (ADDCLUNODE) command (on iSeries server RCHAS841):

```
ADDCLUNODE CLUSTER(COMMERCE) NODE(RCHAS842 ('9.5.8.152'))
```

The ADDCLUNODE command should complete without errors.

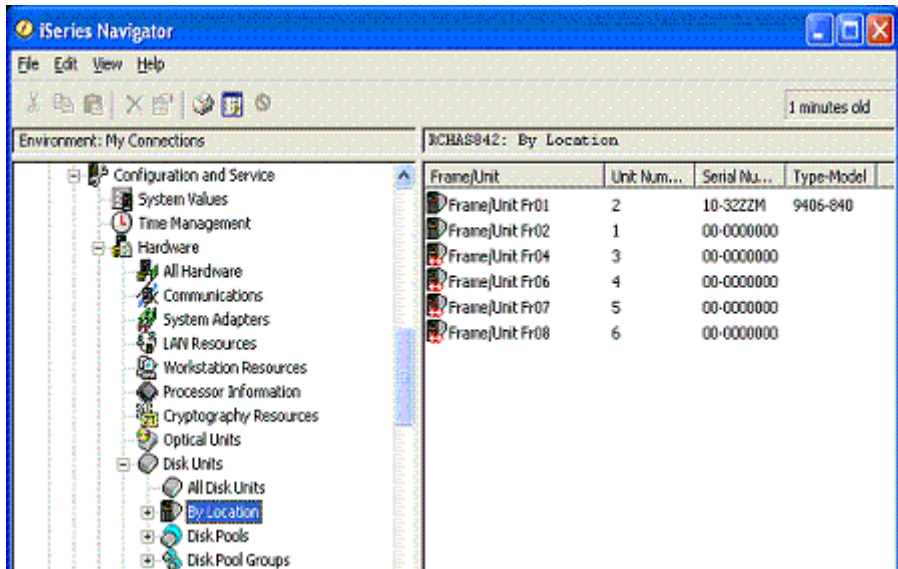
- 6** To see the status of nodes RCHAS842 and RCHAS841 in cluster COMMERCE, enter the Display Cluster Information (DSPCLUINF) command:

```
DSPCLUINF CLUSTER(COMMERCE)
```

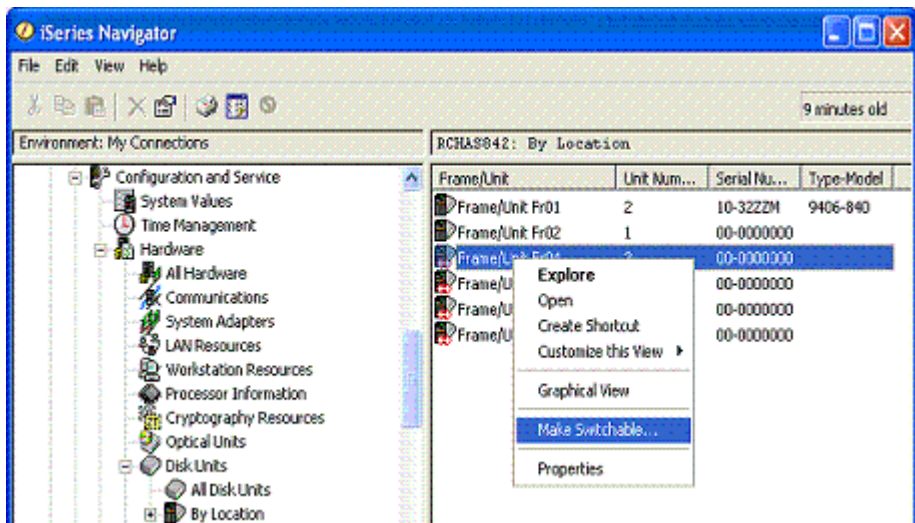
The status for both nodes should be Active.

Step 5. Enable the Tower to be Switchable

Within iSeries Navigator, we expanded the initial owning system under **My Connections**. Next, we expanded **Configuration and Service => Hardware, Disk Units**. When the Service Device Sign-on window appears, we signed on with the DST/SST user ID and password.



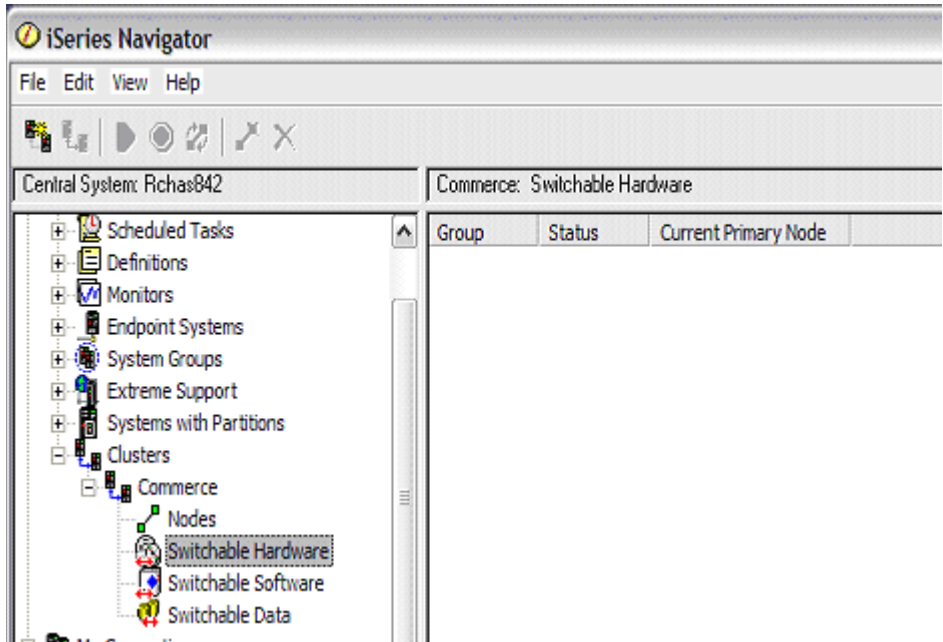
Next, we expanded **By Location** and right-clicked the desired tower in order to select the **Make Switchable** option.



We clicked **OK** within the confirmation window to complete the processing for that step. Note that a red double-arrow designation indicates that a tower has been made switchable.

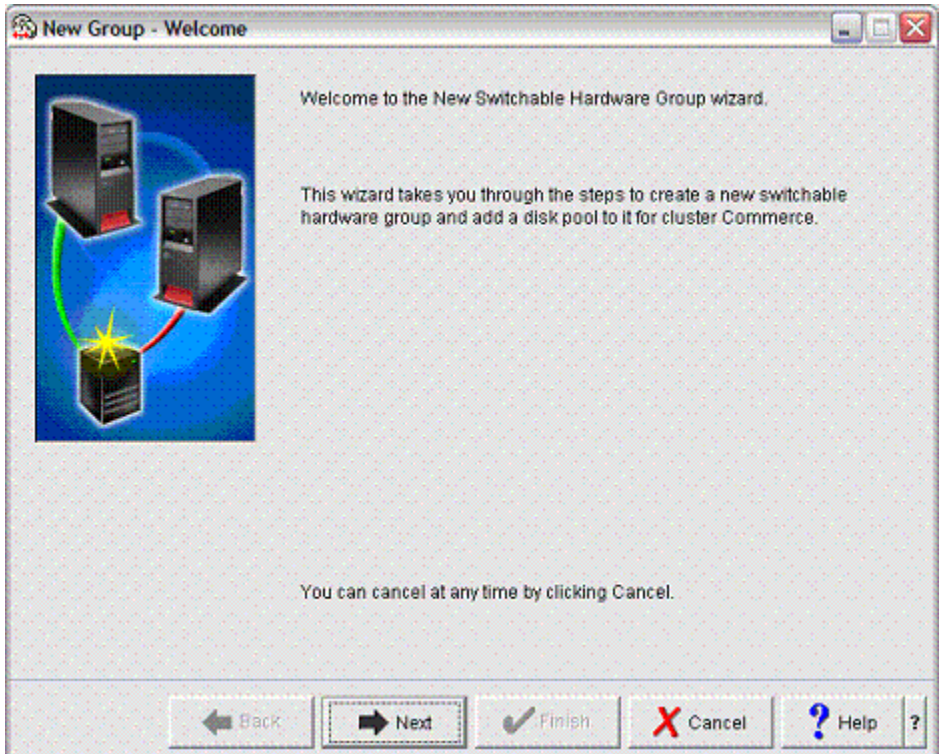
Step 6. Create a Switchable Hardware Group with a New Disk Pool Called 'HADB'

Within Management Central, we clicked on the + sign to expand the content of the Commerce cluster.

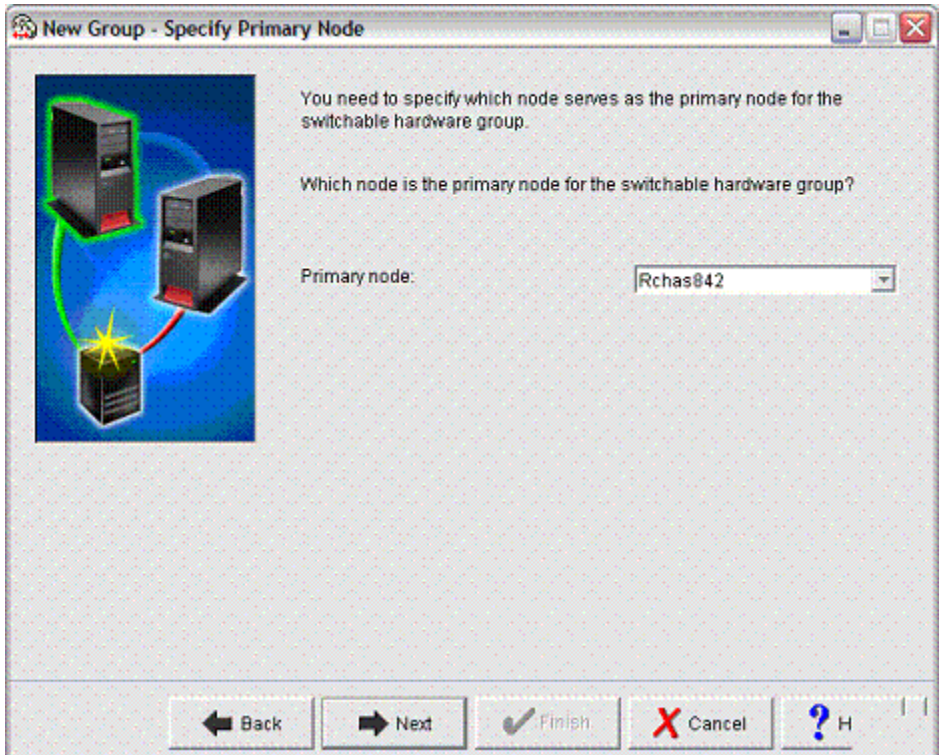


Ensure that both of the cluster nodes are started. To start a node, select it, perform a right-click operation, and select **Cluster => Start**.

