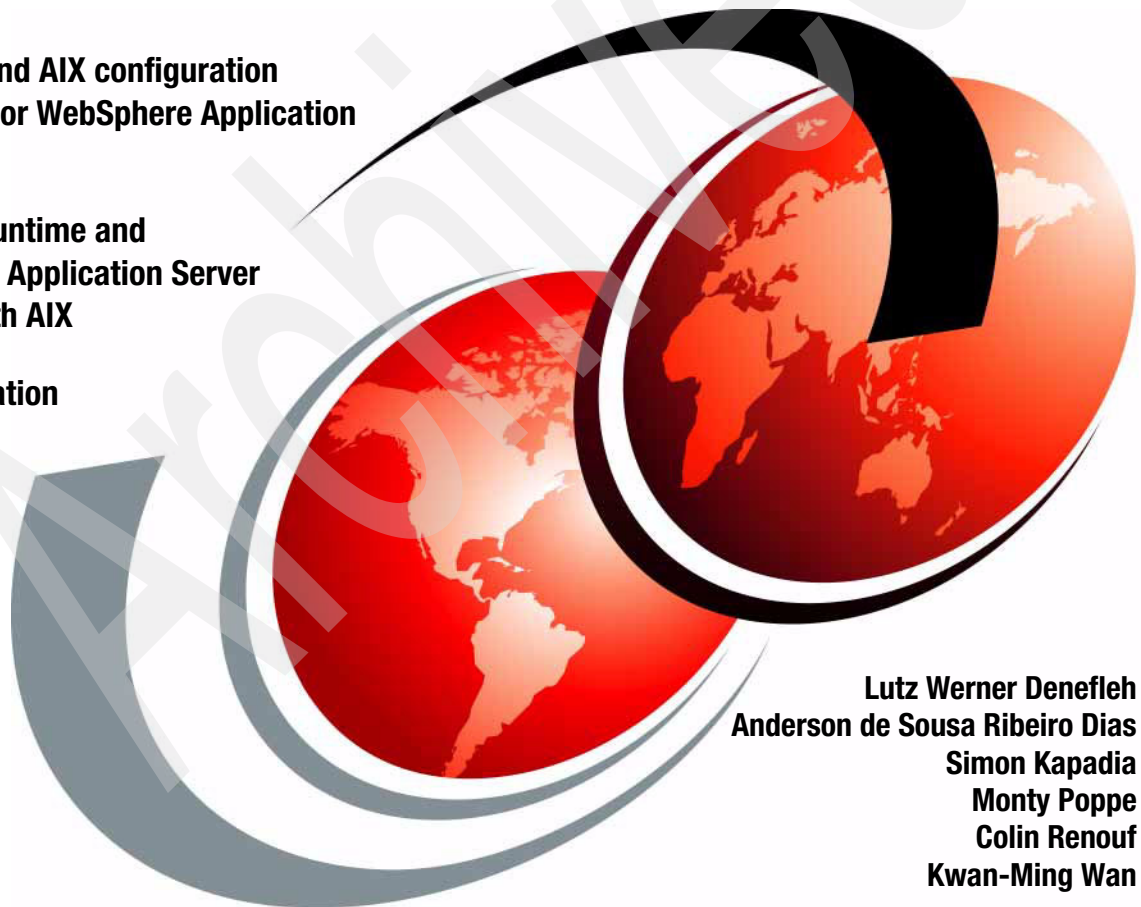


Running IBM WebSphere Application Server on System p and AIX: Optimization and Best Practices

System p and AIX configuration strategies for WebSphere Application Server

How JVM runtime and WebSphere Application Server interact with AIX

Implementation scenarios



Lutz Werner Denefleh
Anderson de Sousa Ribeiro Dias
Simon Kapadia
Monty Poppe
Colin Renouf
Kwan-Ming Wan



International Technical Support Organization

**Running IBM WebSphere Application Server on
System p and AIX: Optimizaton and Best Practices**

September 2008

Archived

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

First Edition September 2008

This edition applies to IBM WebSphere Application Server Version 6.1, IBM AIX Version 5.3, and IBM AIX Version 6.1.

© Copyright International Business Machines Corporation 2008. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	ix
Trademarks	x
Preface	xi
The team that wrote this book	xi
Acknowledgements	xiii
Become a published author	xiii
Comments welcome	xiv
Chapter 1. Introduction to running WebSphere Application Server on System p and AIX	1
1.1 The whole system view: WebSphere, JVM, AIX, and System p	2
1.1.1 Points of view	2
1.1.2 A holistic system approach	3
1.2 System layers and points of view	3
1.2.1 Points of view and terminology	4
1.3 The remainder of this book	5
Chapter 2. WebSphere on System p and AIX 5 strategies	7
2.1 Scalability considerations	8
2.1.1 Clustering	8
2.1.2 Workload management	15
2.2 Session persistence considerations	17
2.3 File store considerations	19
2.4 Install automation considerations	20
2.4.1 Silent installation	20
2.4.2 Installation Factory	21
2.5 JVM tuning	22
2.5.1 Memory management	22
2.5.2 CPU performance	23
2.6 Extended Deployment (XD) considerations	24
2.6.1 Dynamic operations: WebSphere Virtual Enterprise	24
2.6.2 Extended manageability	30
2.6.3 High performance computing: WebSphere eXtreme Scale	32
Chapter 3. System p platform configuration	37
3.1 Setting up System p hardware and partitions	38
3.1.1 Setting up and accessing the Hardware Management Console	38
3.1.2 Basic managed system operation	46

3.1.3 Basic partition operations	49
3.1.4 Advanced partition operations.	55
3.1.5 Dynamic LPAR assignments and partition profiles	64
3.1.6 Virtual I/O Server virtualization configuration	65
3.2 Provisioning	84
3.2.1 Methods	85
3.2.2 Provisioning at the operating system level	85
3.2.3 Accessing the CSM through WebSM	103
3.2.4 Using provisioning at the hardware level	125
3.2.5 Using the provisioning features.	126
Chapter 4. AIX configuration	127
4.1 Asynchronous I/O capabilities for sockets and files	128
4.2 AIX Release Content List package information.	131
4.3 AIX base operating system samples	132
4.4 AIX-specific startup and runtime settings	133
4.5 AIX-specific Java environment settings.	133
4.6 AIX TCP/IP network settings	135
4.7 WebSphere Server start and stop.	137
Chapter 5. WebSphere Application Server on AIX: under the hood.	141
5.1 Java and J2EE	143
5.1.1 Java.	143
5.1.2 J2EE	144
5.2 Application Server 6.1 on AIX	147
5.2.1 WebSphere Application Server and the JMS Messaging Engine	147
5.2.2 Process structure	150
5.2.3 WebSphere Application Server on AIX - disk layout.	151
5.2.4 Application server configuration	157
5.2.5 Key server configuration files	163
5.2.6 server.xml	163
5.2.7 serverindex.xml	175
5.2.8 security.xml	178
5.3 IBM J9 Java Virtual Machine Architecture.	189
5.3.1 Generic Java Virtual Machine overview	189
5.3.2 IBM J9 JVM.	193
5.3.3 IBM J9 JVM internal implementation.	211
5.4 WebSphere Application Server architecture	232
5.4.1 Overview of WebSphere Application Server architecture	233
5.4.2 Eclipse 3.1.2 and OSGI/Equinox Runtime	239
5.4.3 WebSphere Application Server-specific JNI native shared object libraries	243
5.4.4 WebSphere Application Server - startup and operation	244

5.5	WebSphere Application Server high availability deployments	276
5.5.1	WebSphere cluster and cell configuration	279
5.5.2	IBM HTTP Server and the IBM WebSphere Application Server plug-in	282
5.5.3	WebSphere Application Server clustering	289
5.5.4	WebSphere HAManager	292
5.5.5	WebSphere Application Server, WebSphere MQ Server and HACMP	298
5.5.6	The WebSphere Application Server database tier	299
5.5.7	WebSphere Application Server ND versus WebSphere Application Server XD on System p	299
Chapter 6.	Tuning the IBM Java Virtual Machine	303
6.1	The importance of JVM tuning	304
6.1.1	Overview of JVM tuning capabilities	304
6.2	Choosing a garbage collection policy	305
6.2.1	Java memory management: garbage collection and allocation	306
6.2.2	Optimal Throughput GC policy	307
6.2.3	Optimal Average Pause GC policy	310
6.2.4	Generational Concurrent GC policy	310
6.2.5	Subpool GC policy	313
6.2.6	Additional runtime options	313
6.3	The large object area	314
6.4	Heap sizing	316
6.4.1	Analyzing verbose output	316
6.4.2	Determining initial heap size	319
6.4.3	Determining maximum heap size	321
6.4.4	Expansion and contraction	323
6.5	Using shared classes	325
6.5.1	Creating multiple caches	326
6.5.2	Tuning the shared class cache size	327
6.6	Using large page sizes	329
6.6.1	Large page support in AIX	329
6.6.2	64 KB size for POWER5+ and later systems	330
6.6.3	Java large page support	330
6.6.4	Changing page sizes in WebSphere Application Server	330
6.7	Dynamic logical partitions	333
6.8	Just-in-Time compiler	334
Chapter 7.	High availability, clustering and WebSphere	335
7.1	Clustering WebSphere for availability	336
7.1.1	Availability considerations	336
7.1.2	Clustering options	337

7.2	WebSphere clustering	337
7.3	High availability on logical partitions	339
7.3.1	Isolation	339
7.3.2	Ease of use	340
7.3.3	Redundancy	341
7.4	Disaster recovery	342
7.4.1	Environment configuration	343
7.4.2	Planning for disaster	345
7.4.3	Disaster event and recovery	346
7.4.4	Process and procedure	347
7.5	HACMP	348
7.5.1	HACMP, logical partitions and WebSphere	348
7.5.2	How HACMP works	349
7.5.3	WebSphere HACMP configuration	349
7.5.4	HACMP on DLPARs and shared CPU LPARs	353
7.6	Lifecycle management and upgrades	354
7.6.1	The problem	355
7.6.2	The solution	355
7.6.3	Unlimited hardware	356
Chapter 8. Clustering WebSphere Application Server for performance		361
8.1	Clustering for performance overview	362
8.1.1	Clustering options	362
8.2	WebSphere clustering on micropartitions	362
8.2.1	Isolation	363
8.3	AIX Partition Load Manager	364
8.3.1	PLM capabilities	364
8.3.2	AIX PLM configuration	365
8.4	AIX Workload Management feature	368
8.4.1	WLM capabilities	368
8.4.2	Co-locating WebSphere applications	369
8.4.3	AIX WLM configuration	369
8.4.4	Options for WebSphere service workload management	373
8.5	Sharing the technologies	375
Chapter 9. AIX 5L and WebSphere XD		377
9.1	WebSphere XD and DLPAR integration	378
9.1.1	Dynamic operations	378
9.2	Dynamic reconfiguration manager	380
9.2.1	Processor DR events	380
9.2.2	Memory DR events	381
9.3	Performance and scalability with DLPAR	381
9.3.1	WebSphere Partitioning Facility with DLPAR	381

Chapter 10. Implementation scenario	385
10.1 Scenario overview	386
10.1.1 Software products	386
10.1.2 The sample WebSphere-related scenario	388
10.2 Installation summary	391
10.2.1 System description	392
10.3 Installing WebSphere Application Server	393
10.3.1 Installing the WebSphere Application Server 6.1 Fix Pack 2	397
10.4 Installing and configuring Trade 6.1	397
10.4.1 Trade 6.1 installation summary	398
10.4.2 Download the Trade 6.1 installation package	399
10.4.3 Set up and configure the tradedb database	399
10.4.4 Install Trade 6.1 using the installation script	400
10.4.5 Working with Trade 6.1	402
10.5 Performance testing	405
10.5.1 General application performance testing requirement	405
10.5.2 Scenario overview	406
10.5.3 IBM Rational Performance Tester	407
10.5.4 Scenario testing	415
Appendix A. Sample files	423
WebSphere: responsefile.nd.txt	424
Update Installer: response.txt	452
WebSphere V6.1 Fix Pack 2: fp2.response.txt	455
Trade 6.1 Installation script: trade.jacl	456
CSM adapter definition file: p550q_lpar_adapters	471
Appendix B. Additional material	473
Locating the Web material	473
Using the Web material	473
How to use the Web material	474
Related publications	475
IBM Redbooks publications	475
Other publications	476
How to get IBM Redbooks	477
Help from IBM	477
Index	479

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 illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.


Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® and ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

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

AIX 5L™
AIX®
Alerts®
alphaWorks®
AS/400®
BladeCenter®
CICS®
DB2®
developerWorks®
DFS™
Encina®
eServer™
HACMP™
IBM®

IMS™
iSeries®
Lotus®
Micro-Partitioning™
POWER™
POWER Hypervisor™
POWER3™
POWER4™
POWER5™
POWER5+™
POWER6™
PowerVM™
pSeries®
Rational Rose®

Rational®
Redbooks®
Redbooks (logo) ®
RS/6000®
S/390®
System p™
System p5™
Tivoli®
Virtualization Engine™
WebSphere®
z/OS®
zSeries®

The following terms are trademarks of other companies:

NetApp, and the NetApp logo are trademarks or registered trademarks of NetApp, Inc. in the U.S. and other countries.

SUSE, the Novell logo, and the N logo are registered trademarks of Novell, Inc. in the United States and other countries.

Oracle, JD Edwards, PeopleSoft, Siebel, and TopLink are registered trademarks of Oracle Corporation and/or its affiliates.

SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

EJB, J2EE, J2SE, Java, Java runtime environment, JDBC, JDK, JMX, JNI, JSP, JVM, Sun, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Active Directory, Internet Explorer, SQL Server, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

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

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

Preface

This IBM® Redbooks® publication describes how to run the IBM Java™ Virtual Machine for AIX® and WebSphere® Application Server V6.1 on IBM System p™ and the AIX 5L™ Operating System. In terms of provisioning, tuning and maintenance, it consolidates information from all of these areas into a single resource and explains how you can implement, tune, and utilize the unique features of the IBM POWER™ Systems platform, AIX, and WebSphere Application Server together for maximum optimization.

The book is intended for UNIX® system administrators, Java developers, infrastructure designers, J2EE™ architects, project managers, performance testers and anyone who runs WebSphere Application Server on System p and AIX. It may contain some information which you already know, and other information that is new to you, depending on your background.

For example, AIX system administrators may be expert in configuring logical partitions and advanced virtualization, but may gain an understanding from this book about how WebSphere deployment teams may be able to exploit the features of IBM POWER Systems and AIX. WebSphere infrastructure architects may already know exactly how they want their redundant systems to work, but might learn how AIX teams can provide two or three physical servers that provide all of the different levels of application services necessary for the entire application lifecycle environment.

This book will build on the skills that you already possess, and give you an understanding of the skills of those you are working with, thereby enabling you to develop a better system that is optimized for running WebSphere Application Server and your own J2EE applications on IBM POWER Systems.

Note: Readers may also be interested in the companion IBM Redbooks publication *Case Study: AIX and WebSphere in an Enterprise Infrastructure*, REDP-4436, which is available at the following location:

<http://www.redbooks.ibm.com/abstracts/redp4436.html>

The team that wrote this book

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

Lutz Werner Deneffleh is a team leader at the PLM Technical Support Eastern Region Support Center in Mainz, Germany. He holds a Graduate Engineer degree in Fluid Dynamics of the FHD - University of Applied Sciences Darmstadt. He has 18 years of experience in Information Technology, and has worked at IBM for 17 years. His areas of expertise include solution implementations on RS/6000®, such as CATIA, ENOVIA, Lotus® Notes/Domino, Tivoli®, DCE/DFS™, and Internet technologies. Lutz is currently responsible for the IBM internal infrastructure used by the Product Lifecycle Management (PLM) World Wide Technical Support.

Anderson de Sousa Ribeiro Dias is an IT Specialist with IBM Brazil, supporting AIX and Infrastructure teams internationally. He has seven years of experience in IT and has worked at IBM for two years. Anderson holds a degree in Computer Science from Universidade Metodista de Sao Paulo.

Simon Kapadia is the Security Lead for IBM Software Services for WebSphere (ISSW) in EMEA (Europe, Middle East, Africa), and works on designing and implementing large distributed computer systems for IBM customers. He holds a Bachelor's degree in English and Drama and a Master's degree in Computing Science. Prior to joining IBM, Simon developed software for digital exchanges at Bell Laboratories, managed networks and Web applications at an ISP, and supported and consulted on DCE, DFS, and Encina® for Transarc Corporation.

Monty Poppe is a Software Engineer in IBM Systems & Technology Group Systems Assurance in Austin, Texas, where he is responsible for testing the “whole stack”, including AIX, System p, and IBM's Java. He has worked for IBM for nine years and was previously a Software Developer and Consultant. Monty holds a degree in Computer Science from Texas A&M University.

Colin Renouf is a Solutions and Infrastructure Architect who works for a large UK-based bank. He has about 25 years of experience in IT and was formerly an Aeronautical Engineer with an aircraft engines company, specializing in Windows® development and infrastructure. His areas of expertise include AIX, Java/J2EE, and WebSphere Application Server. As an active member of the user communities, Colin runs some of the largest user groups in the world, notably that for AIX and jointly that for all things WebSphere. He holds a BSc degree in Aeronautical Engineering and a BA in IT and Social Sciences, and has studied a number of other subjects at degree level.

Kwan-Ming Wan is a Consulting IT Specialist working for IBM Software Services for WebSphere in London, UK. He has more than 18 years of experience in IT, and has worked as a consulting professional throughout his career. For the past eight years, he has been a WebSphere consultant with a focus on performance tuning, problem determination, and architecture design. Prior to joining IBM, Kwan-Ming was a consultant at Transarc Corporation working on large scale distributed transactional systems with DCE and Encina. He holds a Master of

Science degree in Information Technology from the University of Nottingham, England.

The Project Leader who managed the development of this IBM Redbooks publication:

Chris Almond is a Project Leader and IT Architect based at the IBM ITSO Center in Austin, Texas. Currently he specializes in managing content development projects focused on Linux®, AIX and System p systems engineering, various IBM Software Group topics, and IBM innovation programs. He has 17 years of experience in IT, and has worked for IBM for nine years.

Acknowledgements

The authoring team acknowledges the following individuals for their efforts in supporting the development and review process for this book:

Gerry Coumo
David Currie
Tim Francis
Henning Gammelmark
Trent Gray
Greg Trutty
Tim Vanderham
John Zucker

Become a published author

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

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

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

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

ibm.com/redbooks

- ▶ Send your comments in an email to:

redbooks@us.ibm.com

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

Introduction to running WebSphere Application Server on System p and AIX

There are many IBM Redbooks publications available that focus on System p and AIX systems engineering. Those books are written for hardware specialists, system specialists, system architects, and operating system-level system administrators. There are also many Redbooks that focus on various aspects of the IBM WebSphere Application Server platform. Those books are written for application developers, J2EE software engineers, and application-level administrators. This book is different. In it, we present a whole systems viewpoint, focused specifically on end-to-end system deployment, tuning, and management methods for environments running WebSphere Application Server loads on System p and AIX. Thus, in this book we bridge the gap between two different technical audiences, namely hardware and operating system administrators, and WebSphere Application Server application software engineers. We acknowledge that in a typical enterprise environment, each of these technical audiences will work closely together but still have different perspectives and responsibilities. But, the ultimate measure of how well an enterprise is leveraging its investments in WebSphere Application Server running on System p and AIX will depend on how well the overall system architects understand how to leverage the unique strengths of each product, *together*. So, we begin this book with a clarification of “points of view”.

1.1 The whole system view: WebSphere, JVM, AIX, and System p

Table 1-1 lists levels of abstraction of software and hardware that together comprise a Web-based system.

Table 1-1 Layers in the whole system view

Enterprise Java Code
WebSphere Application Server
Java Virtual Machine
AIX Operating System
Logical Partition
VIO Server
Firmware
System P Hardware

1.1.1 Points of view

Referring to Table 1-1, the firmware for the hard disk is called by the VIO Server, and from the perspective of the hard disk, that is the application calling the firmware. A firmware developer would provide an application programming interface for its applications, and would not care that the device is being plugged into and accessed by a VIO Server rather than, say, a RAID controller or a SAN. As long as the application communicates with the firmware using the correct programming interface, all will be well.

In the same way, from the perspective of the operating system, the “application” running on it is the Java Virtual Machine (JVM™). It is immaterial to the operating system that the JVM is running a large and complex Java application (WebSphere Application Server), or that there is Java code running inside JEE containers inside the Java application.

From the perspective of the Java Virtual Machine, its “application” is the WebSphere Application Server. From the perspective of the WebSphere Application Server, its “application” is the servlet or other JEE constructs deployed inside it. And so on it goes. As you can see, the definition of “application” depends on the perspective.

1.1.2 A holistic system approach

As mentioned, the definition of “application” depends on the perspective. Conversely, however, all of the separate component parts and all of the separate “applications” are interdependent. Thus, to optimize a WebSphere application it is not enough to simply tune the JVM or operating system, or modify the Java code. Instead, you need to develop a holistic view and examine the entire application stack to determine where the interdependencies lie. No single part should be tuned in isolation without an understanding of how it affects the rest of your system.

Similarly, when troubleshooting a WebSphere application, look at the whole application stack, because fixing a problem at one level simply fixes a part of the problem. Customers often blame a single piece of the stack for problems when the issue may lie somewhere else entirely.

1.2 System layers and points of view

To successfully run an application on WebSphere Application Server running on System p, you need to consider all of the perspectives just described. Typically, your enterprise will have teams of specialists that bring a combination of skills in the following areas: AIX operating systems administration, System p hardware configuration and tuning, and Java and WebSphere application engineering.

Specialists focusing on each of these skill sets work from different perspectives within the whole system view, as shown in Table 1-2. As a result, the potential exists for confusion over terminology and overlapping functionality when you try to combine these viewpoints into a single, overall system optimization strategy.

The specialists tend to work at a single level; for example, a UNIX systems administrator will tend to see things from the operating system point of view, and a Java developer will tend to see things from the application programmer point of view. Table 1-2 lists system features and points of interaction at which the WebSphere Application Server and system administration viewpoints might integrate.

Table 1-2 System viewpoints and integration

	AIX	Java
Dynamic Provisioning	Operating system images, NIM Server, Automatic Builds	Silent Install Scripts Installation Factory Restore from backup

	AIX	Java
Dynamic Resource Allocation	Micropartitions Entitlements (Service Level Agreements), Workload Management Priority/Weighting	Configuring new servers into a Network Deployment cell, WebSphere eXtended Deployment
Tuning Format	Operating system Tuning Bring up Runtime (part of load testing, and an ongoing activity)	JVM Tuning Operating system-specific WebSphere Application Server parameters?

1.2.1 Points of view and terminology

Table 1-3 lists terms that are similar but have different uses, as presented from the “systems” point of view and from the “WebSphere” point of view.

Table 1-3 Points of view and system terminology

Term	Systems meaning	WebSphere meaning
cluster	In AIX and POWER5™ terms, <i>cluster</i> is a term for a group of machines, which do not necessarily have to run the same applications.	In WebSphere terms, <i>cluster</i> refers to a group of application server clones that all run exactly the same Java application and share the load.
wlm	In the AIX world, WLM refers to Workload Manager, the AIX workload management product that allows for dynamic provisioning of resources (CPU, memory and I/O) to processes.	In the WebSphere world, <i>wlm</i> refers to WebSphere’s Workload Management, which distributes load between the members of a cluster.
xd	<i>XD</i> refers to HACMP™ Extended Distance.	<i>XD</i> refers to WebSphere eXtended Deployment.
node	A <i>node</i> is a generic term referring to a machine or a logical partition.	A <i>node</i> refers to a grouping of application server processes controlled by a node agent, usually deployed as one node per machine or logical partition.

Term	Systems meaning	WebSphere meaning
server	A <i>server</i> is a generic term referring to a physical machine or sometimes a logical partition.	A <i>server</i> usually refers to an Application Server instance, which is a specific configuration of a Java Virtual Machine on which customer Java code will run.
dmgr/ drmgr	<i>drmgr</i> is the dynamic resource manager configuration command, which is an AIX command to install and configure dynamic logical partitions (dlpar scripts).	<i>dmgr</i> is used as an abbreviation of Deployment Manager, which is a special Java process that controls the configuration of resources within a given WebSphere cell.

1.3 The remainder of this book

The remaining chapters cover the following topics:

- ▶ Chapter 2, “WebSphere on System p and AIX 5 strategies” on page 7, explores scalability, availability, session management, and maintainability.
- ▶ Chapter 3, “System p platform configuration” on page 37, provides detailed information about how to configure and manage System p hardware and logical partitions.
- ▶ Chapter 4, “AIX configuration” on page 127, presents methodologies for AIX system administration and configuration management that can be useful when you set up a standardized operating system image for WebSphere deployment.
- ▶ Chapter 5, “WebSphere Application Server on AIX: under the hood” on page 141, examines the structure and functionality of Java Virtual Machine (JVM) and WebSphere Application Server and explains how they interact with the underlying AIX platform.
- ▶ Chapter 6, “Tuning the IBM Java Virtual Machine” on page 303, describes the use of tuning parameters that are available on the IBM Java Virtual Machine (JVM) on AIX.
- ▶ Chapter 7, “High availability, clustering and WebSphere” on page 335, describes methods for implementing high availability and related concepts by using various WebSphere and AIX capabilities together.
- ▶ Chapter 8, “Clustering WebSphere Application Server for performance” on page 361, explains methods for implementing highly scalable and performant systems using various WebSphere and AIX capabilities.

- ▶ Chapter 9, “AIX 5L and WebSphere XD” on page 377, discusses how WebSphere Extended Deployment (XD) can be optimized in an AIX 5L environment.
- ▶ Chapter 10, “Implementation scenario” on page 385, presents the details of our lab installation and the configuration methods we used in the development of this book.

WebSphere on System p and AIX 5 strategies

High availability and scalability are the two key infrastructure adaptability requirements required by almost all organizations in their e-business solutions. These requirements can be met by utilizing the advanced features and capabilities of IBM WebSphere Application Server Network Deployment V6 (such as Workload Management and Clustering), IBM POWER hardware (with advanced features such as micro-partitioning, DLPARs, partition mobility) and the AIX Operating System.

Although a variety of factors are involved when you are considering the appropriate topology for a WebSphere deployment, the primary considerations are described in this chapter. The topics of scalability, availability, session management, and maintainability are examined.

For detailed information about topologies, including a discussion of advantages and disadvantages, required software, and topology selection criteria, refer to *WebSphere Application Server V6 Planning and Design WebSphere Handbook Series*, SG24-6446.

Note: The considerations discussed in this document can also serve as a basis for other distributed platforms. However, our goal here is to utilize the advanced virtualization features and capabilities offered by POWER5.

2.1 Scalability considerations

On demand computing requires the ability to scale an application up or down, depending on current requirements. Thus, scalability is important for improving efficiency and reducing cost. This section discusses scalability strategies using WebSphere Application Server that can help you to ensure high availability, load balancing, and the removal of bottlenecks.

The basic infrastructure components that comprise a WebSphere application are:

- ▶ Web server and Web server plug-in
- ▶ Web container
- ▶ EJB™ container
- ▶ Databases

WebSphere Application Server implements Web containers and EJB containers in each application server. Each application server runs in its own Java Virtual Machine (JVM). If all components are co-located on a single LPAR, they might compete for LPAR resources, influence each other's behavior, or have unauthorized access to strategic resources.

To avoid these issues, one strategy is to cluster a set of application servers that are managed together and participate in workload management. Application servers participating in a cluster can be deployed on the same node or on different nodes.

Other scaling strategies include:

- ▶ Using a faster machine
- ▶ Creating a cluster of LPARs
- ▶ Using appliance servers
- ▶ Segmenting the workload
- ▶ Batch requests
- ▶ Aggregating
- ▶ Managing connections
- ▶ Cachings

For detailed explanations of these strategies, you can refer to Chapter 2 of “WebSphere Application Server V6 Performance and Scalability Handbook”, SG24-6392.

2.1.1 Clustering

A *cluster* is a set of application servers that are managed together and participate in workload management. Application servers participating in a

cluster can be on the same node, or on different nodes. A Network Deployment cell can contain no clusters or it can have many clusters, depending on the need of the administration of the cell. The cluster is a logical representation of the application servers. It is not necessarily associated with any node, and does not correspond to any real server process running on any node. A cluster contains only application servers, as well as the weighted workload capacity associated with those servers.

Why to use clustering

The use of clustering addresses two major problems, namely high load performance degradation and downtime (the system hosting application server). Scalability, through load balancing, remedies high load performance degradation. High availability, through failover, remedies downtime.

Queuing and clustering considerations

Cloning applications servers to create a cluster can be a valuable asset in configuring highly scalable production environments, especially when the application is experiencing bottlenecks that are preventing full CPU utilization of symmetric multiprocessing (SMP) servers.

When adjusting the WebSphere Application Server system queues in clustered configurations, remember that when a server is added to a cluster, the server downstream receives twice the load as shown in Figure 2-1.

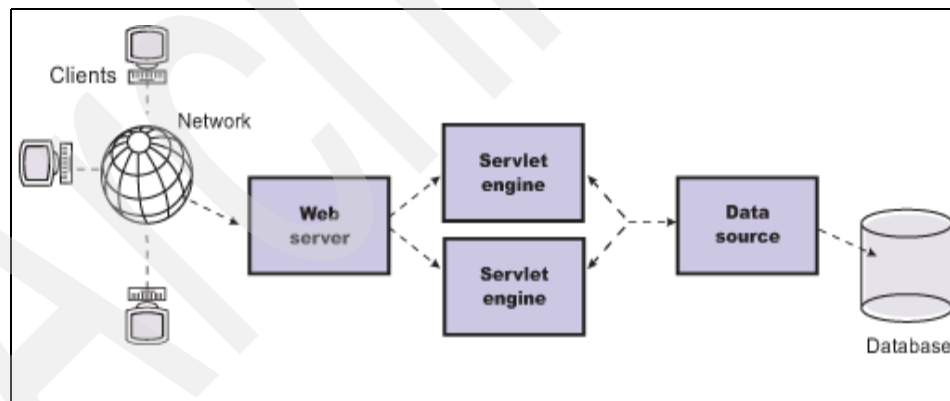


Figure 2-1 Clustering and queuing

The servlet engine is the key processing element within a Web container for supporting servlets and allowing them to respond to requests.

Two servlet engines are located between a Web server and a data source. It is assumed that the Web server, servlet engines, and data source, but not the database, are all running on a single SMP server. Given these constraints, the following queue considerations must be made:

- ▶ Double the Web server queue settings to ensure that ample work is distributed to each Web container.
- ▶ Reduce the Web container thread pools to avoid saturating a system resource like CPU or another resource that the servlets are using.
- ▶ Reduce the data source connection pool size to avoid saturating the database server.
- ▶ Reduce Java heap parameters for each instance of the application server. For versions of the Java virtual machine (JVM) shipped with WebSphere Application Server, it is crucial that the heap from all JVMs remain in physical memory. For example, if a cluster of four JVMs is running on a system, enough physical memory must be available for all four heaps.

Important: When creating a cluster, it is possible to select an existing application server to use as a template for the cluster without adding that application server into the new cluster (the chosen application server is used only as a template, and is not affected in any way by the cluster creation). All other cluster members are then created based on the configuration of the first cluster member.

Cluster members can be added to a cluster in various ways, during cluster creation and afterwards. During cluster creation, one existing application server can be added to the cluster or one or more new application servers can be created and added to the cluster. There is also the possibility of adding additional members to an existing cluster later on. Depending on the capacity of your systems, you can define different weights for the various cluster members.

Cluster members are required to have identical application components, but they can be sized differently in terms of weight, heap size, and other environmental factors. This concept allows clusters to span across a heterogeneous environment, including multiple LPARs. (Do *not*, however, change anything that might result in different application behavior on each cluster member.)

Starting or stopping the cluster starts or stops all cluster members automatically, and changes to the application are propagated to all application servers in the cluster.

Figure 2-2 on page 11 shows an example of a possible configuration that includes server clusters. Server Cluster 1 has two cluster members on node B only. Server Cluster 2, which is completely independent of Server Cluster 1, has

two cluster members on node A and three cluster members on node B. Finally, node A also contains a free-standing application server that is not a member of any cluster.