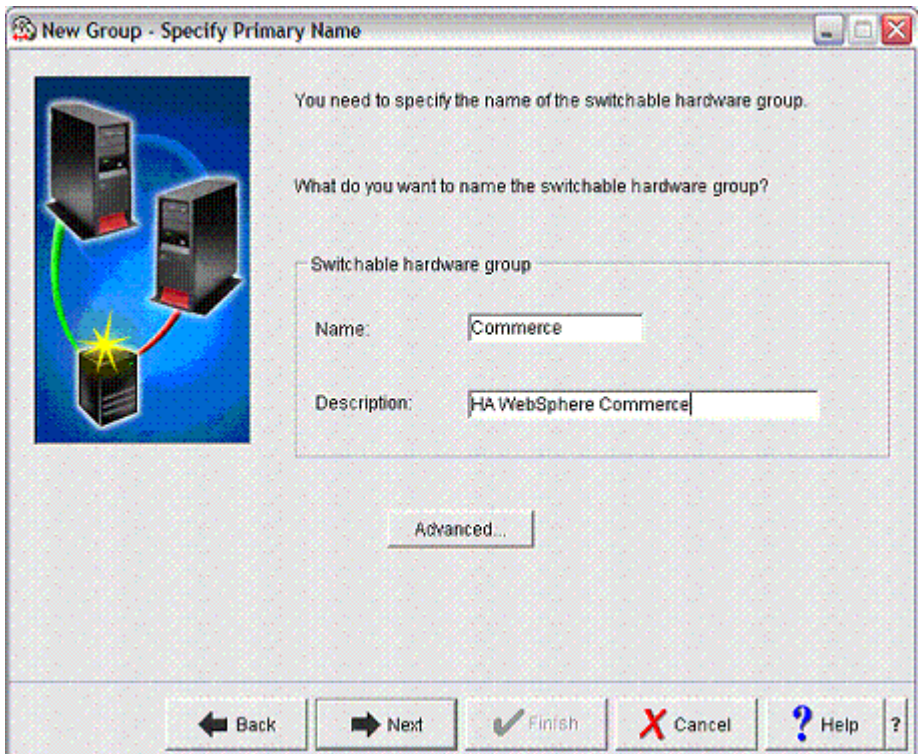
We then selected **Switchable Hardware** and performed a right-mouse click to choose the **New Group** menu option.



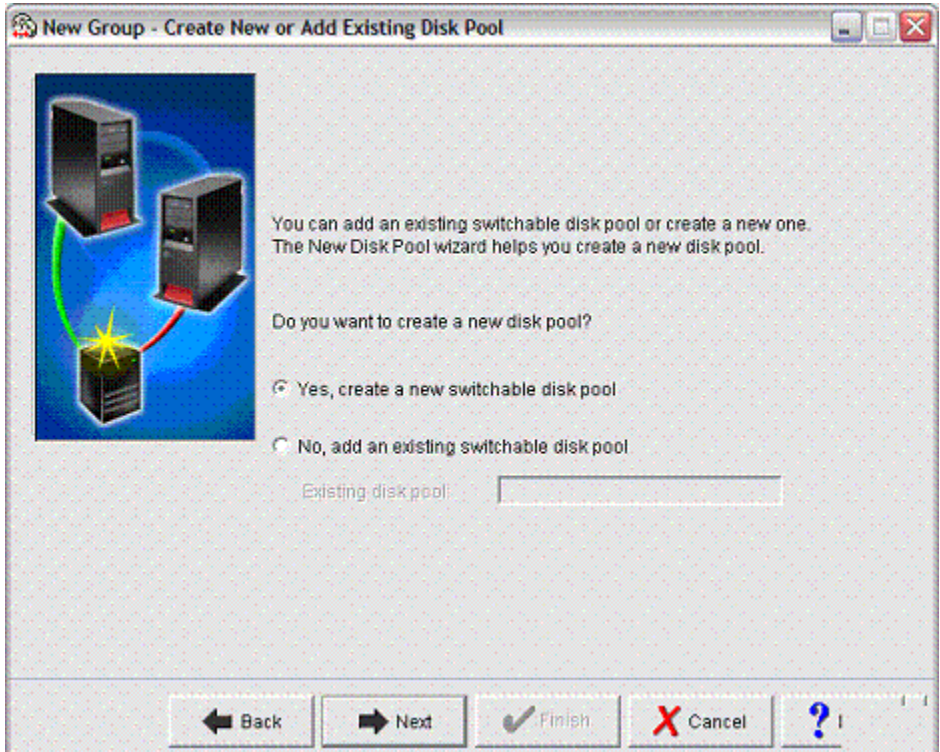
We clicked **Next** on the welcome window.



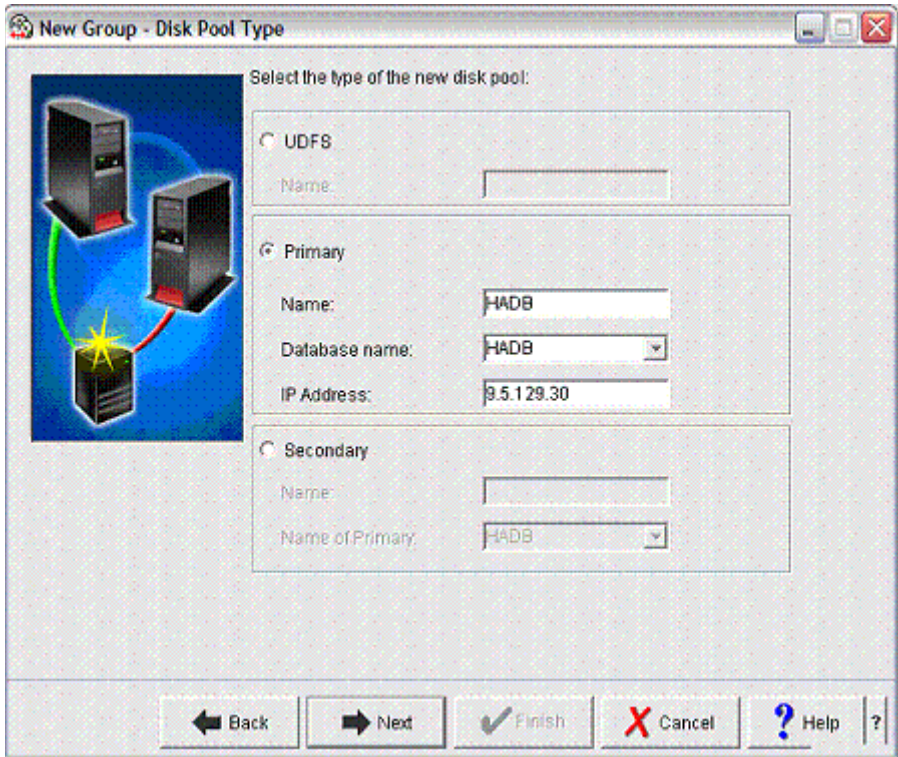
The primary node was selected, and then the **Next** button was clicked.



We entered a name of `Commerce` for the switchable hardware group and a brief description before clicking `Next`.

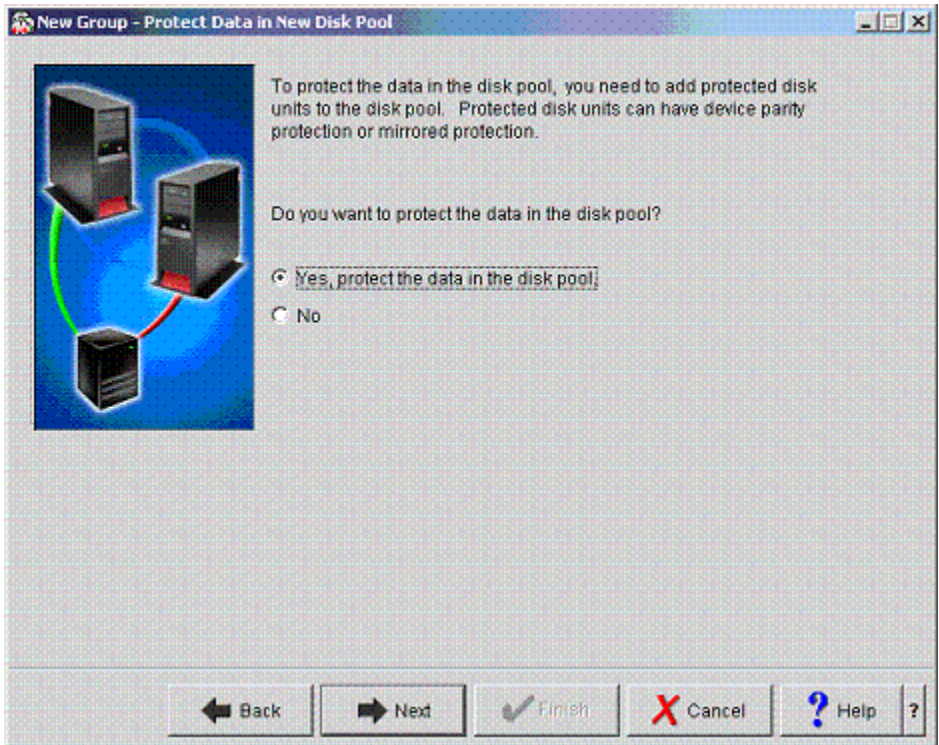


We chose the radio option to create a new switchable disk pool and clicked **Next**.

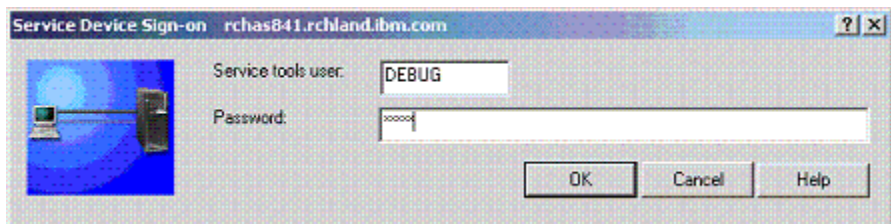


The name we chose for the primary independent ASP was HADB. This will also be the relational database name that will be automatically generated by the system upon creation of the primary independent ASP. Even though HADB is displayed in the screen example above, one should select the 'system-generated' option for the Database name. The IP address noted in the screen above is the IP takeover address that will be used for access to the database by user applications. Using this IP address entry is critical to ensuring that access to the database can be accomplished after independent ASP switchover or failover processing is complete.

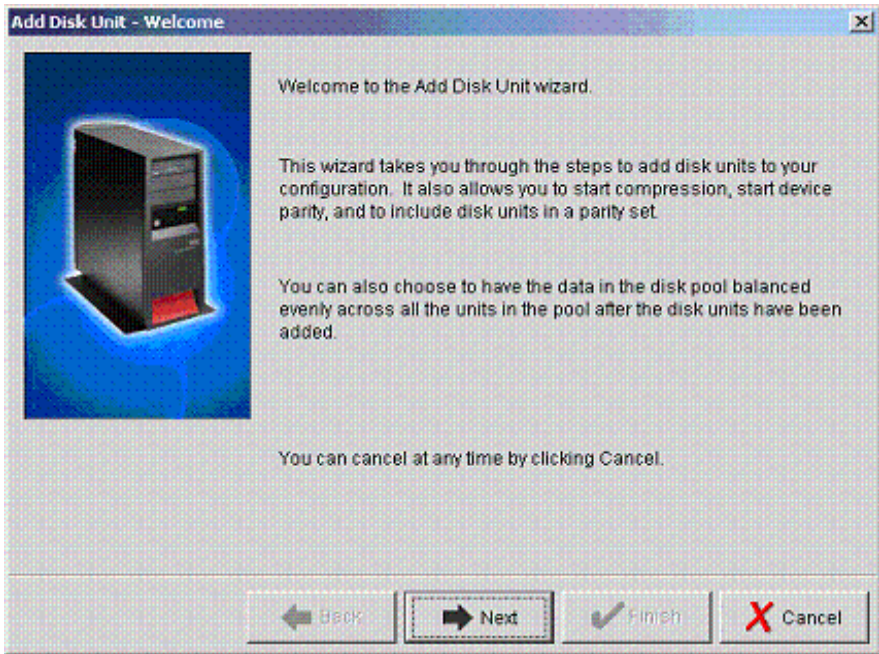
We clicked **Next** to proceed to the next window.