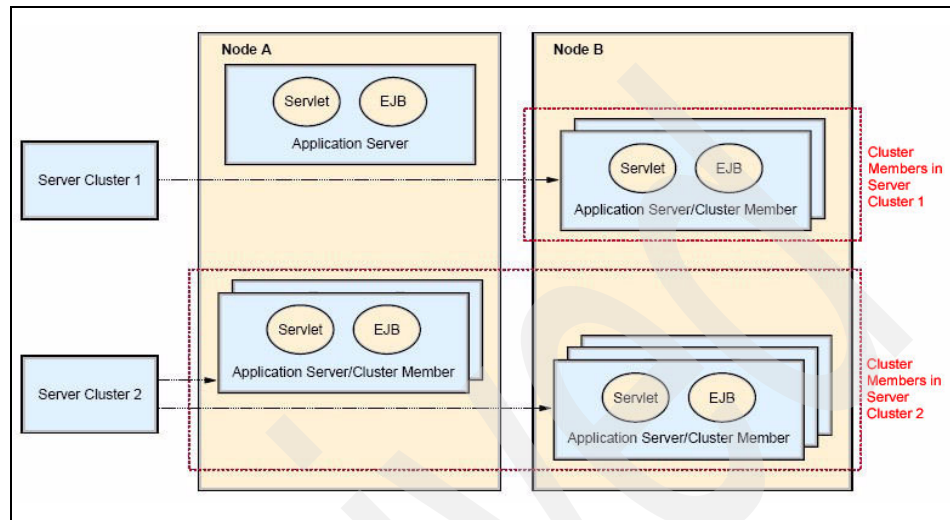


Figure 2-2 Server clusters and cluster members

For customers who plan to have high workload environments, it is important to plan how the cluster will operate both under normal conditions and under failure conditions.

Recommendation: The highly available manager (HAManager) monitors many cluster-wide resources. In general, this takes a certain amount of performance. If the cluster members are paging or otherwise engaged so that HAManager functionality cannot operate effectively, then HA-managed events will begin to occur to account for perceived cluster member anomalies.

For this reason, we recommend that you do not put the application servers under a large load in normal cases, in order to better handle spikes at times when challenges arise. In addition, reducing virtual memory paging as much as possible will result in a more reliable cluster operational environment.

When performing capacity planning for an LPAR, give consideration to the memory and CPU demands of the application. For example, applications that are CPU-bound should have the LPAR provisioned for ample processor units in order for it to meet its peak demand.

Clustering for scalability and failover

Clustering is an effective way to perform vertical and horizontal scaling of application servers.

Scalability pertains to the capability of a system to adapt readily to a greater or lesser intensity of use, volume, or demand. For example, a scalable system can efficiently adapt to work with larger or smaller networks performing tasks of varying complexity. Ideally, it is possible to handle any given load by adding more servers and machines, assuming each additional machine processes its fair share of client requests. Each machine should process a share of the total system load that is proportional to the processing power of the machine.

Using cluster members can improve the performance of a server, simplify its administration, and enable the use of workload management.

Vertical scaling (scale up topology)

In vertical scaling, as illustrated in Figure 2-3, multiple cluster members for an application server are defined on the same LPAR, or node, which might allow the LPAR's processing power to be more efficiently allocated.

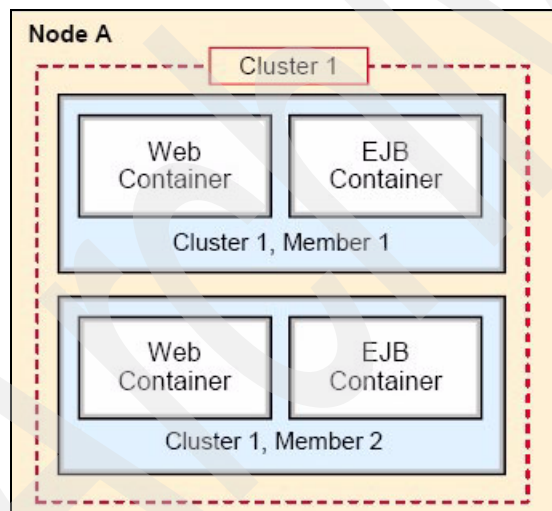


Figure 2-3 Vertical scaling

Even if a single JVM can fully utilize the processing power of the machine, you might still want to have more than one cluster member on the machine for other reasons, such as using vertical clustering for process availability. If a JVM reaches a memory limit (or if there is some similar problem), then the presence of another process provides for failover.

If you install several application server instances on a single LPAR (assuming the LPAR has sufficient resources, such as CPU) to create a vertical cluster, then application throughput increases.

Vertical clusters are valuable for better utilization of the LPAR when the operating system otherwise constrains the availability of resources on a process boundary. For example, if a JVM process is pinned to a single processor on a symmetric multiprocessor (SMP) computer, introducing additional application server instances allows the process to utilize other CPUs on the same computer (presuming they would be assigned to the other processors). Or, if the operating system limits the number of connections that can be formed to a single process, then an increase in the number of effective connections to the computer can be made by increasing the number of application server instances.

Although vertical scaling can improve availability by creating multiple JVM processes, the LPAR itself remains a single point of failure. Therefore, the use of vertical clustering should not be viewed as a means of achieving high availability.

Before implementing a vertical cluster, determine if your applications are bound by CPU, by I/O, or by network issues. Avoid using rules of thumb when determining the number of cluster members for a given machine. The only way to determine what is correct for your environment and application is to tune a single instance of an application server for throughput and performance, and then add it to a cluster and incrementally add additional cluster members. Ensure that you test performance and throughput as each member is added to the cluster. Always monitor memory usage when you are configuring a vertical scaling topology, and do not exceed the available physical memory on a machine.

In general, 85% (or more) utilization of the CPU on a large server shows that there is little, if any, performance benefit to be realized from adding additional cluster members.

Note: You also have the flexibility of removing a resource from an LPAR if the application does not utilize it. The resource can be dynamically moved to another LPAR where required.

Horizontal scaling (scale out topology)

In horizontal scaling, as illustrated in Figure 2-4 on page 14, cluster members are created on multiple physical machines (or LPARs). This allows a single WebSphere application to run on several machines, 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 small- to medium-sized LPARs; client requests that overwhelm a single LPAR can be distributed over several LPARs in the system.

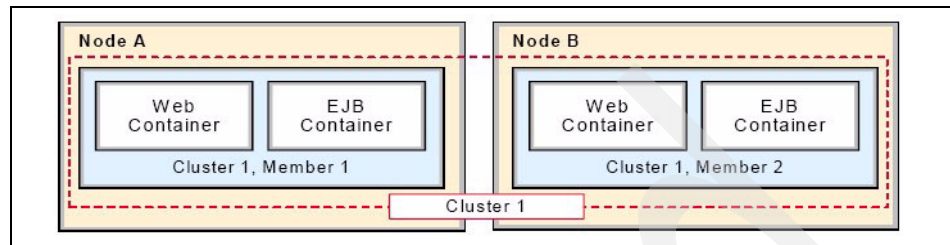


Figure 2-4 Horizontal scaling

Failover is another important benefit of horizontal scaling. If a machine becomes unavailable, its workload can be routed to other machines containing cluster members.

Horizontal scaling can handle application server process failures and hardware failures (or maintenance) without significant interruption to client service. It is common to use similar machines to host members from a cluster. This allows you to easily plan for future capacity need in a linear fashion.

You can also use WebSphere Application Server Edge components, such as the Caching Proxy Edge component and the Load Balancer component set (which includes the Dispatcher component) to implement horizontal scaling.

Combining vertical and horizontal scaling

WebSphere applications can combine vertical and horizontal scaling to reap the benefits of both scaling techniques, as illustrated in Figure 2-5.

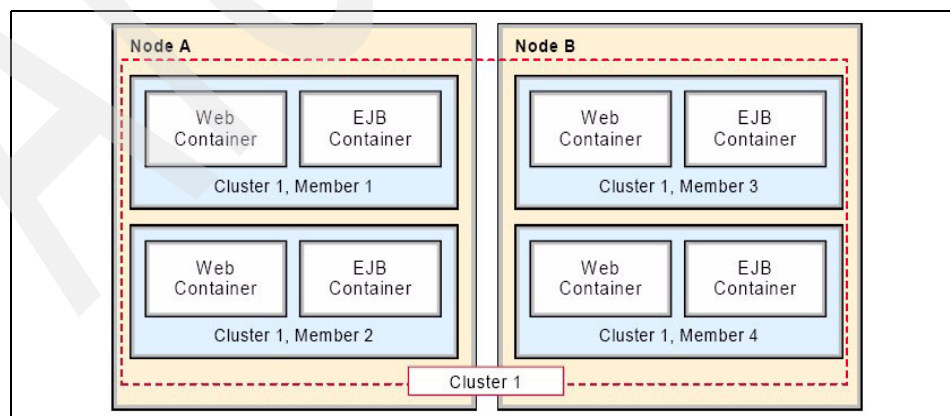


Figure 2-5 Vertical and horizontal clustering

As a rule of thumb for real world applications, first use vertical scaling to improve performance, while carefully monitoring server-to-CPU and server-to-memory ratios. After performance is optimized, start using horizontal scaling to provide failover and redundancy support to maintain 24x7 uptime with the desired performance.

2.1.2 Workload management

Workload management is implemented in WebSphere Application Server Network Deployment V6 by using application server clusters and cluster members. These cluster members can all reside on a single node (LPAR), or be distributed across multiple nodes (LPARs).

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. Workload management is the WebSphere facility to provide load balancing and affinity between application servers in a WebSphere clustered environment. It optimizes the distribution of processing tasks in the WebSphere Application Server environment; incoming work requests are distributed to the application servers that can most effectively process the requests.

Workload management is also a procedure for improving performance, scalability, and reliability of an application. It provides failover when servers are not available. WebSphere uses workload management to send requests to alternate members of the cluster. WebSphere also routes concurrent requests from a user to the application server that serviced the first request, as EJB calls, and session state will be in memory of this application server.

Workload management is most effective when the deployment topology is comprised of application servers on multiple LPARs, because such a topology provides both failover and improved scalability. It can also be used to improve scalability in topologies where a system is comprised of multiple servers on a single, high capacity machine. In either case, it enables the system to make the most effective use of the available computing resources.

Two types of requests, HTTP requests and EJB requests, can be workload managed in IBM WebSphere Application Server Network Deployment V6, as explained here:

- HTTP requests can be distributed across multiple Web containers.

When an HTTP request reaches the Web server, a decision must be made. Some requests for static content might be handled by the Web server. Requests for dynamic content or some static content will be passed to a Web container running in an application server. Whether the request should be

handled or passed to WebSphere is decided by the WebSphere Web server plug-in, which runs in-process with the Web server. We refer to this as WLM Plug-in. For these WebSphere requests, high availability for the Web container becomes an important piece of the failover solution.

- EJB requests can be distributed across multiple EJB containers.

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. We refer to this as EJS WLM.

An application deployed to a cluster runs on all cluster members concurrently. The workload is distributed based on weights that are assigned to each cluster member. Thus, more powerful LPARs receive more requests than smaller systems. When deciding on a topology, it is important to assign the appropriate weighting for each individual LPAR if they are different. Alternatively, it is common to use similar LPARs with the similar capacity.

Workload management also takes care of failing over existing client requests to other, still available application servers, and of directing new requests only to available processes if an application server in the cluster should fail. In addition, workload management enables servers to be transparently maintained and upgraded while applications remain available for users. You can add additional cluster members to a cluster at any point, providing scalability and performance if an existing environment is not able to handle the workload any more.

Distributing workloads

The ability to route a request to any server in a group of clustered application servers allows servers to share work and improve throughput of client requests. Requests can be evenly distributed to servers to prevent workload imbalances in which one or more servers has idle or low activity while others are overburdened. This load balancing activity is a benefit of workload management.

Thus, the proposed configuration should ensure that each LPAR or server in the configuration processes a fair share of the overall client load that is being processed by the system as a whole. In other words, it is not efficient to have one LPAR overloaded while another LPAR is mostly idle. If all LPARs have roughly the same capacity (for example, CPU power), then each LPAR should process a roughly equal share of the load. Otherwise, there likely needs to be a provision for workload to be distributed in proportion to the processing power available on each LPAR.

Using weighted definitions of cluster members allows nodes (or LPARs) to have different hardware resources and still participate in a cluster. The weight specifies that the application server with a higher weight will be more likely to

serve the request faster, and workload management will consequently send more requests to that node.

With several cluster members available to handle requests, it is more likely that failures will not negatively affect throughput and reliability. With cluster members distributed to various nodes, an entire machine can fail without any application downtime. Requests can be routed to other nodes if one node fails. Clustering also allows for maintenance of nodes without stopping application functionality.

To read detailed descriptions of workload management policies, and learn how requests are distributed among available servers, refer to Chapter 6 and to Chapter 7 of *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392.

2.2 Session persistence considerations

Unless you have only a single application server or your application is completely stateless, maintaining state between HTTP client requests is also a factor in determining your topology. Use of session information, however, presents a fine line between convenience for the developer and performance and scalability of the system. It is impractical to eliminate session data altogether, but try to minimize the amount of session data passed. Persistence mechanisms decrease the capacity of the overall system, or incur additional costs to increase the capacity or even the number of servers. Therefore, when designing your WebSphere environment, take session needs into account as early as possible.

If the application maintains state between HTTP requests and you use vertical or horizontal scaling, then you must consider using an appropriate strategy for session management. Each application server runs in its own JVM process. To allow a failover from one application server to another without losing state, you need to share the session data between multiple processes.

There are two methods of achieving this in WebSphere Application Server Network Deployment:

- ▶ Memory-to-memory session replication

This method employs Data Replication Service (DRS) to provide replication of session data between the process memory of different application server JVMs. DRS is included with WebSphere Application Server and is automatically started when the JVM of a clustered member starts.

It also does not require a separate database to store the persistent states, and eliminates a single point of failure found in the session database if the database itself has not been made highly available using clustering software.

► Database persistence

With this method, session data is stored in a database shared by all application servers. It requires external clustering software to make it highly available.

The default maximum pool size of 10 is a useful starting point when you size the connection pool for database persistence. You may want to increase it if your session object is quite large, or your site handles thousands of concurrent users.

Regarding large session objects: You can achieve faster performance by increasing page size, which will allow session object to fit into a single database page. In IBM DB2®, for example, page size can be adjusted from the default of 4 KB to 8 KB, 16 KB, and 32 KB.

Using the multi-row schema option can also improve performance.

For larger session sizes, the overhead of session replication increases. Database replication has a lower overall throughput than memory-to-memory due to database I/O limitations (the database becomes a bottleneck). However, although like database replication with large sessions performs slower, less CPU power is used than with memory-to-memory replication, and the unused processor power can be used by other tasks on the system.

Also, storing session states in a persistent database or using memory-to-memory replication provides a degree of fault tolerance to the system. If an application server crashes or stops, any session state that it may have been working on would normally still be available either in the back-end database or in the running application server's memory, so that other application servers can take over and continue processing subsequent client requests associated with that session.

Neither mechanism provides a 100% guarantee that a session state will be preserved in case of a server crash. However, such a situation represents only a very small window of vulnerability, and a very small percentage of all occurrences that typically happen throughout the life of a system in production.

The overhead of replicating session information might be significant, depending on the number of application servers in your cluster and the size of the sessions. In these situations, database persistence might be the better choice in a memory-constrained environment.

The question becomes, how to choose one option over the other. For configurations where cross-cell session failover is a requirement, then database persistence might be the better choice. In a cascading failure, the database may survive, although memory-to-memory replicators may not. Similarly, with

memory-to-memory, there can only be a single replicator common to the cells, which leads to a single point of failure (SPOF).

Keep in mind that most of the cost of replication is incurred in the serialization and deserialization of the session object, and this applies to both cases. Also in both cases, as a session object's size increases, performance decreases.

Ultimately, your decision will depend upon your application availability requirements, the quality of service needed for your applications, and your comfort zone for the technology, whether database or memory-to-memory.

2.3 File store considerations

The WebSphere Application Server transaction manager is responsible for storing information regarding the state of completing transactions in a persistent form that is used during transaction recovery. This persistent form is referred to as the *transaction recovery log* and is used to complete prepared transactions following a server failure. This activity is referred to as *transaction recovery processing*.

In addition to completing outstanding transactions, this processing also ensures that locks held in the associated resource managers are released. Prior to WebSphere Application Server V6, it was necessary to restart a failed server to perform recovery processing. Version 6.0 introduced the ability for a peer server (another cluster member) to process the recovery logs of a failed server while the peer continues to manage its own transactional workload.

This capability is known as *peer recovery processing*, and it supports the resolution of in-doubt transactions without the need to wait for the failed server to restart. This facility forms part of the overall WebSphere Application Server high availability (HA) strategy.

The basic requirement for transaction recovery requires the transaction recovery log to persist on a highly available file store. The file store is basically the file system it is placed in. The recommendation is to use hardware-based or software-based facilities to maximize the availability of the file systems themselves, such as the use of Storage Area Network (SAN). WebSphere Application Server V6.1 supports either cluster-managed or networked file systems.

Cluster-managed file systems use clustering and failover of shared disks to ensure high availability of files and directories.

Networked file systems use remote servers to store and access files as though they were local servers. Make sure that the file system in use supports access locking to ensure integrity of the file store components, particularly the log file, by the use of exclusive locks. We recommend that you use NFS V4.

For more information, refer to *Transactional high availability and deployment considerations in WebSphere Application Server V6*, which is available at the following address:

http://www-128.ibm.com/developerworks/websphere/techjournal/0504_beaven/0504_beaven.html

2.4 Install automation considerations

Installing and configuring a WebSphere software product is usually a multiple-step process:

1. Install the shipped version of the product.
2. Install the current Refresh Pack.
3. Install the current Fix Pack.
4. Install a Java 2 Software Development Kit (SDK) Fix Pack.
5. Install one or more interim fixes as needed.
6. Create and configure application servers and other artifacts.
7. Deploy applications.

WebSphere Application Server is normally installed using the installation wizard graphical user interface (GUI). But enterprises often automate the installation process for repeatable installation and rapid deployment. WebSphere Application Server installation can be automated using two different supported options, namely silent installation or Installation Factory, as explained in the following sections.

2.4.1 Silent installation

A *silent installation* uses the installation wizard to install the product in silent mode, without the graphical user interface. Instead of displaying a wizard interface, the silent installation causes the installation program to read all of your responses from a file that you provide.

The installation options must be defined in the response file before you issue the installation command. Note that silent installation mode does *not* accept interactive installation options. Refer to “WebSphere: responsefile.nd.txt” on page 424 in Appendix A, “Sample files” on page 423 for a sample response file.

Knowing what component to install and in what order to install is an important consideration. You must consider common installation scenarios for Network Deployment in order to determine how to install your application serving environment.

Installation of WebSphere Application Server Network Deployment typically involves two tasks. First, the installation wizard installs a shared set of core product files. Next, the installation wizard optionally creates a profile. A *profile* is a separate data partition that includes the files that define a runtime environment for an application server process, such as a deployment manager or an application server.

The Installation wizard has the capability to perform a new product installation, an incremental installation by adding features to an existing installation, or an update to an existing installation that updates the installation to a new service level.

2.4.2 Installation Factory

The Installation Factory creates turn-key install packages for installing WebSphere Application Server in a reliable and repeatable way, tailored to your specific needs. Installing and configuring WebSphere Application Server requires just one step: simply install the customized installation package (CIP) created by the Installation Factory. The other steps are performed automatically by the CIP you created. The Installation Factory combines the installation image for a version or release of a WebSphere software product with optional assets to create a CIP.

Optional assets can include any of the following components:

1. Applicable maintenance packages.
2. Scripts or Java classes to be run during install and uninstall, or profile creation and deletion.
3. Enterprise application archive (EAR) files for applications that you intend to deploy with default deployment options on standalone application server profiles.
4. A configuration archive (CAR) file for cloning a standalone application server profile from a previously installed and customized standalone application server.
5. Additional files, such as EAR files that you intend to deploy with customized options using a script that you have written.

The only required asset is the WebSphere Application Server installation image, which is either on the product disk or in a downloaded image.

Installation Factory consists of both a graphical user interface (GUI) tool (the `ifgui` command in the bin directory), and a command line interface tool (the `ifcli` command).

Before creating a CIP, you must first create a build definition for the CIP. The build definition is an XML document that defines how the Installation Factory is to customize the WebSphere Application Server product.

Use the Installation Factory GUI to create a build definition file, which is an XML document that specifies how to build the CIP (for example, where to find the Refresh Pack you plan to use). You can generate the CIP directly from within the GUI, or you can simply choose to save the build definition file and then generate the CIP outside of the GUI using the provided command line interface tool.

The CIP you create will contain an installation program that can be used to install the CIP either interactively, using the Installation wizard, or silently. Further, the CIP can perform a new scratch install of the application server, or it can be applied against an existing installation of the application server. In either case, the resulting installation is at the maintenance level you require; is configured as required; and any applications you have provided will be deployed. This is the recommended procedure for automating a WebSphere installation.

2.5 JVM tuning

There are two main areas to consider when tuning Java on AIX: memory management and CPU performance. This section briefly discusses each area.

Note: For more detailed information about JVM tuning, refer to Chapter 6, “Tuning the IBM Java Virtual Machine” on page 303.

2.5.1 Memory management

The most likely candidate for optimizing Java runtime performance is tuning memory management. Memory management in this context is considered to be the allocation and deallocation of objects within the heap. Because memory management is performed at runtime, rather than in the application code, it is a key differentiator among vendor JVMs. IBM has invested much research into heap management and garbage collection within the JVM. The IBM JVM offers multiple garbage collection algorithms, or policies, with each tailored for a specific purpose. These policies may be further tuned to optimize performance.

If an application requires a large amount of memory, AIX offers the opportunity to use increased page sizes. Using larger page sizes will help performance by reducing the overhead that is associated with paging.

If multiple instances of the JVM will be run on the same machine or partition, a reduced memory footprint and decreased startup time can be achieved through the use of a shared class cache.

2.5.2 CPU performance

CPU performance is defined as the efficient utilization of processor resources. Much of the improvement in optimizing processor utilization for Java has resulted from the creation of runtime compile systems. The inclusion of just-in-time (JIT) or “hotcode” compilers has significantly contributed to improving Java’s runtime performance. IBM’s JIT is included in all IBM JVMs, and is enabled by default. The JIT is tuned for long-running applications, but opportunities still exist for tuning the JIT for a particular application.

A distinguishing feature of the IBM JVM on AIX is built-in micropartitioning and dynamic logical partitioning (DLPAR) support. If the JVM will be running on an LPAR that is micropartitioned and resources will be shared with other partitions, take care to ensure that enough memory will always be available to the JVM.

In addition to these considerations, additional AIX operating system parameters are available to further optimize Java runtime performance. These tunable parameters are described in Chapter 4, “AIX configuration” on page 127.

Running WebSphere Application Server with the default Java runtime parameters will most likely *not* provide the best results. The defaults are configured to be satisfactory for all types of applications. But every application is unique, therefore, determining which tuning parameters will help a specific application is useful. The guidelines presented in Chapter 6, “Tuning the IBM Java Virtual Machine” on page 303, help you find the tuning parameters that take advantage of AIX capabilities and features and provide optimal results.

Note: IBM WebSphere Application Server 6 and WebSphere Application Server 6.1 use a Java 5 implementation of the Java runtime. Java 5 uses MXBeans for providing new dynamic monitoring functions as part of the `javax.management` package.

A description of one approach you can take to use this API for your runtime monitoring is provided in the developerWorks® article available at the following address:

<http://www.ibm.com/developerworks/java/library/j-mxbeans/>

2.6 Extended Deployment (XD) considerations

WebSphere XD provides an IT infrastructure that dynamically and reliably adapts to changing business demands. By extending the capabilities of WebSphere Application Server Network Deployment, WebSphere XD can help you optimize the utilization and management of your deployments and enhance the quality of service of your business-critical applications.

2.6.1 Dynamic operations: WebSphere Virtual Enterprise

WebSphere Virtual Enterprise is a feature of WebSphere XD that provides a set of application infrastructure virtualization capabilities. It is designed to deliver dynamic operations through two key capabilities:

- ▶ Virtualization of WebSphere environments
- ▶ Introduction of a goals-directed infrastructure

A virtualized WebSphere environment allows you to expand your solution as business needs dictate through the dynamic allocation of WebSphere resources.

WebSphere XD implements a virtualized environment by creating pools of resources that can be shared among applications, thereby optimizing utilization and simplifying overall deployment. As resources are needed for expected and unexpected spikes in workload demand, application resources can be allocated to where they are needed most. Allocating resources on an on-demand basis enables better use of computing resources that you already own and, potentially, might allow you to run more applications on the LPARs that you already have in place.

Planning a production environment for dynamic operations is different than planning one for a static environment. In a static environment, you have dedicated servers for each application. To size the servers, take a look at your applications, the requirements, and the expected load during peak time. Your production environment must be prepared for the load during this (possibly, short) period. This means that, during non-peak hours, your servers are underutilized.

In general, a company has more than one critical application. It is likely that the second application has its peak load at a different time of the day. In a static environment, the servers hosting the first application cannot be used for the peak load of the second application. Therefore, the quality of service (QoS) might suffer or you need more or bigger systems to ensure QoS.

To guarantee a good QoS, one possibility would be to add additional nodes. However, doing so would increase the cost and would not solve the underlying

problem. In this case, the problem is that you cannot share your resources and thus optimize utilization.

The following list of questions help you gather the information required to set up an environment for dynamic operations:

- ▶ What applications do you want to include?
This affects the number of node groups and dynamic clusters.
- ▶ What are your critical applications?
This affects the allocation of applications to dynamic clusters and the assignment of service policies.
- ▶ Are there applications that should not run on the same LPAR?
These LPARs have to be in different node groups.
- ▶ Which applications can share the same server configuration?
These can be in the same dynamic cluster.
- ▶ Which servers and hardware do you want to include?
In a heterogeneous environment it could be interesting to have multiple node groups, dependent on the hardware type.
- ▶ Do all servers have the same resources available, such as network drivers, database connections, external drives, additional software?
Because every application can be started on every node in a node group, you need to have all resources available on each node. You must take this into consideration because it might increase license costs and possibly require additional hardware resources on each LPAR.

Based on this information, you should be able to plan your node groups, dynamic clusters, and service policies.

Sharing resources among applications helps to optimize utilization and simplify deployment. WebSphere XD redefines the relationship between traditional J2EE constructs. Instead of deploying applications directly onto an application server, you can map an application into a resource pool. This application can then be deployed on some subset of servers within that pool according to your configured business goals. The WebSphere XD virtualized infrastructure is predicated on two new constructs: node groups (which represent resource pools) and dynamic clusters, as explained in the following sections.

Node groups

In WebSphere Extended Deployment, the relationship between applications and the nodes on which they can be run is expressed as an intermediate construct called a *node group*. In concrete terms, a node group is nothing more than a set

of LPARs. In a more abstract sense, a node group is a pool of LPARs with a common set of capabilities and properties such as connectivity to a given network or the capability to connect to a certain type of database. These characteristics are not explicitly defined; node group attributes are purely implicit in the WebSphere XD design.

Within a node group, one or more dynamic clusters are created. The computing power represented by a node group is divided among its dynamic cluster members. This distribution of resources is modified autonomically according to business goals to compensate for changing workload patterns.

Because a node group's set of common capabilities and properties is required by some suite of applications, a node group defines the collection of machines able to run a given application. Because the administrator now understands what is implied by participation in a given node group, the administrator can ensure that applications are deployed into node groups where they can be accommodated. The resources in a given node group are dynamically allocated according to load and policy to deliver better resource utilization, leading to cost savings. Implementing virtualization using node groups breaks the tie between application clusters and machines, and enables them to be shared among applications, thus optimizing resource utilization and simplifying overall deployment.

Before defining node groups, however, you need to determine what kind of systems you want to include in the environment. That is, are all systems identical in terms of resources? Does an application need to be deployed on a specific set of systems because of its prerequisites? All applications can run on any node in the node group. In general, having more dynamic clusters and thus applications in a node group allows for better system utilization than having multiple node groups and fewer applications mapped to each node group.

In any case, the utilization depends on workload intensity. So you may achieve good utilization in a large node group that hosts one application with high workload, or in a small node group with a big number of applications that each receive only a small workload.

Note: Best practice is that all your nodes belong to one big node group. System utilization is optimized when the application placement controller has all nodes in just one node group.

However, this approach is resource-intensive because all applications are deployed on each node and all licences have to be purchased for the whole node group. So, depending on your environment, it might be better to balance the number of applications and the number of nodes that are part of a node group.

Dynamic clusters

The process of deploying an application in WebSphere XD begins with choosing or defining a node group that satisfies the application's requirements, and continues with the creation of a *dynamic cluster*. A dynamic cluster is a container construct that extends its static counterpart in WebSphere Network Deployment, the static cluster.

Dynamic clusters are associated with a single node group. Cluster applications (.ear files) are then deployed into a dynamic cluster in much the same fashion as they are deployed into WebSphere Network Deployment static clusters. However, WebSphere XD supports autonomic expansion and contraction of a dynamic cluster within its parent node group. Thus, periodic spikes in demand for an application result in a corresponding increase in that application's resource for processing requests. The strategy for increasing these resources is dictated by operational policies that reflect business goals.

Lazy application start

The lazy application start feature in WebSphere XD optimizes server resource allocation during times of inactivity. When a request for that application is received by the On Demand Router (ODR), the application is started automatically on any node in the node group.

A typical environment where the lazy application start feature is beneficial is an environment in which the ratio of the number of dynamic clusters to the number of servers is high, and where many dynamic clusters are not accessed for long periods of time. In such an environment, it is beneficial to hibernate idle dynamic clusters temporarily (stopping all server instances), thereby releasing valuable resources to be used by active dynamic clusters.

Note: Lazy application start can only be configured for the *whole* dynamic cluster.

Vertical stacking

WebSphere XD introduces a new feature called *vertical stacking*. This feature allows you to have more than one application server instance in a dynamic cluster on the same node. The benefit of this capability is better hardware utilization if the CPU and memory of an LPAR is not fully used with a single application server on a node.

To determine how many application server instances can run in parallel on a node, take a look at the resources needed when only one instance is active. We recommend that you determine the stacking number for a dynamic cluster with no other application servers running on that node.

First, start one instance. While monitoring the effective throughput, increase the workload intensity. When throughput saturates, start one more instance. When adding a new instance does not improve the throughput, the stacking number is 1. If adding a new instance improves the throughput, you can conclude that the application has some internal bottleneck that prevents it from effectively using the entire box within a single application server. Thus, you can continue to increase workload and (possibly) the number of active instances until no improvement in throughput can be achieved.

Repeat this approach individually for every application that can possibly run on that node. This way, you can decide for each cluster (and thus, application) whether it should be using stacking or not.

When determining the stacking number for a dynamic cluster, you do not have to consider any other dynamic clusters, because you only want to learn how many instances of *this* dynamic cluster are needed to fully utilize the system. So when you have multiple clusters or applications, each of them would be able to fully utilize the system if no other application is running concurrently (for example, because these applications only run at certain times of the day, week, or month).

The stacking number will become the *maximum* number of instances that are allowed to execute on a node for this dynamic cluster. However, at any time, a smaller number may be started, depending on the current workload. At run time, the application placement controller, ODR, and DWLM will work together to make sure that the node is not overloaded and that all applications meet their policies.

Alternatively, if vertical stacking is not necessary, then it is possible to remove resources from the LPAR.

Operational policy

With WebSphere XD, you can differentiate application service levels according to your business requirements. The goals-directed infrastructure capabilities of WebSphere XD mean that user requests are classified, prioritized, queued, and routed to servers based on application operational policies that are tied to business goals. Application performance is optimized according to these policies that reflect service level goals and relative importance to the organization.

Simply put, you can state what applications are important to you, and these applications will get the highest priority access to your WebSphere resources at the right time. This can help you ensure, for example, that your business-critical transactions get the best quality of service.

WebSphere XD offers two types of operational policies:

► Service policy

Service policies definitions are made up of two key items, namely goals and importance. The goal portion of the service policy defines how incoming work is evaluated and managed in order to ensure and detect if work is meeting its assigned service policy levels. The importance is used in times of resource contention in order to identify the most important work in the system, and give it the priority.

► Health policy

WebSphere XD provides a health monitoring and management subsystem. This subsystem continuously monitors the operation of servers to detect functional degradation that is related to user application malfunctions. The health management subsystem consists of two main elements, namely health policies and the health controller.

WebSphere XD supports the following health policies:

- Age-based condition: Monitors for servers that have been running longer than a configured time period.
- Excessive request time-out condition: This health policy will detect, for each server that is a member of the policy, the percentage of requests directed at that server which have timed out (over an interval of time) after being routed from the ODR.
- Excessive response time-out condition: Monitors for servers that appear to be hung due to requests taking an excessively long time period to complete.
- Workload condition: Monitors for servers that have executed more than a configured number of requests.
- Memory condition, excessive memory usage: Monitors for servers that appear to be consuming more memory than what the server has available.
- Memory condition, memory leak: Profiles the Java Virtual Machine (JVM) heap size after a garbage collection has occurred, and looks for trends of increased consumption. When an increasing trend is detected, the condition is triggered.
- Storm drain condition: Applies only to dynamic clusters and will detect, for each cluster member, a significant drop in the average response time for a member of the cluster coupled with changes to the dynamic workload manager (DWLM) weights for the cluster members.

2.6.2 Extended manageability

The extended manageability capabilities of WebSphere XD help to simplify IT management, while maintaining administrator control. It can be difficult to visualize and manage complex IT environments where tens of applications are deployed on hundreds of application servers. Although the WebSphere Application Server Administrative Console provides excellent built-in capabilities, the special needs of very complex deployments require an aggregated, meaningful view of the application runtime environment.