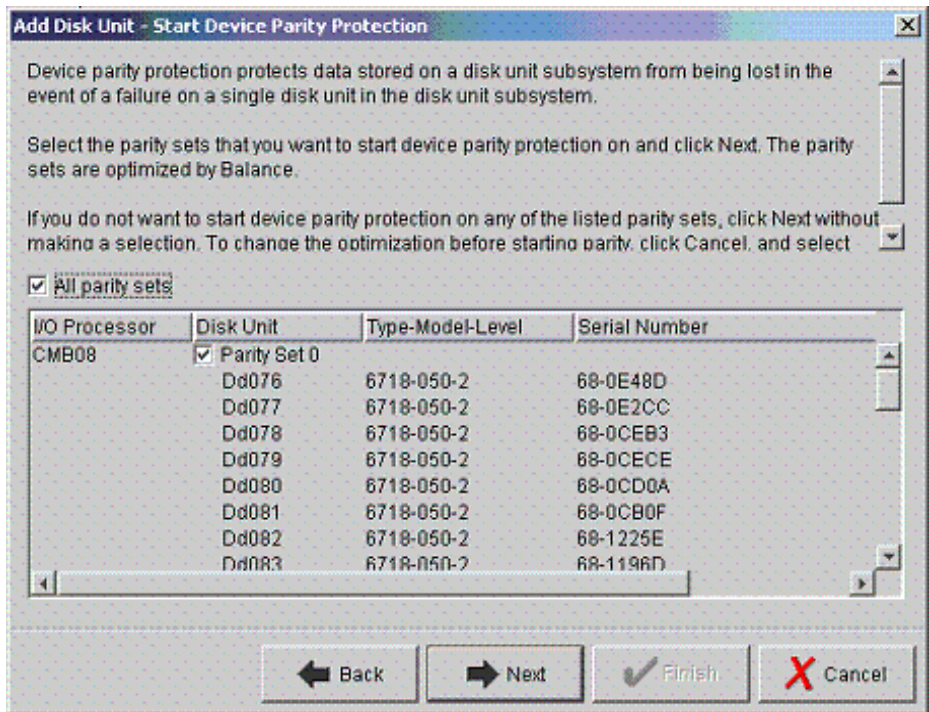
It is imperative that you choose to use either mirrored or device parity protection for the disk units within an independent ASP. Therefore, we chose the Yes option and then clicked **Next**. We were prompted by the following Service Device Sign-on dialogue box:



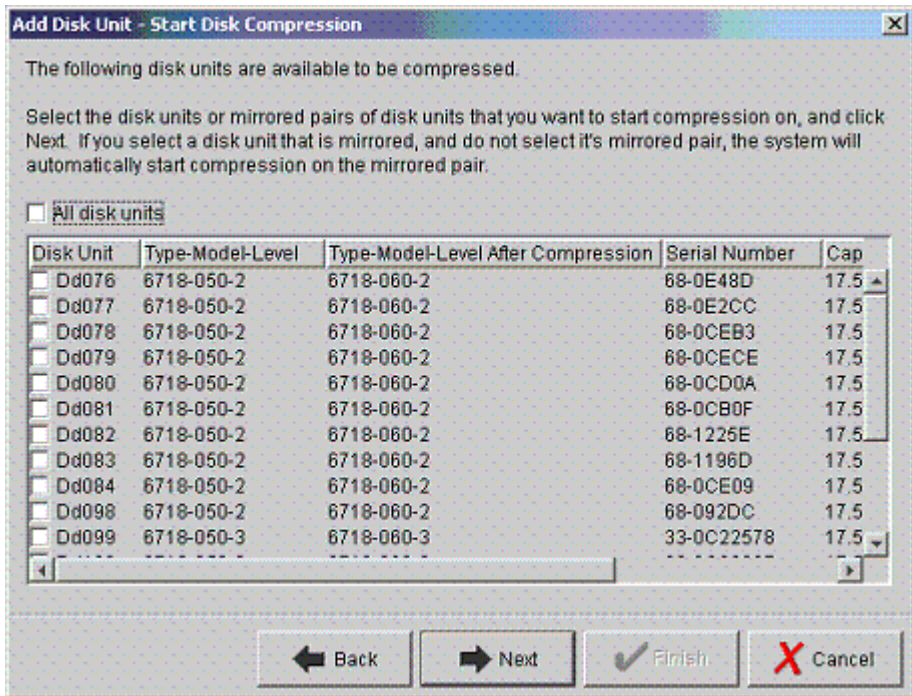
We entered a valid user ID and password and clicked **OK**.



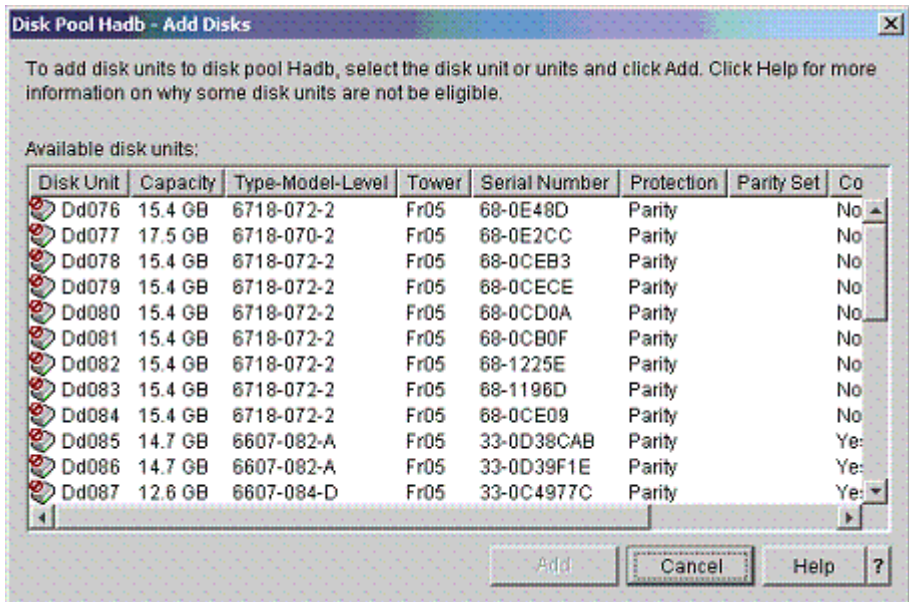
We clicked **Next** to proceed to the next screen.



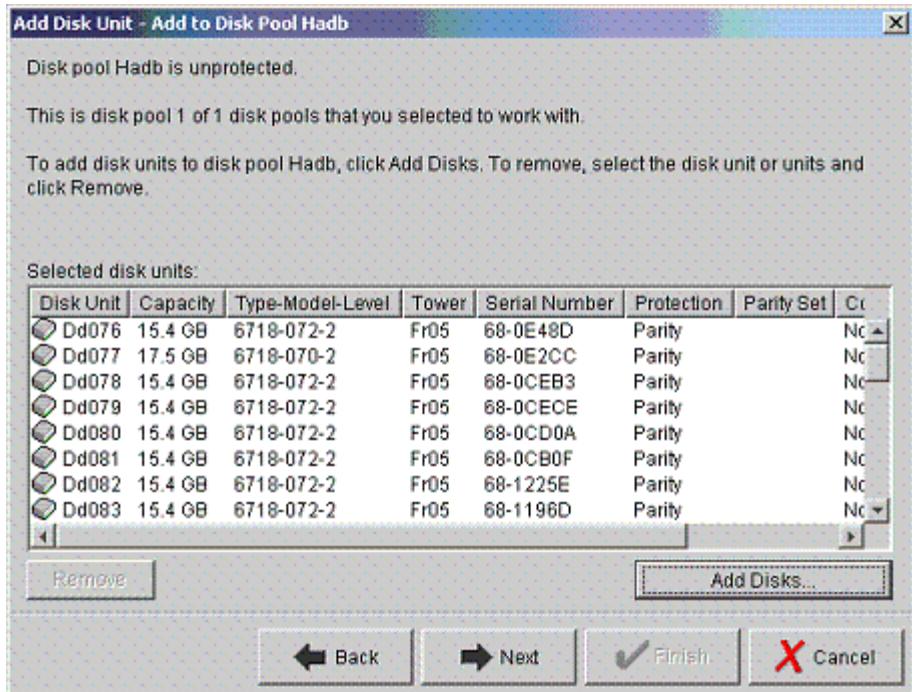
We chose to enable device parity protection and then clicked **Next**. You may encounter another 'Add Disk Unit' screen that has no disk units displayed. Simply click the 'Add Disks' button in that case.



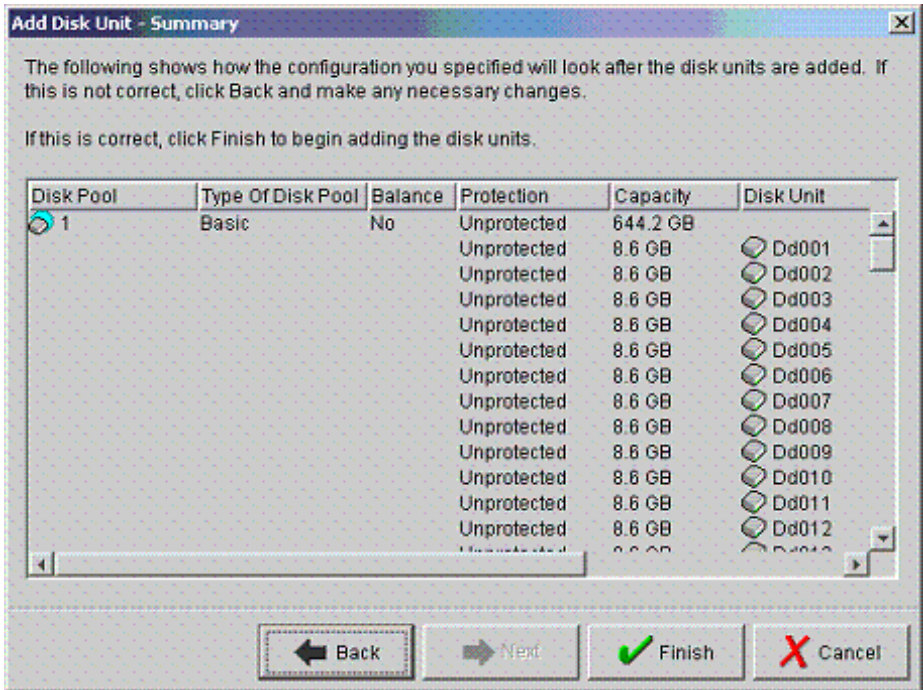
Since enabling compression is generally not recommended for disk units that will store production database collections, we simply clicked Next to go to the next window.



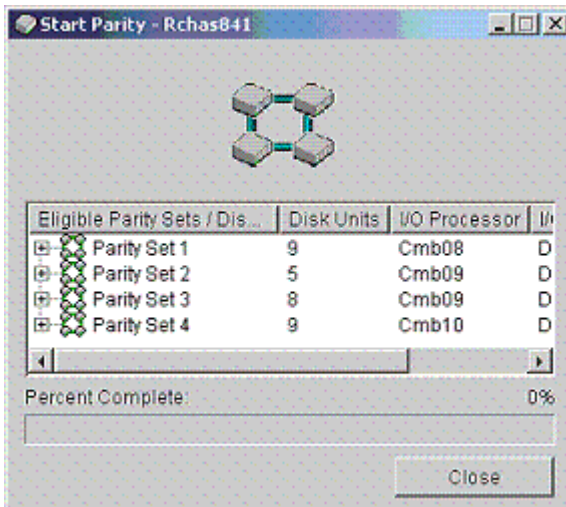
We selected all the disk units to be added to disk pool HADB and then clicked **Add**.



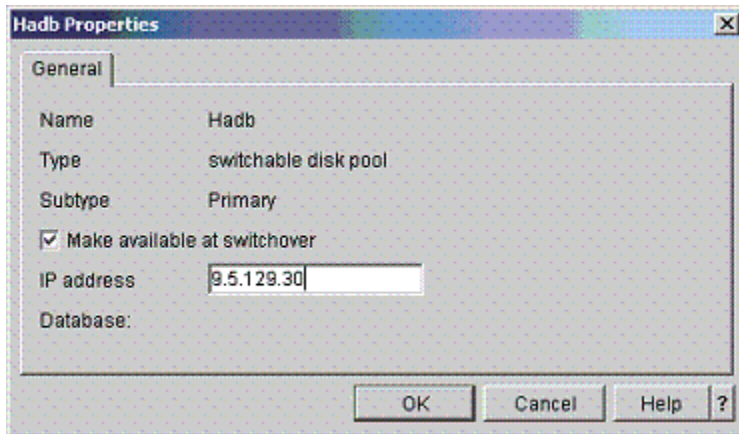
We clicked **Next** to add the units.