WebSphere XD extends the Administrative Console to allow operators to see at a glance what is happening in their infrastructures and the relative health of the components. It also enhances the existing WebSphere Application Server Administrative Console by charting application performance against business goals. Alerts® that notify you when intervention is required to deliver on business goals help decrease human-intensive monitoring and management.

View your infrastructure runtime status

To maintain ease of use in a product that lends itself to the management of complex deployments in an optionally fully automated fashion, WebSphere XD provides enhanced manageability features. These features include a visual console that provides a graphical representation of a dynamic WebSphere XD topology, reporting of performance statistics, and implementation of administrative operations. These visualization functions are delivered as an extension to the WebSphere Application Server Administrative Console.

The WebSphere XD Administrative Console contains several tools that help you visualize the inner workings of a WebSphere XD topology, so you can remain well-informed about the activities taking place within your environment. Operational views represent an intuitive, central distribution point of information pertaining to health, performance and, potentially, autonomic decisions.

Runtime topology

One such view is the runtime topology view, which is a depiction of the view, or the momentary state of WebSphere XD. This view refreshes on a configurable interval to provide updated information.

The runtime topology contains many useful bits of information, including:

- ▶ Application-provisioning activity
- ▶ Deployment of dynamic cluster instances
- ▶ Processor usage (per node)
- ▶ Node-to-node group memberships
- ▶ Dynamic cluster-to-node group memberships
- ▶ Dynamic workload management weight (per application server instance)

- Process identification (per application server instance)

Charting

WebSphere XD charting presents administrators with customizable graphs of runtime data observed throughout a WebSphere XD environment. This view refreshes on the same configurable interval as the runtime topology view. With WebSphere XD, you can chart a wide variety of statistics in six styles of graphs. Supported statistics include:

- Average response time
- Concurrent requests
- Average throughput
- Average queue wait time
- Average service time
- Average queue length
- Average drop rate

Charts can be constructed from several perspectives as well, which gives you flexibility in the scope of the statistics you observe. WebSphere Extended Deployment supports charting from cell, node group, dynamic cluster, service policy, transaction class, J2EE module, and proxy (On Demand Router) perspectives.

The WebSphere Extended Deployment charting facility also offers a brief historical log of statistical data. As new data points are added to the right side of a chart, old data is displayed until it scrolls off the left side of the chart. Of course, the length of the history visible on the chart at any point in time depends on the number of data points per sampling period, as well as the type of chart being viewed. In addition, there is a logging function available that can be used to store statistics for longer periods of time.

Runtime map

WebSphere XD provides an innovative visualization technique for displaying hierarchical data called a *treemap*. A treemap is simply a rectangle that is recursively subdivided into smaller rectangles, each of which represents the collection of nodes at some level in the tree of data being depicted. The significance of each rectangle's size and color is purpose-specific.

Runtime maps provide a robust search capability, which allows you to single out a subset of the data in an entire map (such as all application server instances in a single node), highlight the top 10 performing applications based on response time, or select the dynamic clusters that have the five lowest, concurrent request values.

Monitoring your environment

WebSphere XD extends the Administrative Console to notify you of decisions made by autonomic managers. Notifications can represent either planned or unplanned events.

Planned events

Planned events are expected events for which the WebSphere XD run time has an action plan. An example of a planned event would be an average response time breaching its configured limit, which might trigger an increased dynamic cluster footprint. Depending on the configured level of automation, these events could be shown to you in one of several ways. If WebSphere XD is operating in *on demand mode* (or automatic mode), the action plan runs and a simple notification is shown to you. In *supervisory mode*, you are presented with the mode, and the action plan, and prompted for approval. In *manual mode*, WebSphere XD presents a plan, and you can either follow or ignore the advice.

Over time, as you grow more familiar with WebSphere Extended Deployment and its behavior, such decisions and corresponding actions can happen automatically. With the three modes of operation provided by WebSphere XD, you can introduce autonomic capabilities into your IT infrastructure in a controlled and gradual way.

Unplanned events

Events that are not assigned action plans are displayed to you as warnings to let you know that something unexpected has happened. It is then up to you to develop a plan to correct the situation, if it is indeed problematic.

2.6.3 High performance computing: WebSphere eXtreme Scale

High performance computing is designed to support high volume transaction requirements reliably. To support ultra-high-end transaction processing requirements reliably within a unified WebSphere environment, WebSphere XD includes WebSphere eXtreme Scale. WebSphere eXtreme Scale includes a set of features that provide for dynamic application partitioning and repartitioning, high-end caching, workload management, and autonomic high availability management.

The WebSphere Partitioning Facility

The WebSphere Partitioning Facility (WPF) provides an essential capability required to achieve the next level in performance, scalability and availability in J2EE applications. As the name suggests, *partitioning* is the essential element in the partitioning facility. Although partitioning cannot help you improve

performance and availability on its own, it can establish the foundation upon which these benefits can be achieved.

WebSphere Partitioning Facility lets you create and manage partitions in WebSphere XD. A WebSphere Partitioning Facility partition can be described in several ways. In its simplest form, a partition is a list of labels that can be created by applications (or metadata found in declarations within applications) whenever they are required. For example, an application can render a set of abstract partition names to represent categories of items being bid upon during auctions: sporting goods, automobiles, toys, and antiques.

The WebSphere Partitioning Facility is a programming framework and runtime environment that makes it possible for high transaction volume applications to scale linearly by adding hardware capacity. To accomplish this, an application is partitioned across multiple servers in a cluster. Each partition is a uniquely addressable endpoint within the cluster, to which requests for certain EJBs or certain data are always routed. Partitioning solves some of the traditional challenges of very large clustering, because it can reduce data contention and reduce the overhead of replicating shared data, like caches or state information.

The goal of partitioning is to provide the ability to control specific resources during cluster member execution. Requests can be routed to a specific application server that has exclusive access to certain computing resources, such as a dedicated server process or database server. The requests can be a HTTP, EJB or database read or update, with the endpoint receiving the work being highly available. WebSphere Partitioning Facility offers functionality to route work to a particular cluster endpoint. This reduces overall system overhead while retaining high availability of each endpoint.

As a result, WebSphere Partitioning Facility is designed to make it easier to develop applications with the following characteristics:

- ▶ Write-intensive applications that traditionally do not scale well because of contention
- ▶ Applications that must process high-speed message feeds
- ▶ Applications with a need for singletons that must be made highly available

It is important to note when designing a partitioned application that the requests should be directed to an application server that has exclusive access to specific resources. The WebSphere Partitioning Facility framework provides tools and APIs for the management of these partitions, such as activation and deactivation, rebalancing of partitions, and when a partition is loaded or unloaded. A partition can also be moved to a different application server at any time.

A partition can be activated on any cluster member in a cluster. The High Availability Manager (HAManager) guarantees there is a single instance of an active partition in the cluster at a given time within the cluster for cluster scoped partitions. The HAManager allows a partition to be moved from one member in the cluster to another. When moving a partition, the partition state will be deactivated on the original cluster member and activated on the new target cluster member.

Partitions are by default highly available. A partition can only be hosted on a single cluster member at a time. They are made highly available using the HAManager. If a cluster member fails as a result of a JVM shutdown or JVM failure, then the HAManager moves all partitions which were running on the failed cluster member to another surviving cluster member.

It is very important to plan how a cluster operates in normal conditions and under failure conditions in high workload environments.

Recommendation: The HAManager monitors many cluster-wide resources. In general, this takes a certain amount of performance. If the cluster members are paging or otherwise engaged so that HAManager functionality cannot operate effectively, then HA-managed events will begin to occur to account for perceived cluster member anomalies.

For this reason, we recommend that you do not put the application servers under a large load in normal cases, in order to better handle spikes at times when challenges arise. In addition, reducing virtual memory paging as much as possible will result in a more reliable cluster operational environment.

The number of partitions in a specific solution should be managed carefully. When possible, having fewer partitions is generally simpler and more efficient. Each partition takes system resources to implement within the workload management function; requires additional administrator effort from a system management perspective; and produces a reduction in cluster performance when tracking, from a performance monitoring perspective.

ObjectGrid

ObjectGrid is an extensible transactional object caching framework for Java 2 Platform, Standard Edition (J2SE™) and Java 2 Platform, Enterprise Edition (J2EE) applications. You can use the ObjectGrid API when developing your applications to retrieve, store, delete, and update objects in the ObjectGrid framework. You can also implement customized plug-ins that monitor updates to the cache; retrieve and store data with external data sources; manage eviction of entries from the cache; and handle background cache functionality for your own ObjectGrid application environment.

The ObjectGrid provides an API that is based on the Java Map interface. The API is extended to support the grouping of operations into transactional blocks. You can associate a set of keywords with a key, and all objects associated with a keyword can be evicted from the Map instance at any time. This interface is a superset of the Map interface and adds support for batch operations, invalidation, keyword association, and explicit insert and update.

You can use the ObjectGrid framework with or without WebSphere Application Server or WebSphere Extended Deployment. When you use WebSphere Application Server support, additional distributed transaction support can be exploited. You can also exploit distributed transactional support if you use any environment that has a reliable publish and subscribe messaging system, such as a Java Message Service (JMS) provider.

The ObjectGrid server run time supports full clustering, replication, and partitioning of distributed object caches. The client run time supports the concept of a near cache and workload management routing logic to remote clusters. The client run time also supports local object map creation. The level of support varies, depending on whether you are running the client run time, server run time, integrated ObjectGrid, or the standalone ObjectGrid.

System p platform configuration

This chapter explains how to configure and manage System p hardware and partitions. It also briefly describes various ways to automate day-by-day operation.

3.1 Setting up System p hardware and partitions

This section explains how to set up System p5™ hardware, as well as the partitions and VIO Server.

3.1.1 Setting up and accessing the Hardware Management Console

The Hardware Management Console (HMC) provides a graphical user interface (GUI) for configuring and operating single or multiple managed systems. The HMC consists of a 32-bit Intel®-based desktop personal computer with a DVD-RAM drive, and runs the Linux operating system. The application environment, with a set of hardware management applications for configuration and partitioning, is written in Java.

The applications are based on the object-oriented schema using the Common Information Model (CIM), an industry standard sponsored by the Distributed Management Task Force (DMTF). A CIM Object Manager acts as repository and database lookup for all managed objects.

The GUI can display dynamic events and static information from pSeries® machines running AIX, as well as from partitions on any partitioning-capable pSeries server. Note the following points:

- ▶ Users who log in to the HMC from the local console access the application using the Web-based System Manager GUI.
- ▶ The HMC communicates with the service processor on the managed system using serial communication.
- ▶ The Resource Monitoring and Control (RMC) subsystem on the HMC connects the RMC subsystem on remote nodes, such as partitions, over the TCP/IP network.
- ▶ A remote user can access the HMC using either the ssh or rexec facility over the TCP/IP network.
- ▶ Users who log in to the HMC from the local console can access the remote Web-based System Manager server on remote nodes, such as AIX partitions, over the TCP/IP network.
- ▶ Users using the remote Web-based System Manager client can access the HMC over the TCP/IP network.

For general information about these topics, refer to *Effective System Management Using the IBM Hardware Management Console for pSeries*, SG24-7038, which is available at the following address:

<http://www.redbooks.ibm.com/abstracts/sg247038.html?open>

Note: This chapter was written before general availability of the V7 HMC. For updated information on HMC usage, especially when using POWER6™ systems, refer to *Hardware Management Console V7 Handbook*, SG24-7491.

Appropriate network configuration

To prevent problems with DLPAR operations on the HMC, as well as Inventory Scout, Service Agent, and Service Focal Point, view the Ethernet network between the HMC and partitions as a mandatory administrative network used for these purposes. With careful network planning, you should not encounter any problems using these applications. However, if an AIX administrator mistakenly changes the TCP/IP configuration on a partition without notifying the HMC administrator, it might result in severe communication problems.

Remote access to the HMC GUI

The HMC allows remote access to the GUI from the Web-based System Manager client installed on the following operating systems:

- ▶ AIX
- ▶ Windows

In this publication we explain how to set up and use the Web Based System Manager Remote Client Interface to enable systems management through remote connection.

As described in *Effective System Management Using the IBM Hardware Management Console for pSeries*, SG24-7038, we assume the following conditions:

- ▶ HMC is already configured for remote Management Access
- ▶ Web Based System Manager Remote Client has already been downloaded and installed to the Client System from the HMC using a link similar to:

http://<HMC Hostname>/remote_client.html

Accessing the HMC through WebSM

To access the HMC through WebSM, start the local Java-based Web Based System Manager Remote Client application by double-clicking the WebSM icon shown in Figure 3-1 on page 40.

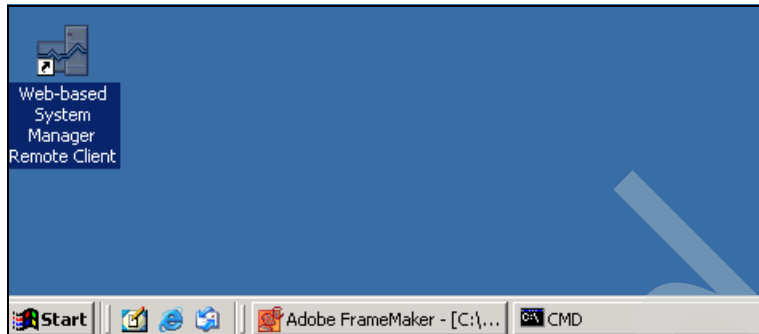


Figure 3-1 WebSM Client startup icon

The Java-based Web Based System Manager Remote Client loading Window displays, as shown in Figure 3-2.

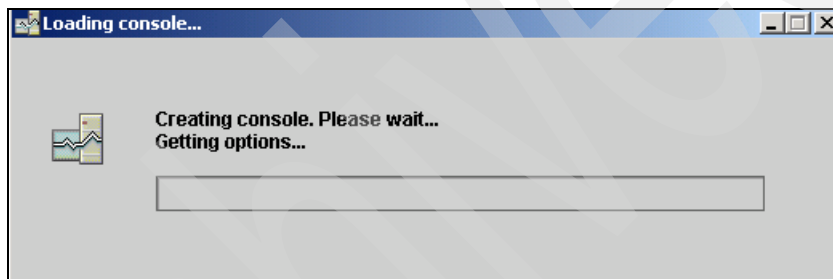


Figure 3-2 WebSM loading the console

After the client software is started, you are prompted for the host name of the server you want to connect. Figure 3-3 shows we used the host name of our HMC `riogrande.itsc.austin.ibm.com`. You can use any AIX server that is enabled for direct WebSM management.

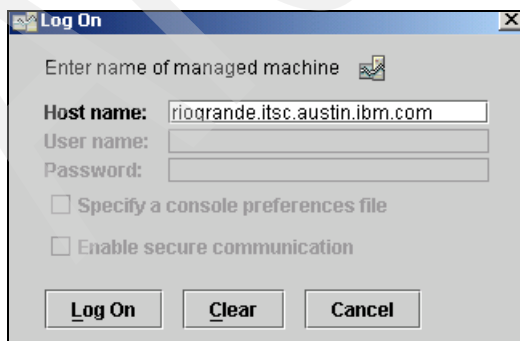


Figure 3-3 WebSM Host name for HMC login panel

The Information panel, shown in Figure 3-4, prompts you to acknowledge a non-secure SSL connection. We recommend setting up a secure HMC connection before continuing to use the remote client.

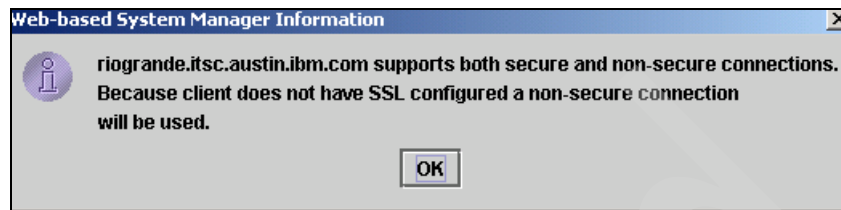


Figure 3-4 WebSM HMC connection SSH informational message

In our case, we now had to provide a valid HMC user and password, as shown in Figure 3-5, to complete the Login process. As shown, we used the ID hscroot to create a functional user and role to access only the managed systems we were responsible for.

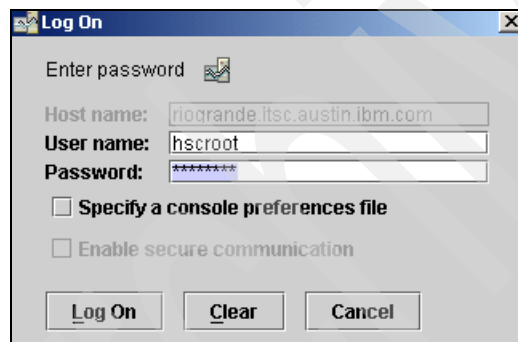


Figure 3-5 WebSM user for HMC login panel

We named the systems p5-9113-550-SN104790E and p5+-9133-55A-SN10D1FAG. Figure 3-6 on page 42 shows the change in the Managed Environment View of WebSM, including the full access for the HMC user hscroot.

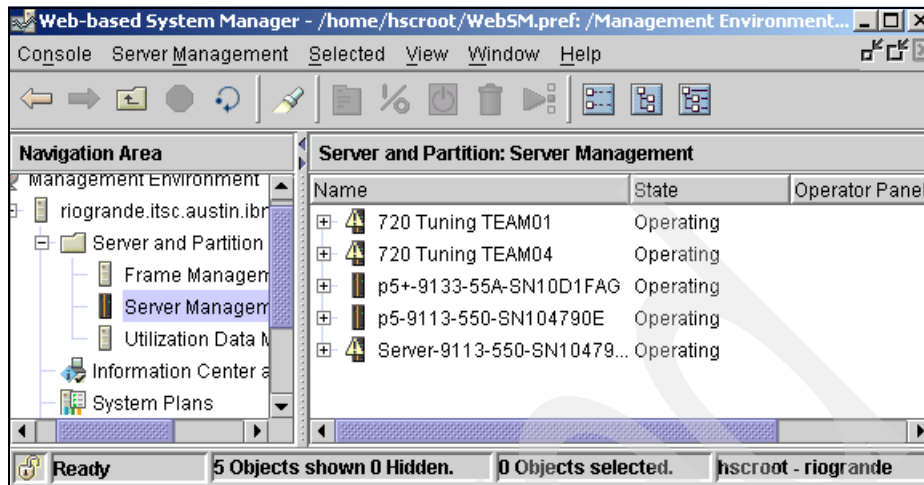


Figure 3-6 WebSM HMC main view hscroot

To limit access to a certain group of managed systems in our lab environment:

- We created a Managed Resources Role WebSphere Application Server P5I and assigned two managed systems to it, as shown in Figure 3-7.

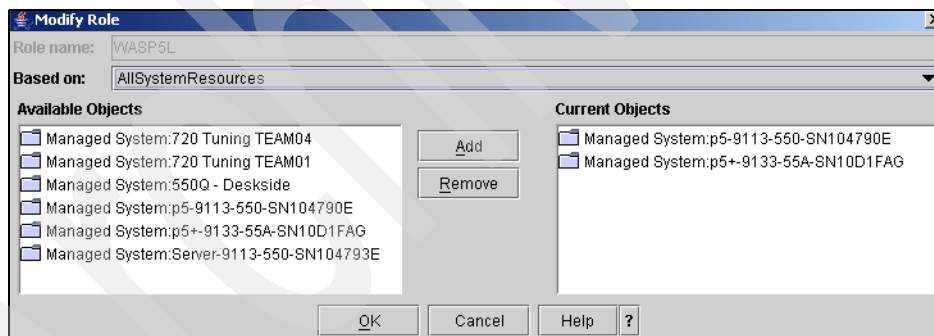


Figure 3-7 HMC Managed Resources Role WebSphere Application Server P5L

- We created a user named wasp5l that is assigned Advanced Operator as the Task Role and WebSphere Application Server P5I as the Managed Resources Role.

After the change, we exited the WebSM Session, restarted the WebSM Session and relogged in to the HMC as user wasp5l. The Managed Environment View then only showed the two managed systems p5-9113-550-SN104790E and p5+-9133-55A-SN10D1FAG, which were assigned to our lab; see Figure 3-8 on page 43.

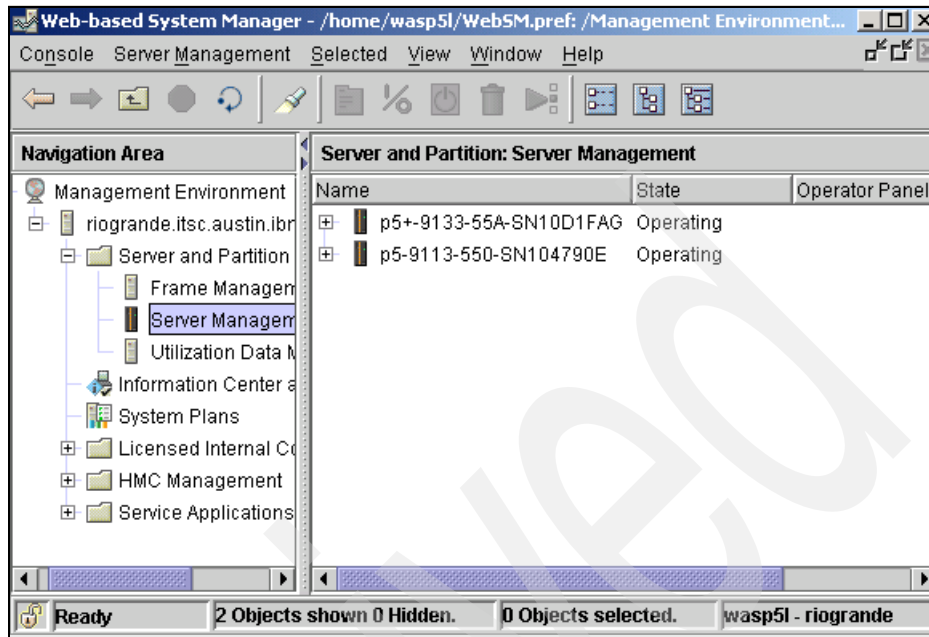


Figure 3-8 WebSM HMC main view wasp5l limited view

Figure 3-9 on page 44 shows the final HMC Managed System view.

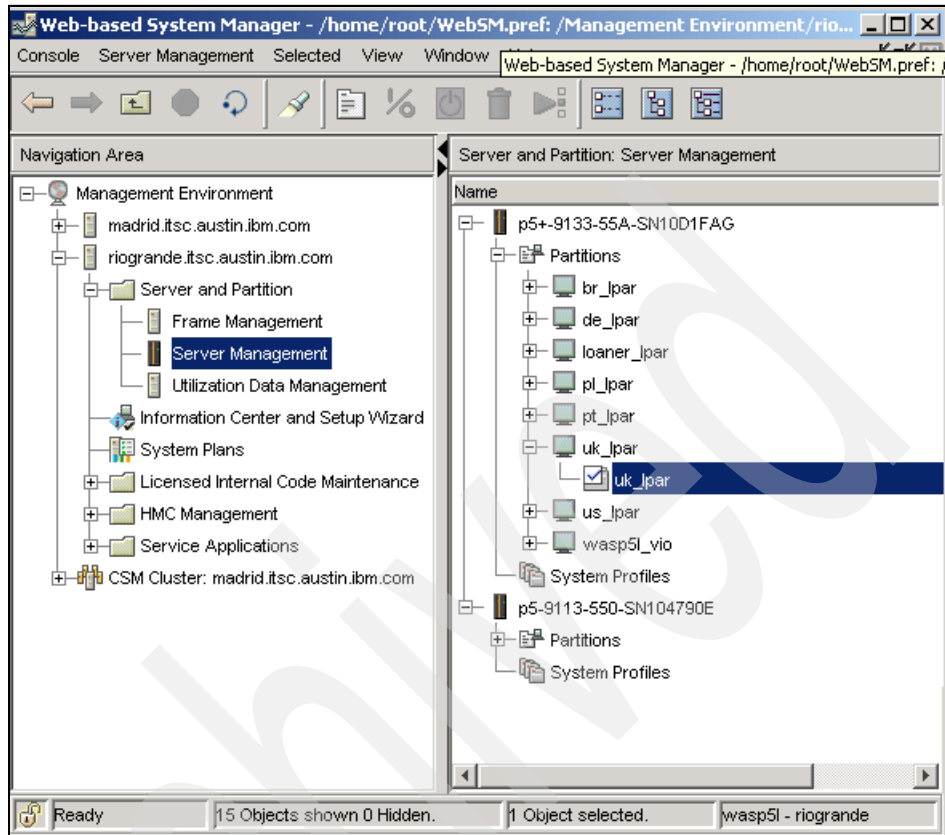


Figure 3-9 HMC Managed Server all nodes view

HMC command line interface

Typically, HMCs are placed inside the machine room where managed systems are located, so you might not be allowed to physically access the HMC. However, you can remotely access it using either the remote Web-based System Manager client or the command line interface. The command line interface of the IBM Hardware Management Console for pSeries (HMC) is especially useful in the following optimization situations:

- If you have to manage several systems, you can achieve consistent results by using the command line interface. The command sequence can be stored in scripts and executed remotely.

- ▶ After you have developed a consistent way to manage your managed systems, you can automate the operations by invoking the scripts from batch processing applications (such as the cron daemon) from other systems.

The HMC provides a set of commands so it can be used for many management tasks; however, those commands are only accessible from the remote system, and *not* from the HMC local console. Any TCP/IP-capable system that supports ssh can use the remote execution of command line functions.

You have two ways to remotely execute the command line interface on the HMC using OpenSSH:

- ▶ Execute commands remotely.

The following sample shows a remote user on a remote AIX system executing the `/opt/hsc/bin/lshmc -r` command as the `wasp5l` user on the HMC (riogrande). You will be prompted to enter the login password of the `hscroot` user, and then the command will list the status of the remote command execution configuration:

```
$ ssh wasp5l@riogrande /opt/hsc/bin/lshmc -r
wasp5l@riogrande password: XXXXXX
Remote Command Execution Configuration:
Remote command execution using the ssh facility:    enabled
```

- ▶ Execute commands after logging in to the HMC.

The following sample shows that the `lshmc -r` command is executed on the HMC after logging in to the HMC as the `wasp5l` user:

```
$ ssh wasp5l@riogrande
wasp5l@riogrande password: XXXXXX
Last login: Mon Sept 26 14:32:13 2006 from madrid.itsc.austin.ibm.com
[wasp5l@riogrande]$ lshmc -r
Remote Command Execution Configuration:
Remote command execution using the ssh facility:    enabled
```

A collection of some HMC commands are listed in Table 3-1.

Table 3-1 Sample HMC commands

Command	Short description
lshwres	Lists hardware resources of a managed system

Command	Short description
lssyscfg	Lists attributes of partitions, partition profiles, or the managed system
chhwres	Adds, removes, or moves hardware resources of a managed system
chsysstate	Changes the state of a partition (for example, power off)
chsyscfg	Changes attributes of partitions, partition profiles, or the managed system
lshwinfo	Lists environmental information, such as input power levels and ambient air temperatures
lsparutil	Lists utilization metrics for the managed system and partitions
viosvr cmd	Issues an I/O server command line interface (ioscli) command to a Virtual I/O Server

For a full explanation of all HMC commands, refer to HMC Command Descriptions, which is available at the following address:

<http://publib.boulder.ibm.com/infocenter/eserver/v1r3s/index.jsp?topic=/iphcx/hmckickoff.htm>

3.1.2 Basic managed system operation

Managed systems are physically attached to and managed by the HMC. The HMC can perform tasks that affect the entire managed system, such as powering the system on and off. You can also create partitions and profiles within each managed system. These partitions and profiles define the way you configure and operate your partitioned system.

Within your managed system, you can assign resources to create partitions. Each partition runs a specific instance of an operating system. The HMC can perform tasks on individual partitions. These tasks are similar to those you can perform on traditional, non-partitioned servers. For example, you can use the HMC to start the operating system and access the operating system console.

Because the HMC provides a virtual terminal for each partition, a terminal window can be opened for each partition. This virtual terminal can be used for software installation, system diagnostics, and system outputs. Only one instance

of the terminal can be opened at a time. The managed system firmware and device drivers provide the redirection of the data to the virtual terminal.

A profile defines a configuration setup for a managed system or partition. You can then use the profiles you created to start a managed system or partition in a particular configuration.

Basic operation on the managed system cover the following tasks:

- ▶ Viewing properties of the managed system
- ▶ Power on/Power off the managed system
- ▶ Create/Activate/Reset/Shut down partitions
- ▶ Create/Back up/Delete/Initialize the managed system profile

Managed system information

To view the properties of your managed system, you select the managed system in the Contents area. The Properties panel includes the five property tabs of the managed system, as listed in Table 3-2.

Table 3-2 Properties of the managed system

Property name	Properties
Machine	Serial Number, Model/Type, Service Processor Version, Capability
Processor	installed processors, identified by their processor ID and their assignment to partitions
Policy	Service Processor Surveillance Policy
I/O Slot	Assignment of I/O slots to partitions and adapter-type information
Memory	Assigned memory amount to partitions and page table usage information

To illustrate the Properties output of a managed system, Figure 3-10 on page 48 displays the details of the allocated memory to all partitions.

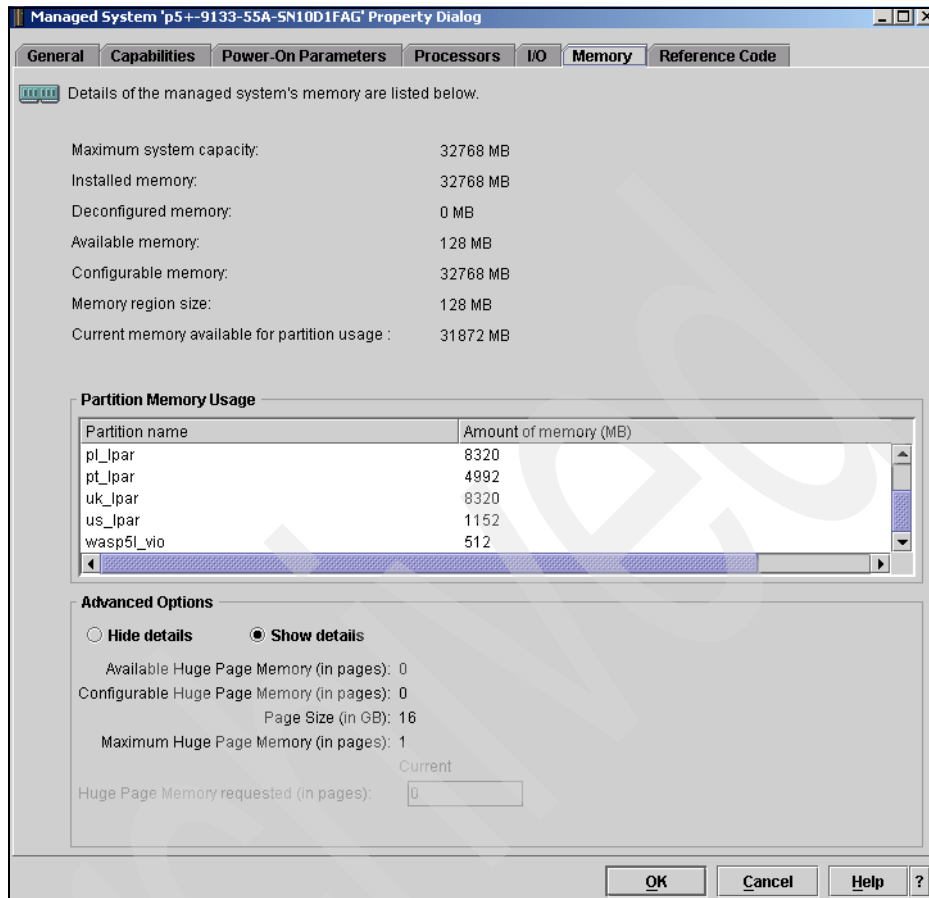


Figure 3-10 Managed Systems Memory Properties tab

System Information can be retrieved by using HMC command line remote execution as shown in Example 3-1. The HMC command **lssyscfg** with option **-r sys** lists the attributes for all managed systems.