This is simply a summary screen. We clicked **Finish** to complete the process.



A status screen will appear to give you an indication of progress. Once the operation completes, it is a good idea to double check that the switchable disk pool will be made available at switchover time and that an IP takeover address has been specified. Within Management Central, we right-clicked the primary disk pool HADB and selected the Properties context menu option to display the following window:



Lastly, make the new disk pool available through the following sequence of steps:

- 1 In iSeries Navigator, expand My Connections (or your active environment).
- 2 Expand any iSeries server.
- 3 Expand Configuration and Service.
- 4 Expand Hardware.
- 5 Expand Disk Units.
- 6 Sign on to service tools if the Service Tools Signon dialog displays.
- 7 Expand Disk Pools.
- 8 Right-click the unavailable disk pool and select Make Available.
- 9 From the dialog displayed, click Make Available to make the disk pool available.

You can also use the Vary Configuration (VRYCFG) CL command to make the disk pool available.

The independent ASP configuration is now complete. From this point forward on a regular basis it is a good idea to verify that both the cluster nodes and switchable hardware are started and that the IASP is available, especially if any of the cluster nodes occasionally get rebooted due to system maintenance activities. This will help provide assurance that OS/400 clustering will be in the proper activation state to help meet your high-availability objectives in the event of a planned or unplanned outage.

Step 7. Restore INVEST Database Collection into the iASP

First, we saved a production copy of our INVEST database collection to a save file object and then transferred the save file to the primary node in our new cluster via ftp commands. Next, the following restore command was issued on the primary node in the cluster to place the database collection into the 'HADB' switchable disk pool:

```
RSTLIB SAVLIB (INVEST) DEV (*SAVF) SAVF (DARIN/INVEST) RSTASPDEV (HADB)
```

A user profile called 'dbjava' with *pgmr authority was granted authority to the objects in the INVEST library. That profile will be used for JDBC access to the database by the WebSphere application.

Step 8. Enable WebSphere Persistent Sessions Database to be Stored in the iASP

We signed on to the primary node in the cluster and used the following interactive SQL commands to create a collection called 'HAWAS' that will be used for the persistent storage of servlet session data.

```
STRSQL
Connect to HADB
Create collection HAWAS
```

Lastly, the 'dbjava' user profile was granted authority to the HAWAS library.

Step 9. Enable WebSphere Distributed Application Access to the iASP

In our scenario, the user applications (i.e. WebSphere servlets) that require access to the database were hosted on WebSphere application servers that resides on remote iSeries servers. Therefore, additional relational database entries needed to be added on each of the iSeries servers that are running the application server. Issue the ADDRDBDIRE CL command on those systems to add the following relational database entry:

```
ADDRDBDIRE RDB(HADB) RMTLOCNAME('9.5.129.30' *IP) TEXT('RDB entry to access iASP database')
```


Notice that HADB is specified as the relational database name and that the IP takeover address is used for the remote location value. One can then verify that basic access to the remote HADB database works correctly by using interactive SQL to issue simple queries against the database after using the 'connect to HADB' statement. Keep in mind that a user ID and password may need to be specified on the 'connect to' statement so that one has proper authority to the database objects used in the test queries.







This relational database entry needs to be added to all remote WebSphere servers that access database objects that are stored in the iASP.

Within the WebSphere Administrative Console, configuration changes needed to be done with regard to datasources definitions for access to our INVEST and HAWAS database collections.

[JDBC Providers](#) > [Toolbox.JDBC](#) > [Data Sources](#) > [invest5](#) > [Custom Properties](#) >

databaseName

Custom properties that may be required for Resource Providers and Resource Factories. For example, most database vendors require additional custom properties for data sources that will access the database. 

Configuration		
General Properties		
Scope	<input type="radio"/> cells:ND839Network	 The scope of the configured resource. This value indicates the configuration location for the configuration file.
Required	false	
Name	databaseName	 Name associated with this property (for example, PartNumber and ConnectionURL).
Value	<input type="text" value="HADB"/>	 Value associated with this property in this property set.
Description	Specifies the database to use for the connection, including one stored in an independent auxiliary storage pool. This property applies only when connecting to a V5R2 or later version of OS/400, in V5R1 and earlier this property is ignored. When you specify a database name, the name must exist in the relational database directory on the server.	 Text to describe any bounds or well-defined values for this property.
Type	java.lang.String	 Fully qualified Java type of this property (java.lang.Integer, java.lang.Byte).

The databaseName property value for our Invest5 datasource was changed to HADB and the **Apply** button was clicked. Perform this same change for any other datasources that require access to the IASP.

Step 10. Verify Switchable Disk Pool Configuration

This document covers numerous technologies. Mission-critical applications should be tested in a 'normal' environment prior to deployment to a more complex HA environment to ensure that the applications are sound. Then, as each application component is made more highly-available through the technologies described in this document, care should be taken to do some basic run-time and failover testing along the way. This will ease the debug efforts and reduce the overall time required to complete the deployment of an end-to-end solution involving numerous high-availability components.

Therefore, a suite of independent ASP failover and switchover tests should be conducted prior to going into production mode with the final solution to help ensure that your particular high-availability requirements can be met when there is a failure of one of the cluster nodes or when a planned maintenance event occurs that may involve switchover processing.

Factors that can affect IASP failover or switchover time

- 1** The job priority of the process that performs the switchover processing can be a major factor. This is especially true on systems with many active jobs or even just a few resource-intensive jobs since the switchover processing will have to compete for available system resources. The priority associated with the user profile who initiates the planned switchover is used for switchover processing. If you decide that the switchover processing should be more important than the currently active jobs, then you need to take steps to ensure that the switchover processing will run at an important enough priority level, such as priority 20. You may wish to create a special subsystem and profile that is solely used for IASP system management functions to achieve that goal without having to make changes to existing job descriptions or profiles that are used for other tasks not related to IASP functions.
- 2** The Group ID and User ID numbers for user profiles should be synchronized across the nodes in the cluster to avoid potentially lengthy reconciliation processing during failover or switchover processing. One should use Management Central to perform the synchronization of the Group ID and User ID numbers to eliminate this factor.
- 3** Systems with more processors will generally witness improved failover and switchover times since the database catalog merge processing step is a multi-threaded process. Also, bear in mind that using the Capacity on Demand models can have significant performance effects for any system environment due to potential changes in the number of processors. Moreover, the number of processors affects the query engine's choice of access plans.
- 4** The total number of database fields that are resident in *SYSBAS and the IASP affects the failover and switchover time. A system in a cluster that has millions of database fields will experience noticeably longer failover/switchover times than systems with only a few thousand database fields. One can issue the following interactive SQL statement to get an idea of how many database fields exist on a system: `select count(*) from qsys/qadbifld.`
- 5** If the IASP vary-on processing is flagged as abnormal by the system, then the database recovery steps during the vary-on processing will take longer. This most often occurs with failover events that involve unplanned outages. Also, in the case of an unplanned outage, please bear in mind that database access path rebuilds are performed asynchronously once the failover processing is complete. It is wise to either use explicit journaling of access paths or to adjust the system-managed access path protection via the EDTRCYCAP CL command to minimize or eliminate the need for access path rebuilds associated with an abnormal termination of a system or partition.
- 6** Per normal system performance considerations, the number of disk arms in the system ASP and the IASP can effect the efficiency of I/O operations. For instance, it would be unwise to have an IASP with just one or two disk units.
- 7** As you might expect, simply adding more switchable towers will increase the failover and switchover time for an IASP.

- 8 If there is an application CRG exit program associated with the switchable IASP, then the extra time to perform that processing is a factor in the total switchover or failover time.

Useful IASP tasks

Here are some instructions on doing various tasks on IASPs.

Switching access to the backup server

In a multi-system clustered environment that uses switchable independent disk pools, an independent disk pool can only be accessed by one node at a time. Current access to a switchable independent disk pool is managed through the switchover function within the cluster.

To switch access from the current node in the cluster to the backup node:

- 1 Make the disk pool unavailable. (This step is optional. The switchover processing in the next step attempts to make the disk pool unavailable if it is currently available.)
- 2 Switch the independent disk pool to the backup cluster node by performing a switchover in the cluster. Performing a manual switchover causes the current primary node to switch over to the backup node, as defined in the cluster resource group's recovery domain. When this happens, the current roles of the nodes in the recovery domain of a cluster resource group change such that:
 - The current primary node is assigned the role of last active backup.
 - The current first backup is assigned the role of primary.
 - Subsequent backups are moved up one in the order of backups.

A switchover is only allowed on CRGs that have a status of ACTIVE.

- a In iSeries Navigator, expand **Management Central**
- b Expand **Clusters**
- c Expand the cluster containing the desired resource
- d Click **Switchable Hardware**
- e Right-click the desired resource, and select **Switch** (to switch a resource, the resource must have a status of **Started**)

You can also use cluster APIs such as the Change Cluster Resource Group Primary (CHGCRGPRI) command to perform the switchover.

Deleting an independent disk pool

If you do not need to access the data in a disk pool ever again, you can choose to clear the disk pool or delete the disk pool. Both actions destroy all data on the disk units in the disk pool. If you choose to clear the disk pool, the disk units are still available for new data storage. If you delete the disk pool all disk units are removed and you can no longer access the disk pool. If you want to delete or clear an independent disk pool that is unavailable, you can do so when your system is fully restarted.

If you delete an independent disk pool that is participating in a clustered environment, it is strongly recommended that you first remove the disk pool from the cluster resource group (CRG) using the Remove Cluster Resource Group Device Entry (RMVCRGDEVE) command. Under certain circumstances, you must end the CRG first; for example, use the End Cluster Resource Group (ENDCRG) command first if you plan to remove a subset

of an independent disk pool group or remove the last independent disk pool in the CRG. If you must delete the independent disk pool first, make sure you remove it from the CRG afterward.

To delete or clear a disk pool, follow these steps:

- 1 In iSeries Navigator, expand **My Connections**.
- 2 Expand the desired iSeries server.
- 3 Expand **Configuration and Service**.
- 4 Expand **Hardware**.
- 5 Expand **Disk Units**.
- 6 If the Service Tools Signon dialog displays, sign on to service tools.
- 7 Expand **Disk Pools** and select the disk pools you want to clear.
- 8 Make sure the disk pool is unavailable (see “Making a disk pool unavailable” on page 341).
- 9 Right-click a selected disk pool and select **Clear** or **Delete**.
- 10 Follow the instructions on the dialog that is displayed.

Making a disk pool available

To access the disk units in an independent disk pool and the objects in the corresponding database you must make the disk pool available (i.e. varied on). In a multi-system clustered environment, you can make the disk pool available to the current node or to another node in the cluster. The independent disk pool can only be varied on to one node at a time. When you want to access the independent disk pool from a different node, you must switch the independent disk pool to the backup cluster node.

NOTE If you make a primary or secondary disk pool available, all of the disk pools in the disk pool group will also be made available at the same time.

To make an independent disk pool available:

- 1 In iSeries Navigator, expand **My Connections** (or your active environment).
- 2 Expand any iSeries server.
- 3 Expand **Configuration and Service**.
- 4 Expand **Hardware**.
- 5 Expand **Disk Units**.
- 6 Sign on to service tools if the Service Tools Signon dialog displays.
- 7 Expand **Disk Pools**.
- 8 Right-click the unavailable disk pool and select **Make Available**. You can select multiple disk pools to make available at the same time.
- 9 From the dialog displayed, click **Make Available** to make the disk pool available.

You can also use the Vary Configuration (VRYCFG) command in the character-based interface to make the disk pool available.

Making a disk pool unavailable

You can select an independent disk pool to make unavailable (vary off). You will not be able to access any of the disk units or objects in the independent disk pool or its corresponding database until it is made available (varied on) again. The pool can be made available again on the same system or another system in the recovery domain of the cluster resource group.

IMPORTANT Before an independent disk pool can be made unavailable, no jobs can hold reservations on the disk pool.