Example 3-1 Output of HMC command lssyscfg for all managed systems

```
#ssh wasp5l@riogrande "lssyscfg -r sys"
wasp5l@riogrande's password:xxxxxxx
name=p5+-9133-55A-SN10D1FAG,type_model=9133-55A,serial_num=10D1FAG,ipadr=192.168.252.255,state=Operating,sys_time=09/29/2006
17:23:26,power_off_policy=1,cod_mem_capable=0,cod_proc_capable=1,hca_capable=1,huge_page_mem_capable=1,micro_lpar_capable=1,os400_capable=0,5250_application_capable=0,redundant_err_path_reporting_capable=1,shared_eth_failover_capable=1,sni_msg_passing_capable=0,sp_failover_capable=0,
```

```
vet_activation_capable=1,virtual_io_server_capable=1,assign_5250_cpw_percent=0,max_lpars=40,max_power_ctrl_lpars=1,service_lpar_id=none,curr_sys_keylock=norm,pend_sys_keylock=norm,curr_power_on_side=temp,pend_power_on_side=temp,curr_power_on_speed=fast,pend_power_on_speed=fast,curr_power_on_speed_override=none,pend_power_on_speed_override=none,power_on_type=power
on,power_on_option=autostart,pend_power_on_option=autostart,power_on_method=02,power_on_attr=0000,sp_boot_attr=0000,sp_boot_major_type=08,sp_boot_minor_type=01,sp_version=00030030,mfg_default_config=0,curr_mfg_default_ipl_source=a,pend_mfg_default_ipl_source=a,curr_mfg_default_boot_mode=norm,pend_mfg_default_boot_mode=norm
#
```

The managed dystem profile file

When a partition is created, a profile also has to be created by default to define the resources associated with this partition. The application requires that you create at least one profile when a partition is created. The first profile created is the default profile. The default partition profile can be changed at any time.

The profile data for partitions of a managed system is stored in at least the following three locations at any given time:

- ▶ NVRAM of the managed system
- ▶ CIM Object Manager on the HMC
- ▶ Profile data backup files under the /var/hsc/profile/MT-MDL*S/N directory on the HMC

3.1.3 Basic partition operations

Partitioning your system is similar to partitioning a hard drive. When you partition a hard drive, you divide a single physical hard drive so that the operating system recognizes it as a number of separate logical hard drives.

You have the option of dividing the system's resources (memory, processor, or adapters) by using the HMC to partition your system. On each of these partitions, you can install an operating system and use each partition as you would a separate physical system. With the introduction of Virtual I/O Server, it is possible to further partition I/O adapters.

Before you begin to create partitions, complete the following activities:

- ▶ Record the required subnet mask, any gateway information, and address of your DNS server.
- ▶ Check that you have a suitable LAN (hub or switch and cables) to connect to each HMC and each network adapter used by partitions.

- ▶ Record the TCP/IP names and addresses to be resolved by a DNS server, or to be entered into the `/etc/hosts` file in each partition, and on the HMC.

Determine the following information:

- ▶ Your current resources for each partition
- ▶ The operating system host name for each partition
- ▶ The partition you want to use for service actions
- ▶ The operating system to be loaded on the partition

For more details about this topic, refer to *Partitioning Implementations for IBM eServer p5 Servers*, SG24-7039, and *IBM eServer Certification Study Guide eServer p5 and pSeries Enterprise Technical Support AIX 5L V5.3*, SG24-7197.

Partition resources

Logical partitioning allows you to assign dedicated processors or, when you use the Micro-Partitioning™ feature of POWER5- and POWER6-based systems, to assign processing units to partitions. You can define a partition with a processor capacity as small as 0.10 processing units, which represents 10% of a physical processor. You can also assign physical memory and physical I/O devices or virtual I/O-devices (SCSI or Ethernet) to partitions.

Partition and system profiles

The information about resources that are assigned to a partition is stored in a partition profile. Each partition can have multiple partition profiles. By switching from one partition profile to another, you can change how resources are assigned. To change partition profiles, you must shut down the operating system instance that is running in the partition and stop (deactivate) the partition. There are two types of profiles, partition and system, as explained here:

- ▶ Partition profile

A partition profile stores the information about the assigned resources for a specific partition, such as processor, memory, physical I/O devices, and virtual I/O devices (Ethernet, serial, and SCSI). Each partition must have a unique name and at least one partition profile. A partition can have several partition profiles, but it reads only one partition profile when it is started (activated).

You select a partition profile when you activate the partition. Otherwise, the default partition profile is used. You can designate any partition profile as the default partition profile. If there is only one partition profile for a partition, it is always the default.

► System profile

A system profile provides a collection of partition profiles that should be started at the same time. The partition profiles are activated in the order of the list that is defined in the system profile.

Processors

Partitions can have processors dedicated to them, or they can have their processors virtualized from a pool of shared physical processors. This is known as Micro-Partitioning technology. With this technology, both types of partitions can coexist in the same system at the same time.

Micro-Partitioning technology differs from dedicated processor partitions in that physical processors are abstracted into virtual processors, which are then assigned to partitions. These virtual processors have capacities ranging from 10% of a physical processor up to the entire processor. Therefore, a system can have multiple partitions that share the same physical processor, and that divide the processing capacity among themselves.

A dedicated processor partition, such as the partitions that are used on POWER4™ processor-based servers, have an entire processor that is assigned to a partition. These processors are owned by the partition where they are running and are not shared with other partitions. Also, the amount of processing capacity on the partition is limited by the total processing capacity of the number of processors configured in that partition, and it cannot go over this capacity (unless you add or move more processors from another partition to the partition that is using a dynamic LPAR operation).

Memory

To support any operating system (including AIX and Linux) which requires real mode code execution and the ability to present a real address space starting at zero (0) to each partition in the system, the logical memory concept is adopted.

Logical memory is an abstract representation that provides a contiguous memory address to a partition. Multiple non-contiguous physical memory blocks are mapped to provide a contiguous logical memory address space.

The logical address space provides the isolation and security of the partition operating system from direct access to physical memory, allowing the hypervisor to police valid logical address ranges assigned to the partition. The contiguous nature of the logical address space is used more for simplifying the hypervisor's per-partition policing than it is used because it is an operating system requirement. The operating system's VMM handles the logical memory as though it were physical memory in a non-partitioned environment.

In a partitioned environment, some of the physical memory areas are reserved by several system functions to enable partitioning in the partitioning-capable pSeries server. You can assign unused physical memory to a partition. You do not have to specify the precise address of the assigned physical memory in the partition profile, because the system selects the resources automatically.

Depending on the overall memory in your system and the maximum memory values that you choose for each partition, the server firmware must have enough memory to perform logical partition tasks. Each partition has a Hardware Page Table (HPT). The size of the HPT is based on an HPT ratio of 1/64 and is determined by the maximum memory values that you establish for each partition.

Physical I/O slots

Physical I/O devices are assignable to partitions on a PCI slot (physical PCI connector) basis. It is not the PCI adapters in the PCI slots that are assigned as partition resources, but the PCI slots into which the PCI adapters are plugged.

When using physical I/O devices to install an operating system, you have to assign at least one, typically an SCSI adapter that is able to boot the operating system, and an adapter to access the install media. Instead of physical I/O devices, you can assign a virtual I/O device that behaves like a physical I/O device.

After installation, you need at least one physical device adapter that is connected to the boot disk or disks. For application use and system management purposes, you also have to assign at least one physical network adapter. You can allocate physical slots in any I/O drawer on the system.

Virtual I/O

Virtual I/O allows a server to support more partitions than it has slots for I/O devices by enabling the sharing of I/O adapters between partitions.

- ▶ Virtual Ethernet enables a partition to communicate with other partitions without the need for an Ethernet adapter. A shared Ethernet adapter, supported by the Virtual I/O Server, allows a shared path to an external network.
- ▶ Virtual SCSI enables a partition to access block-level storage that is not a physical resource of that partition. With the Virtual SCSI design, the virtual storage is backed by a logical volume on a portion of a disk or an entire physical disk. These logical volumes appear to be the SCSI disks on the client partition, which gives the system administrator maximum flexibility in configuring partitions.

Minimum, desired, and maximum values

In a partition profile, you need to specify three kinds of values for each resource. For memory, you must specify minimum, desired, and maximum values.

For processor, you define whether you use dedicated or shared processors.

- ▶ If you chose to use dedicated processor, you can specify minimum, desired, and maximum values.
- ▶ For shared processors, you need to specify minimum, desired, and maximum values for both processing units and virtual processors.

For physical and virtual I/O slots, you must specify the required and desired values.

If any of the three types of resources cannot satisfy the specified minimum and required values, the activation of a partition fails. If the available resources satisfy all the minimum and required values but do not satisfy the desired values, the activated partition will get as many of the resources that are available.

Logical Partition creation

For planning and implementation guidance about creating LPARs, refer to the IBM Redbooks publication *LPAR Simplification Tools Handbook*, SG24-7231. Before you start to create partitions, you need to plan your layout in terms of:

- ▶ Which type of Partition: LPAR, DLPAR, Virtual I/O
- ▶ Available physical resources in terms of: Network Adapter, Storage Adapter

You can verify your plan by using the system planning tool:

<http://www-03.ibm.com/servers/eserver/support/tools/systemplanningtool>

Tip: If you plan to use a virtual device for your partition, we recommend that you first install and configure the VIO Server and virtual device before creating a logical partition.

To create a logical partition, perform the following steps within the HMC Console View:

1. In the Contents area, select the managed system.
2. From the Selected menu, select **Create**.
3. Select **Logical Partition**. The Create Logical Partition and Profile wizard opens, as shown in Figure 3-11 on page 55.
4. In the first window of the Create Logical Partition and Profile wizard, provide a name for the partition profile that you are creating. Use a unique name for

each partition that you create. Names can be up to 31 characters in length. Then click **Next**.

5. Type the name of the profile you are creating for this partition. Click **Next** (unless you want to assign the partition to a workload management group).
6. Select the desired, minimum, and maximum number of processors you want for this partition profile, then click **Next**.
7. Select the desired and minimum number of memory, then click **Next**.
8. The left side of the new window displays the I/O drawers available and configured for use. To expand the I/O tree to show the individual slots in each drawer, click the icon next to each drawer. Because the HMC groups some slots, if you attempt to assign a member of one of these grouped slots to a profile, the entire group is automatically assigned. Groups are indicated by a special icon named Group_XXX.

Select the slot for details about the adapter installed in that slot. When you select a slot, the field underneath the I/O drawer tree lists the slot's class code and physical location code.

9. Select the slot you want to assign to this partition profile and click either **required** or **desired**. If you want to add another slot, repeat this process.

Note: The slots in the I/O drawers field are not listed in sequential order.

Slots are added individually to the profile; you can add slots one at a time, unless they are grouped. Minimally, assign a boot device to the required list box. This might be real or virtual depending on the choice of your server.

There are two groups to which you can add adapters:

- A desired group
- A required group

Desired adapters will be used if they are available at the time of activation. Required adapters are adapters that you require for this partition. If the adapters in this group are not available at the time of activation, the partition does not activate. Click **Next**.

10. This window enables you to set service authority and boot mode policies for this partition profile. Select the boot mode that you want for this partition profile, then click **Next**.
11. This window supplies you with summary information about this partition. Review the information to ensure that you have the appropriate resources assigned to this partition.

If you want to change the configuration, click **Back**. Otherwise, click **Finish** to create the partition and profile.

The new partition, along with the default profile you just created, appears under the Managed System tree in the Contents area. After you have created a partition, you must install an operating system and configure inventory data collection on the HMC and on the partition.

Figure 3-11 shows the Partition creation wizard initial screen, where you can choose the Partition environment and provide the Partition name.

This wizard helps you create a new logical partition and a default profile for it. You can use the partition properties or profile properties to make changes after you complete this wizard.

Ensure you have your logical partition planning information before you use this wizard. You may also find it helpful to be familiar with logical partition concepts. Click Help for more information.

To create a partition, complete the following information:

System name : p5+-9133-55A-SN10D1FAG

Partition ID : 9

Partition name :

Partition environment

☒ AIX or Linux

☐ i5/OS

☐ Virtual I/O server

Help ? < Back Next > Finish Cancel

Figure 3-11 Partition creation wizard - creating an AIX logical partition

3.1.4 Advanced partition operations

A dynamic logical partition provides you with the ability to logically attach and detach a managed system's resources to and from a partition's operating system without rebooting. You can perform these tasks only by using the HMC GUI or command line interface.

Script-based LPAR event handling

Management of LPARs can be automated by using the SSH remote command execution functionality of the HMC. With this functionality you can create automated processes which are managed and executed from a central system through SSH remote execution (for example, you can select a server to perform central operations like the NIM or CSM Server).

Single and simple remote command execution

“Single and simple remote command execution” means that you can execute command after command on the HMC command line interface as illustrated in “HMC command line interface” on page 44 and the Sample Commands Table 3-1 on page 45. You can use the HMC command line interface to retrieve significant information by using a single command, or change the LPAR configuration.

For example, to obtain a list of all LPARs and their profile properties for Managed System p5+-9133-55A-SN10D1FAG, use the command `lssyscfg`. Example 3-2 shows output from the HMC command `lssyscfg` executed from the CSM Management Server `madrid.itsc.austin.ibm.com` using a ssh session:

```
ssh wasp5l@riogrande "lssyscfg -r lpar -m p5+-9133-55A-SN10D1FAG"
```

The command was passed to the HMC named `riogrande.itsc.austin.ibm.com` using the user ID `wasp5l`.

Example 3-2 Output of HMC command lssyscfg for Managed System

```
#ssh wasp5l@riogrande "lssyscfg -r lpar -m p5+-9133-55A-SN10D1FAG"
wasp5l@riogrande's password:
name=wasp5l_vio,lpar_id=1,lpar_env=vioserver,state=Running,resource_con
fig=1,os_version=0.0.0.0.0.0,logical_serial_num=10D1FAG1,default_profil
e=wasp5l_vio_limited,curr_profile=wasp5l_vio_limited,work_group_id=none
,shared_proc_pool_util_auth=0,power_ctrl_lpar_ids=none,boot_mode=norm,l
par_keylock=norm,auto_start=1,redundant_err_path_reporting=0
name=de_lpar,lpar_id=2,lpar_env=aixlinux,state=Running,resource_config=
1,os_version=0.0.0.0.0.0,logical_serial_num=10D1FAG2,default_profile=lo
aner_lpar,curr_profile=loaner_lpar,work_group_id=none,shared_proc_pool_
util_auth=0,power_ctrl_lpar_ids=none,boot_mode=norm,lpar_keylock=norm,a
uto_start=0,redundant_err_path_reporting=0
name=br_lpar,lpar_id=3,lpar_env=aixlinux,state=Running,resource_config=
1,os_version=0.0.0.0.0.0,logical_serial_num=10D1FAG3,default_profile=br
_lpar,curr_profile=br_lpar,work_group_id=none,shared_proc_pool_util_aut
h=0,power_ctrl_lpar_ids=none,boot_mode=norm,lpar_keylock=norm,auto_star
t=0,redundant_err_path_reporting=0
name=uk_lpar,lpar_id=4,lpar_env=aixlinux,state=Running,resource_config=
1,os_version=0.0.0.0.0.0,logical_serial_num=10D1FAG4,default_profile=uk
_lpar,curr_profile=uk_lpar,work_group_id=none,shared_proc_pool_util_aut
```

```

h=0,power_ctrl_lpar_ids=none,boot_mode=norm,lpar_keylock=norm,auto_star
t=0,redundant_err_path_reporting=0
name=us_lpar,lpar_id=5,lpar_env=aixlinux,state=Running,resource_config=
1,os_version=0.0.0.0.0,logical_serial_num=10D1FAG5,default_profile=us
_lpar,curr_profile=us_lpar,work_group_id=none,shared_proc_pool_util_aut
h=0,power_ctrl_lpar_ids=none,boot_mode=norm,lpar_keylock=norm,auto_star
t=0,redundant_err_path_reporting=0
name=pt_lpar,lpar_id=6,lpar_env=aixlinux,state=Running,resource_config=
1,os_version=0.0.0.0.0,logical_serial_num=10D1FAG6,default_profile=pt
_lpar,curr_profile=pt_lpar,work_group_id=none,shared_proc_pool_util_aut
h=0,power_ctrl_lpar_ids=none,boot_mode=norm,lpar_keylock=norm,auto_star
t=0,redundant_err_path_reporting=0
name=pl_lpar,lpar_id=7,lpar_env=aixlinux,state=Running,resource_config=
1,os_version=0.0.0.0.0,logical_serial_num=10D1FAG7,default_profile=pl
_lpar,curr_profile=pl_lpar,work_group_id=none,shared_proc_pool_util_aut
h=0,power_ctrl_lpar_ids=none,boot_mode=norm,lpar_keylock=norm,auto_star
t=0,redundant_err_path_reporting=0
#

```

Example 3-3 illustrates the same approach, but in this case we limited the output to the Partition Name (option value name) and Operating Environment (option value lpar_env).