To make an independent disk pool unavailable:

- 1 In iSeries Navigator, expand **My Connections** (or your active environment).
- 2 Expand any iSeries server.
- 3 Expand **Configuration and Service**.
- 4 Expand **Hardware**.
- 5 Expand **Disk Units**.
- 6 Sign on to service tools if the Service Tools Signon dialog displays.
- 7 Expand **Disk Pools**.
- 8 Right-click the disk pool you want to make unavailable and select **Make Unavailable**.
- 9 From the dialog that displays, click **Make Unavailable** to make the disk pool unavailable.

You can also use the Vary Configuration (VRYCFG) command in the character-based interface to make the disk pool unavailable.

Balancing a disk pool

You can balance the data on a disk pool in your system. Balancing a disk pool improves system performance by balancing disk capacity across all the disk units in a disk pool.

There are two ways to balance a disk pool using iSeries Navigator:

- Use the Add Disk Unit wizard when you add disk units to a pool.
- Use the Add Disk Unit wizard when you create a new disk pool.

After you start the wizard, follow the instructions provided by the wizard. If you add disk units to an existing disk pool that contains data, one of panels in the wizard asks whether you want to balance the disk units you are adding.

Recovering an independent disk pool

If you are experiencing problems accessing an independent disk pool or making it available, there may be a problem with the disk pool. Possible problems include:

- The configuration source is corrupted. When corruption occurs, the independent disk pool will appear to have no disk units in it. The disk pool may also appear to have no disk units in it if it is switched to another node in a clustered environment. Before you attempt a recovery, make sure that no other system owns the disk pool. If you know the serial numbers of the disk units in the independent disk pool that may need recovery, make sure you are on the system that owns those disk units and that they show as non-configured. If the configuration source is corrupted, you can select to recover the configuration information on the configuration source. Recovering the

configuration attempts to determine the original configuration and recover it. During this process, the independent disk pool may need to be cleared, destroying all data on the disk units in the pool. If the disk pool needs to be cleared, a message displays warning you of this and allowing you to cancel the recovery.

- The mirrored disk unit of the configuration source is damaged. When this happens, the mirrored configuration source becomes unknown. The disk pool will be unavailable, and you must recover the configuration information of an unknown configuration source before making it available. You should only attempt to recover the state of the unknown configuration source when you know its mirrored disk unit was active before the failures that caused the state to become unknown.

To attempt to recover an independent disk pool, follow these steps:

- 1 In iSeries Navigator, expand **My Connections** (or your active environment).
- 2 Expand any iSeries server.
- 3 Expand **Configuration and Service**.
- 4 Expand **Hardware**.
- 5 Expand **Disk Units**.
- 6 If the Service Tools Signon dialog displays, sign on to service tools.
- 7 Select **Disk Pools**.
- 8 Right-click the problematic disk pool. If iSeries Navigator detects one of the problems listed above, then **Recover Configuration** or **Recover Unknown Configuration Source** appears in the list. If you see either of these options, select it to continue.
- 9 Follow the instructions on the dialog displayed.

Changing the server takeover IP address

The server takeover IP address is associated with a primary disk pool in a clustered, switchable environment. Specifically, it is the IP address for a server associated with the relational database name in the device description for a switchable independent disk pool. The specified address must exist on all nodes in the recovery domain if the cluster resource group is active.

To change the server takeover IP address for a primary disk pool, follow these steps:

- 1 In iSeries Navigator, expand **Management Central**.
- 2 Expand **Clusters**.
- 3 Expand the cluster that contains the switchable hardware group.
- 4 Expand **Switchable Hardware**.
- 5 Click the switchable hardware group, then right-click the desired primary disk pool and select **Properties**. Note that server takeover IP address can only be associated with a primary switchable independent disk pool.
- 6 Change the server takeover IP address in the IP address field.

You can also use the Change Cluster Resource Group Device Entry (CHGCRGDEVE) command in the character-based interface to change the server takeover IP address.

Setting the threshold of a disk pool

You can set the threshold of a disk pool by following these steps:

- 1** In iSeries Navigator, expand **My Connections** (or your active environment).
- 2** Expand any iSeries server.
- 3** Expand **Configuration and Service**.
- 4** Expand **Hardware**.
- 5** Expand **Disk Units**.
- 6** Expand **Disk Pools**.
- 7** Right-click the disk pool for which you want to change the threshold and select **Properties**.
- 8** Select the **Threshold** tab. On this page, specify whether you want to increase or decrease the threshold for the disk pool.

Removing a disk unit from an IASP

You can remove a disk unit from an IASP when it is unavailable by following these steps:

- 1** In iSeries Navigator, expand **My Connections** (or your active environment).
- 2** Expand any iSeries server.
- 3** Expand **Configuration and Service**.
- 4** Expand **Hardware**.
- 5** Expand **Disk Units**.
- 6** Expand **Disk Pools**.
- 7** Select the disk unit to be removed, right-click, and select Remove.
- 8** Confirm the action.
- 9** A window opens that indicates successful completion.

Adding a disk unit to an existing IASP

You can add a non-configured disk unit to an IASP by following these steps:

- 1** In iSeries Navigator, expand **My Connections** (or your active environment).
- 2** Expand any iSeries server.
- 3** Expand **Configuration and Service**.
- 4** Expand **Hardware**.
- 5** Expand **Disk Units**.
- 6** Expand **Disk Pools**.
- 7** Right-click the disk pool to which to add a unit and select Add Unit.
- 8** Confirm the action. If you chose to balance the data during the disk unit add, a warning window opens that indicates that the balancing cannot take place until the IASP is made available.
- 9** A window opens that indicates successful completion

Save independent ASPs

You can save independent ASPs separately or you can save them as part of a full system save (GO SAVE: Option 21), or when you save all user data (GO SAVE: Option 23). In either case, you must make the independent ASPs available before you perform the save. The following scenarios shows you how you can save the IASPs independently. See links to resources at the end of the chapter for information on performing a full system save or saving all user data.

Save the current ASP group

Perform the following commands to save the current independent ASP group (the primary ASP and any associated secondary ASPs).

- 1 SETASPGRP ASPGRP(primary-ASP-name)
- 2 SAVSECDTA ASPDEV(*CURASPGRP)
- 3 SAVLIB LIB(*ALLUSR) ASPDEV(*CURASPGRP)
- 4 Unmount any QDEFAULT user-defined file systems in the current independent ASP group
- 5 SAV OBJ('/dev/*') UPDHST(*YES) ASPDEV(*CURASPGRP)
- 6 Mount any QDEFAULT user-defined file systems that were unmounted in an earlier step

Save UDFS ASP

Perform the following commands to save an available UDFS ASP.

- 1 SAVSECDTA ASPDEV(ASP-name)
- 2 Unmount any QDEFAULT user-defined file systems in the UDFS ASP that you are saving
- 3 SAV OBJ('/dev/*') UPDHST(*YES) ASPDEV(ASP-name)
- 4 Mount any QDEFAULT user-defined file systems that were unmounted in an earlier step

Resources for Learning

Use the following links to find relevant supplemental information about various topics discussed in this chapter. The information resides on IBM and non-IBM Internet sites, whose sponsors control the technical accuracy of the information.

These links are provided for convenience. Often, the information is useful all or in part for understanding a topic discussed in this chapter. When possible, links are provided to technical papers and Redbooks that supplement the broad coverage of the release documentation with in-depth examinations of particular product areas.

Independent ASPs

- Independent disk pools
<http://publib.boulder.ibm.com/series/v5r2/ic2924/index.htm?info/rzaly/rzalyoverview.htm>
This iSeries Information Center topic contains information on iASPs.
- *iSeries Independent ASPs: A Guide to Moving Applications to IASPs* (May 2003)

<http://www.redbooks.ibm.com/redbooks/pdfs/sg246802.pdf>

This IBM Redbook explains how to install and configure the new independent auxiliary storage pool (IASP) functionality of OS/400 V5R2. It is designed to help IBM technical professionals, Business Partners, and Customers understand and implement IASP in the IBM iSeries server and under OS/400 V5R2.

In addition, this redbook provides the background information that is necessary to plan, implement, and customize this functionality to your particular environment. It provides advice on running native OS/400 applications with either application data or most application objects residing in an IASP. Considering you can also use IASPs in a cluster environment, this redbook shows you the basic steps to make your IASP switchable between two iSeries servers in a high-speed link (HSL) loop.

- *Clustering and IASPs for Higher Availability on the IBM eServer iSeries Server* (April 2002)

<http://www.redbooks.ibm.com/redbooks/pdfs/sg245194.pdf>

CHAPTER 20

TESTING

This chapter will describe the process of testing your highly available e-business environment in addition to listing some common tests that should be considered to verify your highly available environment (the list of tests is by no means complete - we do not go into firewall testing or network testing, etc.).

Preparation

Applications Used to Drive Workload

You want an application (or applications) that is representative of the applications that you will be deploying in your production environment to drive the workload. For example, if you plan to use EJBs, access databases, and use session objects, then you want an application that has the following characteristics:

- Uses EJBs
- Accesses DB2
- Creates session objects

Tool Used to Drive Workload

A tool is required to drive the workload. LoadRunner, from Mercury Interactive Corporation (see <http://www.merc-int.com>), is the tool we usually use to drive applications in order to create a workload sufficient to load the system and test the high-availability aspects of the architecture we developed. There are many other tools that perform a similar function. In most tools that drive workload, scripts need to be created that would represent a client. These scripts are then fed into the tool and would represent single or multiple clients.

The Tests

There are two areas one needs to consider:

- Infrastructure availability tests (planned and unplanned outages)
- Application availability tests (planned and unplanned outages)

Infrastructure Availability Tests

In the context of the infrastructure, planned outages are activities such as upgrading system software or applying PTFs. Unplanned outages are all the things that can go wrong to cause a component of the system to stop working.

Planned Outages

In the area of planned outages, one must consider the following:

- Installing new maintenance level of the operating system
- Installing new maintenance level of WebSphere Application Server Network Deployment for iSeries
- Installing operating system fixes.

Unplanned Outages

You should run a series of tests to verify that the architecture developed for high availability will continue to provide unbroken application service during a wide range of unplanned outages.

Every time a test is about to be performed, you first must start a workload using whatever tool you have chosen to drive workload. With the workload running at full volume, the test is started by causing an outage in a particular component in your e-business environment. Table 20.1 on page 348 lists some of the tests you should run.

Table 20.1 List of unplanned outage tests

Test	Description	Expected results
Load balancer	Load balancer failure	A failure of the primary causes the backup to take over the workload. Application users do not notice this failure.
Web server	HTTP server failure	If one fails, the other takes over the entire workload. Performance may be degraded.
Server Failure	Failures of multiple LPARs	Workload is transferred to the remaining LPARs on the second physical LPAR complex. If not using session persistence, sessions that were being handled by WebSphere in the failed LPAR will have to be restarted by the user.
WebSphere	Failure in application server	Active transactions on this WebSphere will terminate. No new transactions are routed to this system until WebSphere is restarted. If not using session persistence, sessions that were being handled by WebSphere in the failed application server will have to be restarted by the user.
WebSphere with session persistence	Failure in application server	The current transactions that were running in this application server will fail, but later transactions in this session are handled by other servers in the cluster. No new transactions are routed to the application server until WebSphere is restarted.
DB2	DB2 failure	Database operations fail until the take-over IP address is activated on backup database server.

In all cases, the application should remain continuously available from the user's perspective, even though a component may take an unplanned outage. For most

unplanned outages, the system recovers automatically—but for some, the system does not recover without intervention. For those cases, you should have a written plan documenting the steps one would follow to recover the failed component.

Application Availability Tests

In this section we introduce our approach to upgrading a J2EE application within our high availability WebSphere environment. There are two reasons to upgrade an application:

- For a planned outage (new version/release of an application)
- For an unplanned outage (an application error occurred and needs to be fixed)

We use the same methodology to upgrade an application in both cases; the difference is that we can choose the time and date for a planned outage.

Upgrading an Application

In this section, we describe the necessary steps to install a new version/release of a J2EE application.

To identify an upgrade path for an application, we need to look at the packaging first. In our case the packaging consists of two components:

- One EAR file containing the JSPs, servlets and EJBs running on WebSphere Application Server
- DB2 tables

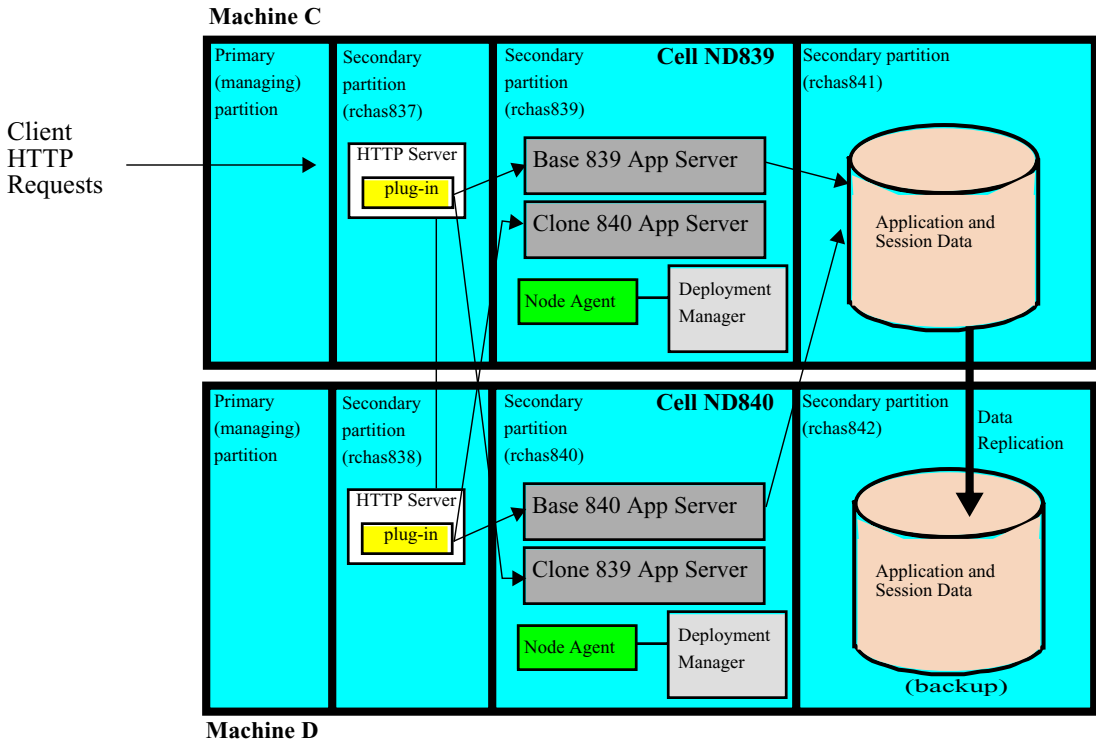
Since we are not changing the DB2 tables, we only have to deal with the upgrade of the EAR file located in WebSphere.

One EAR file containing all these parts is one way to package an application. There are different ways to package an application, depending on the infrastructure you are using. One possibility would be to separate the presentation logic (servlet and JSPs) from the business logic (EJBs). Then you can deploy the presentation logic part of the application into a different application server. Using this method, you still can use the upgrade path we use as a model and derive your own upgrade path from it.

The upgrade steps will be different depending on the topology chosen. For example, to upgrade an application in the peer cell topology discussed in “Recommended Topologies (Data Center)” on page 235 (see Figure 20.1 on page 350 below), we identified the following steps:

- 1 End Web server in secondary partition rchas837.
- 2 Deploy the new application in cell ND839.
- 3 Test new application in cell ND839.
- 4 Start Web server in secondary partition rchas837.
- 5 End Web server in secondary partition rchas838.
- 6 Deploy the new application in cell ND840.
- 7 Test new application in cell ND840.
- 8 Start Web server in secondary partition rchas838.

Figure 20.1 WebSphere Application Server dual cell with HA HTTP Server



On the other hand, if you had a WebSphere cell that was active and a backup cell that was inactive, you would probably perform the following steps:

- 1 Upgrade application in backup cell.
- 2 Test application in backup cell.
- 3 Switch so backup cell is activated (need to end and then restart HTTP servers after updating to point to backup plug-in configuration file).
- 4 Upgrade application in production cell (which is now inactive).
- 5 Test application in production cell.
- 6 Perform step 3 to switch roles (optional).

Planned Outages

We consider deploying a new version/release of an application to be a planned outage, since we can choose when to perform this task. For our deployment, we followed the steps described in “Upgrading an Application” on page 349.

Unplanned Outages

An unplanned outage occurs due to an application error which needs to be resolved as soon as possible.

Once an application error occurs, you need to perform these tasks:

- 1 Problem identification
- 2 Workaround (if possible)
- 3 Rolling upgrade

To simulate an unplanned outage, we usually deploy a different application from the ones described in “Applications Used to Drive Workload” on page 347, one that causes a memory leak in the JVM. After calling this application n-times, the server could not handle any new requests because of a `java.lang.OutOfMemoryError`.

Problem identification

Continuing with the memory leak example in the application, we can diagnose a memory leak within a Java application by using:

- Jinsight 2.1 technology allows you to diagnose memory leaks. It is available on: www.alphaworks.ibm.com/tech/jinsight.
- WebSphere Studio Application Developer provides a profiling tool that collects data related to Java runtime behavior, and presents this data in both graphical and non-graphical views. It allows you to identify an application's performance behavior during the development cycle, and also allows you to define remote hosts and collect their data.
- Execution of the Dump Java Virtual Machine (DMPJVM) or Analyze Java Virtual Machine (ANZJVM) OS/400 CL commands at periodic intervals against an OS/400 JVM process can be used to diagnose leaks. The output of those commands is dumped to a spool file.

Workaround

So we know why the application is failing, but until the application developer analyzes the problem and fixes the failing application, we still need to find a way to keep the application as available as possible. To achieve this, we can either increase the JVM maximum heap size, or create a script that periodically would end and restart the application server.

Rolling upgrade

Once the application developer provides the new application, the upgrade can be performed as described in “Upgrading an Application” on page 349.

Verification

Finally, you want to prove to yourself that transactions are still able to run at an undiminished rate (or close to it) while doing the various tests you come up with to test your highly available e-business environment. You can do this by having a methodology that you can go back to time and time again. For example:

- Run baseline tests with each workload (see “Tool Used to Drive Workload” on page 347) to establish the “normal” throughput levels (i.e. no interruptions are caused by disabling components of the e-business environment). Run the baseline tests for an appropriate amount of time (e.g. 15 minutes). Throughput levels should then be recorded as transactions per second.

- Then record the average throughput (transactions per second) each time one of the tests is run. Again, the test workload should run for an appropriate amount of time during each test.
- In each case, calculate the average throughput difference from the baseline. Depending on your criteria, ensure that the difference is acceptable.

We think the methodology described above is a good way to prove that the throughput rates can remain close to constant while outages are occurring and recovery was being performed. One thing to remember is that having extra capacity on the servers is very important if your goal is to have the same throughput rate during a server outage. For example, if you have two servers, you would ensure there was some extra capacity on the two servers, so that if one failed, the other could handle the workload at the same throughput rate.

CHAPTER 21

END-TO-END MONITORING

By now you may be thinking “How can I monitor the health of all the components in my infrastructure to make sure that they are active and performing well?”

That's a good question. As we discussed in Chapter 20, “Testing” on page 347, some component failures will be automatically recovered, while most require some intervention on your part. Knowing that the component has gone down is the first problem, since the redundancy built into our architecture is there to ensure that application workloads continue to run even through multiple component failures. Therefore, you need a way of realizing that a component has failed even though application throughput remains high.

We cannot fully address that topic in this document, but we can give you a very brief overview of some of the system monitoring tools that are available today to help.

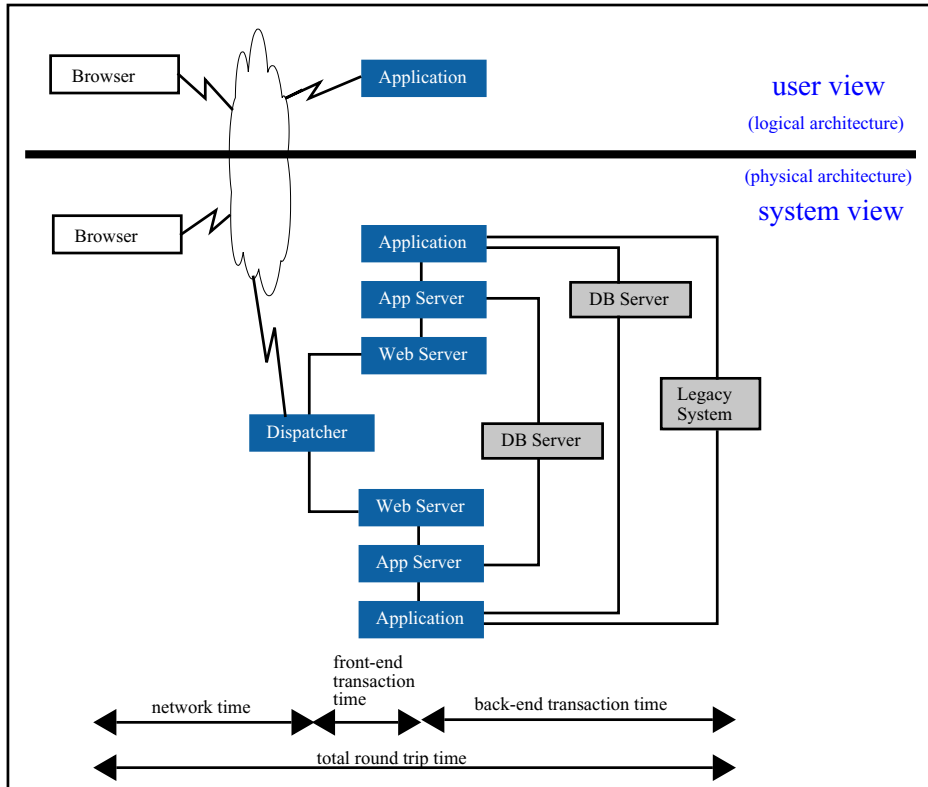
Before going into detail regarding specific products, we need to point out that ensuring the performance and availability of a Web application is not as simple as it may seem. From the user's point of view - the logical one - the architecture is a simple two-tier setup: a browser connected to the Internet talks to a Web server.

Behind the scenes, however, there are many more components involved - even when ignoring all the networking issues. What is perceived as a Web server by a user may very well be implemented as a number of specialized Web servers, application servers and database servers, each providing special parts of the whole application in a direct (or supporting) role.

In a typical implementation, a Web application is hosted by an Application Server (WebSphere Application Server) front-ended by a Web server (IBM HTTP Server). To provide extra capacity for peak loads, as well as backup in case of failure of the primary servers, a second set may be implemented, and to provide load balancing, a WebSphere Edge Server needs to be put in place.

At the back end, the Java application running inside the WebSphere Application Server probably interacts with a database. It may also interface to legacy systems.

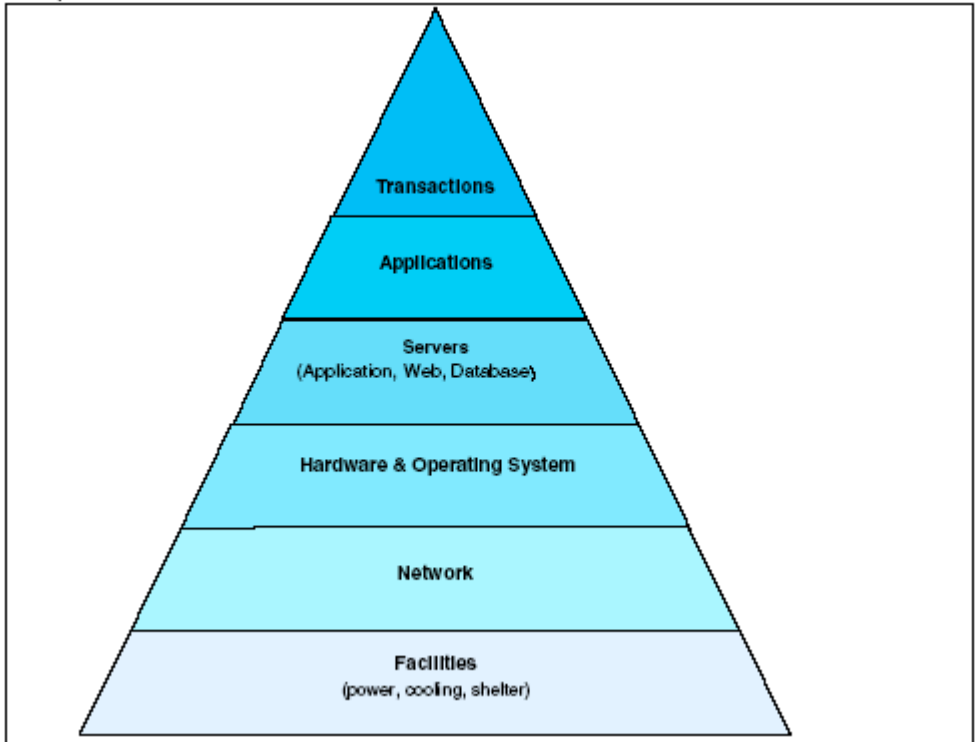
Figure 21.1 Logical versus physical Web application architecture



As depicted in Figure 21.1 on page 354, the complexity of the physical infrastructure is far greater than the complexity of the logical infrastructure. In this graphic, the dark squares represent all the infrastructural components that need to be put in place to facilitate a small Web application with minimum resilience. Web servers, Edge Servers, Application Servers, and a database are all required in this case. However, when discussing systems management, remember all of the servers run on a piece of machinery, controlled by an operating system, and they all require shelter, power, and perhaps cooling. In addition to this, we must have access to a network infrastructure to allow all the components to communicate.

For each layer of components shown in Figure 21.2 on page 355, facilities, network, hardware, operating system, servers and so on, every component within a layer must be monitored to ensure availability, performance, and sufficient capacity of the component in question in order to allow for successful provisioning of the applications that are the reason for putting all this in place.

Figure 21.2 Web application infrastructural layers to be managed



In addition, we need to apply our usual management processes for configuration, authentication, authorization, accounting, backup/recovery, licensing, version control etc. And when discrepancies between actual monitored values and predefined thresholds occur, our normal problem management and change management procedures must also be activated. Furthermore, we need to manage the application performance and business logic, too.

Table 21.1 on page 355 lists companies that provide monitoring tools for such a complex environment.

NOTE We do not have any experience with any of these tools. The list is provided for your convenience.

Table 21.1 List of monitors

Vendor	Product(s)	Use	Web site link
IBM	IBM Tivoli Monitoring	Systems and application monitoring	http://www.ibm.com/software/tivoli/solutions/availability/monitoring/
Quest Software	JProbe	Java performance tuning and monitoring.	http://www.quest.com/
Wily	Introscope	Enterprise Web application management	http://www.wilytech.com/index.html

Glossary

A

access path journaling

A method of recording changes to an access path as changes are made to the data in the database file so that the access path can be recovered automatically by the system.

activation group

A substructure of a job in which Integrated Language Environment (ILE) programs and service programs are activated. This substructure contains the resources necessary to run the program. These resources include: static and global program variables, dynamic storage, temporary data management resources, certain types of exception handlers and ending procedures.

address

The unique code assigned to each device or workstation connected to a network. A standard IP address is a 32-bit address field. This field contains two parts. The first part is the network address; the second part is the host number.

Address Resolution Protocol (ARP)

In TCP/IP, a protocol that dynamically maps between internet and baseband-adapter addresses on a local area network.

advisor

The advisors are a function of the Load Balancer. Advisors collect and analyze feedback from individual servers and inform the manager function.

application client module

A Java archive (JAR) file containing a client for accessing a Java application that executes inside of a client container and can connect to remote or client-side J2EE resources.

Application Response Measurement (ARM)

An Open Group standard composed of a set of interfaces implemented by an ARM agent that provides information on the elapsed time for process hops.

application server

A server program in a distributed network that provides the execution environment for an application program.

ARP

See *Address Resolution Protocol (ARP)*.

ASP

See *auxiliary storage pool (ASP)*.

asynchronous I/O

A series of input/output operations that are being done separately from the process (job) that requested them.

asynchronous messaging

A method of communication based on the Java Message Service (JMS) programming interface.

asynchronous operation

An operation that occurs without a regular or predictable time relationship to a specified event; for example, the calling of an error diagnostic routing that may receive control at any time during the execution of a computer program.

authentication

In computer security, a process that ensures that the identities of both the sender and the receiver of a network transaction are true.

authorization

In computer security, the right granted to a user to communicate with or make use of a computer system or resource.

auxiliary storage pool (ASP)

One or more storage units that are defined from the storage devices or storage device subsystems that make up auxiliary storage. An ASP provides a way of organizing data to limit the impact of storage-device failures and to reduce recovery time.

B

Backup Recovery and Media Services (BRMS)

An IBM licensed program that provides user-modifiable backup, archive, recovery, and media management functions and policies.

bootstrapping

Process by which an initial reference of the naming service is obtained. The bootstrap setting and the host name form the initial context for Java Naming and Directory Interface (JNDI) references.

BRMS

See *Backup Recovery and Media Services (BRMS)*.

bytecode

Intermediate code that is generated by the Java compiler. The code must be interpreted or translated to run on a specific platform or processor.

C

Caching Proxy

A caching proxy server that can help speed up end-user response time through highly-efficient caching schemes. Flexible PICS filtering helps network administrators control access to Web-based information at one central location.

CBR

Content Based Routing. A component of Load Balancer. CBR works with Caching Proxy to load balance incoming requests, based on Web page content using specified rule types, to HTTP or HTTPS servers.

cell

An arbitrary, logical grouping of one or more nodes in a WebSphere Application Server distributed network.

cell-scoped binding

A binding scope where the binding is not specific to, and not associated with any node or server..

CGI

See *common gateway interface (CGI)*.

CGI program

A program that uses the common gateway interface (CGI) to perform tasks that are not usually done by the server, such as database access and form processing.

CGI script

A program that uses the common gateway interface (CGI) to perform tasks that are not usually done by the server, such as form processing. A CGI script is typically written in a programming language that is interpreted, such as Perl or Net.Data.

checkpoint

A place in a computer program at which a check is made, or at which a recording of data is made to allow the program to be restarted at a later time.

checksum protection

A function that protects data stored in an auxiliary storage pool from being lost because of the failure of a single disk. When checksum protection is in effect and a disk failure occurs, the system automatically reconstructs the data when the system program is loaded after the device is repaired.

class loader

Part of the Java virtual machine code that is responsible for finding and loading class files. A class loader affects the packaging of applications and the run-time behavior of packaged applications deployed on application servers.

cluster

(1) A collection of complete systems that work together to provide a single, unified computing capability. An iSeries cluster is made up of only iSeries servers. (2) In WebSphere, a group of application servers that collaborates for the purposes of workload balancing and failover.

cluster membership list

A set of cluster nodes that have been configured for a cluster.

cluster resource

For OS/400, any part of the system that is available across multiple cluster nodes (e.g. resilient application and its associated IP address, which can be switched).

cluster resource group

For OS/400, a collection of related cluster resources that defines actions to be taken during a switch-over operation of the access point of resilient resources. The group describes a recovery domain and supplies the name of the cluster resource group exit program that manages the movement of an access point.

commitment control

A means of grouping committable resource operations to allow either the processing of a group of committable resource changes as a single unit through a commit operation, or the removing of a group of committable resource changes as a single unit through the rollback operation.

common gateway interface (CGI)

A standard for the exchange of information between a Web server and computer programs that are external to it. The external programs can be written in any programming language that is supported by the operating system on which the Web server is running.

Common Object Request Broker (CORBA)

An OMG specification for application interoperability independent of platform, programming language, and protocol.

configured name binding

Persistent storage of an object in the name space that is created using either the administrative console or the wsadmin program.

connection factory

Used by an application component to access a connection instance, which the component then uses to connect to the underlying Enterprise Information System (EIS).

connection handle

A representation of a physical connection.

connection pooling

A technique used for establishing a pool of resource connections that applications can share on an application server.

connector

A portable service API to external resources.

container.

In J2EE, an entity that provides life-cycle management, security, deployment, and run-time services to components.

container-managed persistence (CMP)

In J2EE technology, a data transfer between the variables of an entity bean and a resource manager administered by the entity bean container. (Sun)

container transaction

A transaction that has its boundaries set by the container for method invocations of the enterprise bean.

content based routing (CBR)

An optional feature of the caching proxy that provides intelligent routing to back-end application servers. This routing is based on HTTP session affinity and a weighted round-robin algorithm.

CORBA

See *Common Object Request Broker Architecture (CORBA)*.

D**data access bean**

A class library that provides a rich set of features and functions, while hiding the complexity associated with accessing relational databases.

demilitarized zone (DMZ)

A configuration including multiple firewalls to add layers of protection between a corporate intranet and a public network, like the Internet.

deployment descriptor

An Extensible Markup Language (XML) file that describes how to deploy a module or application by specifying configuration and container options.

device parity protection

A function that protects data stored on a disk unit subsystem from being lost because of the failure of a single disk unit in the disk unit subsystem. When a disk unit subsystem has device parity protection and one of the disk units in the subsystem fails, the system continues to run. The disk unit subsystem reconstructs the data after the disk unit in the subsystem is repaired or replaced.

Dispatcher

A component of Load Balancer that efficiently balances TCP or UDP traffic among groups of individual linked servers. The Dispatcher machine is the server running the Dispatcher code.

Domain Name Server (DNS)

A general-purpose distributed, replicated, data query service chiefly used on Internet for translating hostnames into Internet addresses. Also, the style of hostname used on the Internet, though such a name is properly called a fully qualified domain name. DNS can be configured to use a sequence of name servers, based on the domains in the name being looked for, until a match is found.

E**EJB container**

Provides a run-time environment for enterprise beans within the application server. Handles all aspects of enterprise bean operation within the application server and acts as an intermediary between the user-written business logic within the bean and the rest of the application server environment.

EJB module

Assembles one or more enterprise beans into a single deployable unit. An EJB module is stored in a standard Java archive (JAR) file.

enterprise application

An application that conforms to the Java 2 Platform Enterprise Edition specification.

enterprise archive (EAR)

A specialized Java archive (JAR) file, defined by the J2EE standard used to deploy J2EE applications to J2EE application servers. An EAR file contains enterprise beans, a deployment descriptor, and Web archive (WAR) files for individual Web applications.

enterprise bean

A Java component that can be combined with other resources to create J2EE applications. There are three types of enterprise beans: entity beans, session beans, and message-driven beans.

Enterprise Information system (EIS)

Applications that provide an information infrastructure for an enterprise.

entity bean

An enterprise bean that represents persistent data maintained in a database. Identified by a primary key.

environment variable

A variable that specifies how an operating system or another program runs, or the devices that the operating system recognizes.

exit program

A user-written program that is given control during operation of a system function.

F**failover**

An event where the primary system/service switches over to a backup system/service due to the failure of the primary system/service.

federation

Process of hooking together naming systems so that the aggregate system can process composite names that span the naming systems.

Firewall

A computer that connects a private network, such as a business, to a public network, such as the Internet. It contains programs that limit the access between two networks. See also *proxy gateway*.

G**garbage collection**

A routine that searches memory to reclaim space from program segments or inactive data.

gateway

A middleware component that bridges Internet and intranet environments during Web service invocations.

General Inter-ORB Protocol (GIOP)

A protocol that Common Object Request Broker Architecture (CORBA) uses to define the format of messages.

H**high-speed link (HSL)**

A hardware connectivity architecture that links system processors to system input/output buses and other system units.

horizontal scaling

A topology in which more than one application server running on multiple computing nodes is used to run a single application.

hot deployment

Process of adding new components to a running server without stopping and restarting the application server or application.

HSL

See *high-speed link (HSL)*.

Hypertext Transfer Protocol (HTTP)

An Internet protocol that is used to retrieve hypertext objects from remote hosts.

Hypertext Transfer Protocol Secure (HTTPS)

A TCP/IP protocol that is used by World Wide Web servers and Web browsers to transfer and display hypermedia documents securely across the Internet.

I**independent ASP**

See *independent disk pool*.

independent disk pool

One or more storage units that are defined from the disk units or disk-unit subsystems that make up addressable disk storage. An independent disk pool contains objects, the directories that contain the objects, and other object attributes such as authorization ownership attributes. An independent disk pool can be made available (varied on) and made unavailable (varied off) without restarting the system. An independent disk pool can be either a) switchable among multiple systems in a clustering environment or b) privately connected to a single system.

initial context

Starting point in a namespace.

Internet Inter-ORB Protocol (IIOP)

A TCP/IP-based protocol that Common Object Request Broker Architecture (CORBA) uses to encode and decode General Inter-ORB Protocol (GIOP) messages.

Interoperable object reference (IOR)

An object reference with which an application can make a remote method call on a CORBA object. This reference contains all the information needed to route a message directly to the appropriate server.

J**J2EE application**

Any deployable unit of J2EE functionality. This unit can be a single module or a group of modules packaged into an enterprise archive (EAR) file with a J2EE application deployment descriptor. (Sun)

J2EE Connector architecture

A standard architecture for connecting the J2EE platform to heterogeneous enterprise information systems (EIS).

J2EE server

A run-time environment that provides enterprise bean or Web containers.

Java

An object-oriented programming language for portable interpretive code that supports interaction among remote objects. The Java language was developed and specified by Sun Microsystems, Incorporated.

Java Authentication and Authorization Service (JAAS)

A package through which services can authenticate and authorize users while enabling the applications to remain independent from underlying technologies.

Java Command Language (Jacl)

A scripting language for the Java 2 environment that is used to create Web content and to control Java applications.

Java Message Service (JMS)

A Java API that supports the creation and communication of various messaging implementations.

Java Naming and Directory Interface (JNDI)

A Java extension that provides an interface for various directory and naming services in an enterprise.

Java Runtime Environment (JRE)

A subset of the Java Software Development Kit (SDK) that contains the core executables and files that constitute the standard Java platform. The JRE includes the Java virtual machine (JVM), core classes, and supporting files.

Java virtual machine (JVM)

An interpretive computing engine responsible for executing the byte code in a compiled Java program into the native instructions of the host machine.

Java Virtual Machine Profiler Interface (JVMPi)

A profiling tool that supports the collection of information, such as data about garbage collection and the Java virtual machine (JVM) API that runs the application server.

JavaServer Pages (JSP) files

Application building blocks coded to the Sun Microsystems JavaServer Pages (JSP) specification. JSP files enable the separation of the Hypertext Markup Language (HTML) code from the business logic in Web pages so that HTML programmers and Java programmers can collaborate when creating and maintaining pages.

Java 2 Connector security.

An architecture designed to extend the end-to-end security model for J2EE-based applications to include enterprise information systems (EIS)

JDBC

An API that supports database and data source access from Java applications.

journal

A system object that identifies the objects being journaled, the current journal receiver, and all the journal receivers on the system for the journal.

journal entry

A record in a journal receiver that contains information about a journaled change or other activity that is journaled.

journaling

The process of recording, in a journal, the changes made to objects, such as physical file members or access paths, or the depositing of journal entries by system or user functions.

journal receiver

A system object that contains journal entries added when events occur that are journaled, such as changes to a database file, changes to other journaled objects, or security-relevant events.

JVM

See *Java virtual machine (JVM)*.

L**Lightweight Third Party authentication (LTPA).**

A protocol that uses cryptography to support security in a distributed environment.

load balancing

The monitoring of application servers and management of the workload on servers.

logical partition

A subset of a single iSeries server that contains resources (processors, memory, and input/output devices). A logical partition operates as an independent system. If hardware requirements are met, multiple logical partitions can exist within a system.

logical partitioning (LPAR)

A function of the OS/400 licensed program that enables the creation of logical partitions. See *logical partition*.

loopback alias

An alternative IP address associated with the loopback interface. The alternative address has the useful side affect of not advertising on a real interface.

loopback interface

An interface that bypasses unnecessary communications functions when the information is addressed to an entity within the same system.

M**medium access control (MAC)**

For local area networks, the method of determining which device has access to the transmission medium at any time.

message-driven bean

An enterprise bean that provides asynchronous message support and clearly separates message and business processing.

mirrored protection

A function that protects data by duplicating all disk data in an auxiliary storage pool (ASP) to another disk unit (mirrored unit) in the same ASP. If a disk failure occurs, the system keeps running, using the operational mirrored unit of the mirrored pair until the disk unit is repaired or replaced.

mirroring

The process of writing the same data to two disk units within the same auxiliary storage pool at the same time. The two disk units become a mirrored pair, allowing the system to continue when one of the mirrored units fails.

N**naming contexts**

A logical namespace containing name and object bindings.

naming federation

Process of hooking together naming systems so that the aggregate system can process composite names that span the naming systems.

NAT

See *network address translation (NAT)*.

Net.Data

A program that allows you to create interactive Web applications by using scripts to generate dynamic Web documents. Net.Data is part of the IBM HTTP Server licensed program.

network address translation (NAT)

(1) The conversion of a network address that is assigned to a logical unit in one network into an address in an adjacent network. (2) In a firewall, the conversion of secure Internet Protocol (IP) addresses to external registered addresses. This enables communications with external networks but masks the IP addresses that are used inside the firewall.

NIC

Network Interface Card. An adapter circuit board installed in a computer to provide a physical connection to a network.

node

A logical grouping of managed servers.

node agent

Manages all application servers on a node and represents the node in the management cell.

node name

The machine name of the installation platform; an arbitrary WebSphere Application Server-specific name that must be unique.

node federation

Process of combining the managed resources of one node into a distributed network such that the central manager application can access and administer the resources on the node.

O**object adapter**

A Common Object Request Broker Architecture (CORBA) term, denoting the primary interface that a server implementation uses to access Object Request Broker (ORB) functions.

object reference

A Common Object Request Broker Architecture (CORBA) term, denoting the information needed to reliably identify a particular object.

Object Request Broker (ORB)

In object-oriented programming, software that serves as an intermediary by transparently enabling objects to exchange requests and responses.

P**Performance Monitoring Infrastructure (PMI)**

A set of packages and libraries assigned to gather, deliver, process, and display performance data.

persistence

A characteristic of data that is maintained across session boundaries, or of an object that continues to exist after the execution of the program or process that created it, usually in nonvolatile storage, such as a database system.

PICS

Platform for Internet Content Selection. PICS-enabled clients allow the users to determine which rating services they want to use and, for each rating service, which ratings are acceptable and which are unacceptable

plug-in

A self-contained software component that modifies function in a particular software system. When a user adds a plug-in to a software system, the foundation of the original software system remains intact. The development of plug-ins requires well-defined application programming interfaces (APIs).

point-to-point

A style of messaging application in which the sending application knows the destination of the message.

R**Redundant Array of Independent Disks (RAID)**

A method of protecting data loss due to disk failure based on the Redundant Array of Independent Disks specification published by the University of California in 1987.

replication

The process of copying objects from one node in a cluster to one or more other nodes in the cluster, which makes the objects on all the systems identical.

resilient application

A server program that can be restarted on a different node without requiring a user to reconfigure the clients.

resilient resource

Data, a process, or an application that can be recovered if a node in a cluster fails.

resource adapter

A system-level software driver that a Java application uses to connect to an Enterprise Information System (EIS).

resource adapter archive (RAR)

A Java archive (JAR) file that is used to package a resource adapter for the Java 2 Connector (J2C) architecture for WebSphere Application Server.

reverse proxy

An IP-forwarding topology where the proxy is on behalf of the back-end HTTP server. It is an application proxy for servers using HTTP.

S**SAN**

See *storage area network (SAN)*.

save-while-active operation

An operation that the user runs to save objects while application programs that change the objects are running.

Secure Sockets Layer (SSL)

A security protocol that provides transport layer security: authenticity, integrity, and confidentiality, for a secure connection between a client and a server. The protocol runs above TCP/IP and below application protocols.

servlet

A Java program that runs on a Java-enabled Web server and extends the capabilities of a Web server, similarly to the way applets run on a browser and extend the capabilities of a browser.

session

A series of requests to a servlet originating from the same user at the same browser.

session affinity

Application configurations where a client is always connected to the same server. These configurations disable workload management after an initial connection by forcing a client request to always go to the same server.

session bean

An enterprise bean that is created by a client and that usually exists only for the duration of a single client and server session.

Simple Object Access Protocol (SOAP)

A lightweight, XML-based protocol for exchanging information in a decentralized, distributed environment.

stateful session bean

A session bean that enables clients to rely on a single point of contact on the server, manages persistence for long-running sessions and reduces traffic between the client and server.

stateless session bean

A session bean that is a collection of operations. The server can optimize resources by reusing bean instances on every method call.

sticky time

The interval between the closing of one connection and the opening of a new connection during which a client will be sent back to the same server used during the first connection. After the sticky time, the client may be sent to a server different from the first.

storage area network (SAN)

The connectivity of multiple systems that attach to a single storage device.

subnet mask

For Internet subnetworking, a 32-bit mask used to identify the subnetwork address bits in the host portion of an IP address.

switchover

An event where the primary server/service switches over to a backup server/service.

T**thin application client**

A lightweight, downloadable Java application run time capable of interacting with enterprise beans.

thin client

A system that runs a light operating system with no local system administration and executes applications over the network.

thread

The basic unit of program execution. Several threads can run concurrently, performing different jobs.

Tivoli Performance Viewer

A Java client that retrieves the Performance Monitoring Infrastructure (PMI) data from an application server and displays it in various formats.

transaction

A specific set of input data that triggers execution of a specific processor job; a message destined for an application program.

transport

The request queue between a WebSphere Application Server plug-in for Web servers and a Web container in which the Web modules of an application reside. When a user at a Web browser requests an application, the request is passed to the Web server, then along the transport to the Web container.

U**Uniform Resource Identifier (URI)**

A compact string of characters for identifying an abstract or physical resource.

Uniform Resource Locator (URL)

An identifier that points to an electronically-accessible resource, such as a directory file on a machine in a network, or a document stored in a database.

V**vertical scaling**

Setting up multiple application servers on one machine, usually by creating cluster members.

virtual host

A configuration enabling a single host machine to resemble multiple host machines. Resources associated with one virtual host cannot share data with resources associated with another virtual host, even if the virtual hosts share the same physical machine.

virtual machine

An abstract specification for a computing device that can be implemented in different ways in software and hardware.

W**WAN**

Wide Area Network. A network that provides communication services to a geographic area larger than that served by a local area network or a metropolitan area network, and that may use or provide public communication facilities.

Web application

An application comprised of one or more related servlets, JavaServer Pages technology, and HyperText Markup Language (HTML) files that you can manage as a unit.

Web archive (WAR)

A compressed file format, defined by the J2EE standard for storing all the resources required to install and run a Web application in a single file.

Web component

A servlet, JavaServer Page (JSP) file, or a HyperText Markup Language (HTML) file. One or more Web components make up a Web module.

Web container

Handles requests for servlets, JavaServer Pages (JSP) files, and other types of files that include server-side code. It creates servlet instances, loads and unloads servlets, creates and manages request and response objects, and performs other servlet management tasks.

Web module

Represents a Web application. A Web module is created by assembling servlets, JavaServer Pages (JSP) files, and static content such as HyperText Markup Language (HTML) pages into a single deployable unit. Web modules are stored in Web archive (WAR) files, which are standard Java archive (JAR) files.

Web server plug-in

Supports the Web server in communicating requests for dynamic content, such as servlets, to the application server.

Web service

A self-contained, modular application that you can use to describe, publish, locate, and invoke over a network.

workload management

The optimization of the distribution of incoming work requests to the application servers, enterprise beans, servlets and other objects that can effectively process the request.