Example 3-3 Output of HMC command lssyscfg for LPAR Env

```

[0:root@MADRID:]/home/root # ssh wasp5l@riogrande
wasp5l@riogrande's password:
Last login: Mon Oct  2 15:51:11 2006 from madrid.itsc.austin.ibm.com
wasp5l@riogrande:~> lssyscfg -r lpar -m p5+-9133-55A-SN10D1FAG -F
name,lpar_env
wasp5l_vio,vioserver
de_lpar,aixlinux
br_lpar,aixlinux
uk_lpar,aixlinux
us_lpar,aixlinux
pt_lpar,aixlinux
pl_lpar,aixlinux

```

In both of these examples, you would need to manually enter the password for user wasp5l. The next step would be to enable remote execution without a password prompt. If you want to automate this procedure, perform an SSH key exchange from your management console onto the HMC. For a detailed explanation of this task, refer to *Setting up secure script execution between SSH clients and the HMC*, which is available at the following address:

<http://publib.boulder.ibm.com/infocenter/eserver/v1r3s/topic/iphai/settingupsecurecriptexecutionsbetweensshclientsandthehmc.htm>

Complex remote command execution

A more complex remote command execution can be a script to run on AIX that sends commands via ssh into the HMC to collect information about the HMC, the systems managed by the HMC, and the LPARs in those systems.

For a detailed explanation of how to have this script run without prompting for the password of the HMC user, see the Techdocs Library document *passAIX ssh client to pSeries HMC*, which is available at the following address:

<http://www-03.ibm.com/support/techdocs/atsmastr.nsf/032f6e163324983085256b79007f5aec/f962418b4b10f21c86256dc6004abcf?OpenDocument>

Automation of partition start up and shut down sequence

After you have set up secure script execution between SSH clients and the HMC, you can start using automated procedures. For example, you may want to remotely start all logical partitions with another system profile as the default assigned one in order to check changed settings before activating them by default. If the new system profile (Virtual_IO_Client_Profile) exists, you could use the following command:

```
chsysstate -m p5+-9133-55A-SN10D1FAG -o on -r sysprof -n  
Virtual_IO_Client_Profile
```

As another example, suppose you want to stop and start an LPAR in order to activate a partition profile for testing purposes. This testing would take place after business hours, and without manual intervention. If the temporary partition profile *de_limited* exists, the following commands would be executed by a scheduled process.

1. Use the **chsysstate** command to stop the partition by issuing the **shutdown -F** command to shut down an AIX or Virtual I/O Server partition.

```
chsysstate -m p5+-9133-55A-SN10D1FAG -r lpar -o osshutdown -n  
de_lpar --immed
```

2. Issue the command **lssyscfg** to get the actual state of the partition.

```
lssyscfg -r lpar -m p5+-9133-55A-SN10D1FAG --filter  
""lpar_names=de_lpar"" -F state
```

3. If the command returns Not Activated for the state of the partition, start the partition.

```
chsysstate -m p5+-9133-55A-SN10D1FAG -r lpar -o on --id 2 -f  
de_limited
```

The same result can be achieved by a single command while using the **--restart** option together with the **-o osshutdown** option:

```
chsysstate -m p5+-9133-55A-SN10D1FAG -r lpar -o osshutdown --restart
-n de_lpar --immed
```

Applications and the drmgr command

By default, an application running in a DLPAR does not receive any information about changes in the resources; instead, the application has to explicitly register itself to receive such resource change notifications. When a DR event occurs, the applications are made aware of resource changes if they have registered a DR script. The DR script can be any type of shell script, Perl script, or even an AIX 5L executable.

The DR manager notifies the registered DR scripts three times after a DLPAR operation request is initiated using either the graphical user interface or command line interface on the HMC.

- ▶ During the check phase (before doing anything, an application can deny or grant the DR event from proceeding at this time)
- ▶ During the pre-phase (just before altering a resource)
- ▶ During the post-phase (after altering a resource)

The pre-phase and the post-phase run if the outcome of the check phase was a grant.

The DR manager calls each DR script with the name of the phase as the first argument. For example, checkacquire indicates the check phase for the resource add, preacquire indicates the pre-phase for the resource add, and postacquire indicates the post-phase of the resource add. Similarly, checkrelease, prerelease, and postrelease are the first arguments for the resource remove. The resource type is identified (CPU or memory) using the second argument.

Application scripts are invoked for both add and remove operations. When removing resources, scripts are provided to resolve conditions imposed by the application that prevent the resource from being removed. The presence of particular processor bindings and the lack of pinnable memory might cause a remove request to fail. A set of commands, as listed in Table 3-3 on page 59, is provided to identify these situations so that scripts can be written to resolve them.

Table 3-3 DLPAR processes resource dependencies

Command	Purpose
ps	Display bindprocessor attachments and block system call status at the process level.

Command	Purpose
bindprocessor	Display online processors and make new attachments.
kill	Send signals to processes.
ipcs	Display pinned shared memory segments at the process level.
lsrest	Display processor sets.
lsclass	Display Workload Manager (WLM) classes, which might include processor sets.
chclass	Change WLM class definitions.

The **drmgr** command must be used to manage DLPAR scripts. The following functions are provided by the **drmgr** command:

- ▶ Listing the registered DLPAR scripts and showing their information.
- ▶ Registering or uninstalling DLPAR scripts in the DLPAR script database.
- ▶ Changing the script install directory path. The default directory is `/usr/lib/dr/scripts/all`.

You can use the command **drmgr -l** to check all installed DLPAR scripts on the Partition. Example 3-4 illustrates checking all partitions by using the **dsh** command executed on the CSM Management Server.

Example 3-4 drmgr -l output for all nodes

```
#dsh -n pt,pl,uk,br "drmgr -l"
pt.itsc.austin.ibm.com: DR Install Root Directory: /usr/lib/dr/scripts
pt.itsc.austin.ibm.com: Syslog ID: DRMGR
pl.itsc.austin.ibm.com: DR Install Root Directory: /usr/lib/dr/scripts
pl.itsc.austin.ibm.com: Syslog ID: DRMGR
uk.itsc.austin.ibm.com: DR Install Root Directory: /usr/lib/dr/scripts
uk.itsc.austin.ibm.com: Syslog ID: DRMGR
br.itsc.austin.ibm.com: DR Install Root Directory: /usr/lib/dr/scripts
br.itsc.austin.ibm.com: Syslog ID: DRMGR
```

For more detailed information about this topic, refer to IBM Redbooks publication *Partitioning Implementations for IBM eServer p5 Servers*, SG24-7039, and to the following address:

http://publib16.boulder.ibm.com/pseries/en_US/aixprgdd/genprogcd/dynamic_reconfig.htm

Additionally, sample DLPAR scripts are provided in the directory `/usr/samples/dr/scripts`, which is included in the AIX fileset `bos.adt.samples`; refer to 4.3, “AIX base operating system samples” on page 132.

Dynamically changing resources

You can change resources, such as processors, memory, and I/O slots, to a partition without rebooting the partition's operating system. Table 3-4 lists and explains the rules to be applied while dynamically changing memory and processor resources.

Table 3-4 Rules for changing LPAR resources

Dynamic task	Resource memory	Resource processor
add	You can only add up to the amount of free memory, or memory that is not assigned to a running partition. You cannot exceed the maximum number specified in the partition's active profile.	You can add up to the amount of free system processors (processors that are not assigned to a running partition) to a partition. You cannot exceed the maximum number specified in the partition's active profile.
remove	The size of the memory remaining after the removal operation cannot be less than the minimum value specified in this partition's active profile.	The number of processors remaining after the removal operation cannot be less than the minimum value specified in this partition's active profile.
move	The partition whose memory will be removed must be assigned more than, or equal to, the memory size defined in the partition profile as the minimum value after the operation.	The partition whose processors will be removed must be assigned more than, or equal to, the number of processors defined in the partition profile as the minimum value after the operation.

Changing resources using HMC command line interface

Resources can be changed through the HMC command line interface by using the **chhwres** command and the option **-o**. Example 3-5 on page 61 illustrates adding two processing units to the DLPAR named `uk_lpar`, and verifying the result using the **lshwres** command.

Example 3-5 *chhwres* command to add two procs to an DLPAR

```
#lshwres -r proc -m p5+-9133-55A-SN10D1FAG --level lpar --filter
"lpar_names=uk_lpar"
lpar_name=uk_lpar,lpar_id=4,curr_min_procs=1,curr_procs=1,curr_max_proc
s=4
#chhwres -r proc -m p5+-9133-55A-SN10D1FAG -o a -p uk_lpar --procs 2
```

```
#lshwres -r proc -m p5+-9133-55A-SN10D1FAG --level lpar --filter  
"lpar_names=uk_lpar"  
lpar_name=uk_lpar,lpar_id=4,curr_min_procs=1,curr_procs=3,curr_max_procs=4
```

Example 3-5 shows that the value for the property `curr_procs` for the available processing units changed from 1 to 3. The change can be seen on the LPAR AIX Operating System in Example 3-6; the count of processors available to the operating system has changed from 2 to 6.

Example 3-6 bindprocessor output for changed DPLAR

```
[0:root@uk:]/home/root # bindprocessor -q  
The available processors are: 0 1  
#  
[0:root@uk:]/home/root # bindprocessor -q  
The available processors are: 0 1 2 3 4 5
```

Changing resources using the HMC GUI

“Changing resources using HMC command line interface” on page 61 explains how to change the available processing units for an LPAR.

You can achieve the same result by using the HMC GUI to select the LPAR you want to change and open the Add Processor Tab; see Figure 3-12 on page 63.

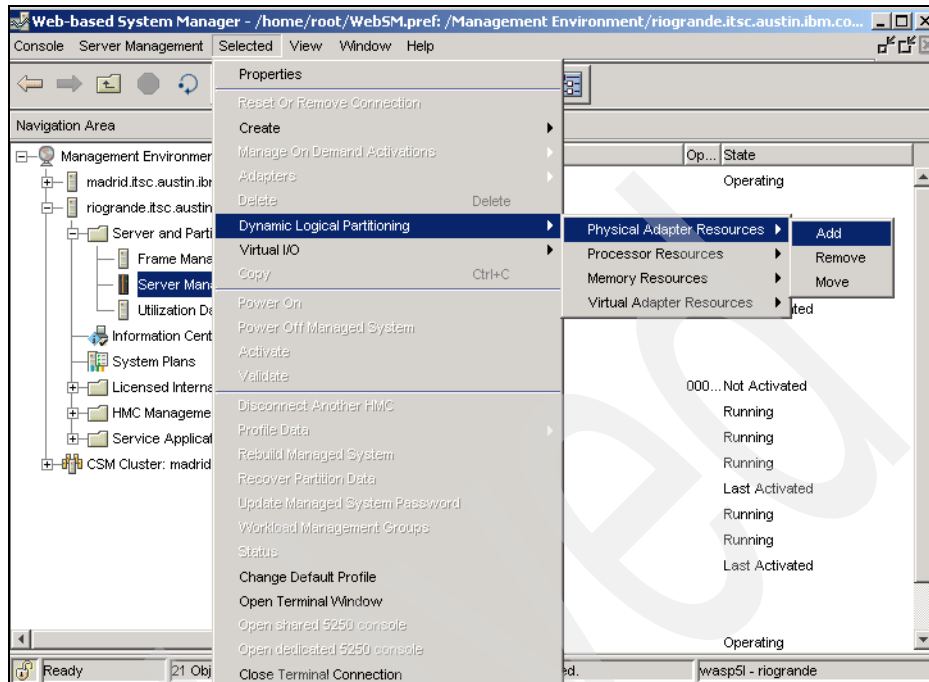


Figure 3-12 HMC Selected - Add process

Figure 3-14 on page 64 shows that the Current Processing units count was 0.50 and the Maximum Processing units count was set to 2.00.

We should have been able to add 1.50 more processing units. However, because only one virtual processor was available, this effort would fail with the error message shown in Figure 3-13.

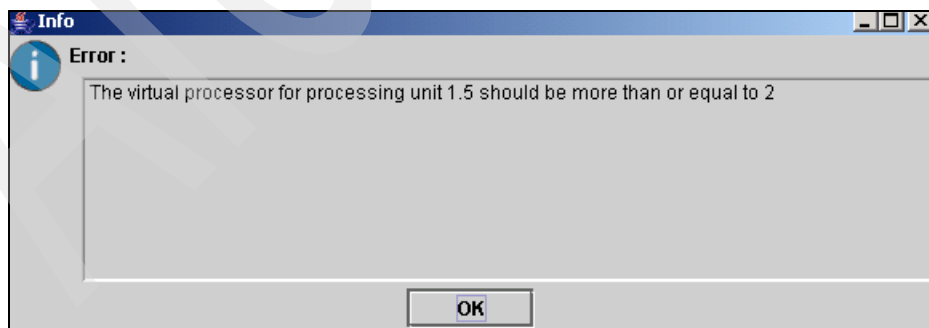


Figure 3-13 Add Processing units - error message

For this reason, we added two virtual processors to the system first, as shown in Figure 3-14.

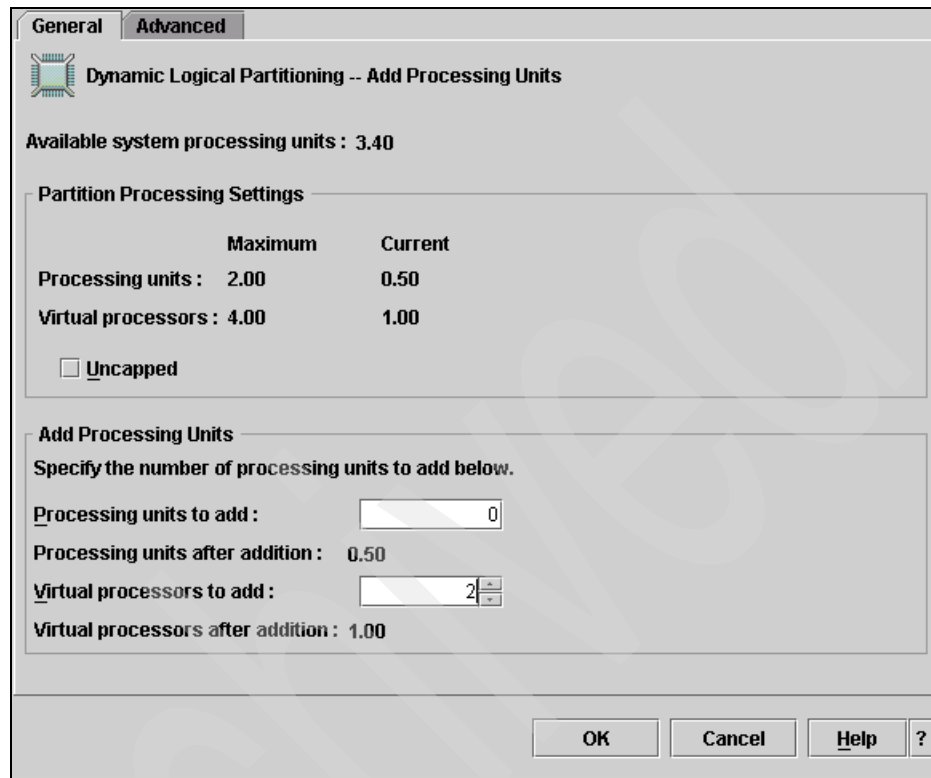


Figure 3-14 HMC GUI to add two processors to a DLPAR

3.1.5 Dynamic LPAR assignments and partition profiles

When you remove, change, or add I/O adapters, memory, CPU, or virtual adapters using dynamic LPAR partition, they are not automatically reflected in the partition's profile defined for the partition. Therefore, whenever you reconfigure, we recommend you update the profile.

A good procedure is first to put the changes into the partition profile, and then make the changes dynamically to prevent them from being lost in case of a deactivation or shutdown. The changes made to the partition in the partition profile are not reflected in the LPAR if you perform a reboot using, for example, the **shutdown -Fr** command; instead, the changes are activated only from a Not Activated state.

If you have multiple workloads that require multiple partition profiles, then instead of cleaning up and renaming the partition profiles, simply activate them and use a naming convention that is meaningful to the scenario (such as WebSphere Application Server _Prod_High or WebSphere Application Server _Prod_Med).

3.1.6 Virtual I/O Server virtualization configuration

The Virtual I/O Server is an appliance that provides virtual storage and shared Ethernet capability to client logical partitions on a POWER5 system. It allows a physical adapter with attached disks on the Virtual I/O Server partition to be shared by one or more partitions, thus enabling clients to consolidate and potentially minimizing the number of physical adapters.

The Virtual I/O Server is the link between the virtual world and the real world. It can be seen as an AIX-based appliance, and it is supported on POWER5 servers only. The Virtual I/O Server runs in a special partition which cannot be used for execution of application code.

It mainly provides two functions:

- ▶ Server component for Virtual SCSI devices (VSCSI target)
- ▶ Support of shared Ethernet adapters for Virtual Ethernet

Using the Virtual I/O Server facilitates the following functions:

- ▶ Sharing physical resources between partitions on the system
- ▶ Creating partitions without requiring additional physical I/O resources
- ▶ Creating more partitions than there are I/O slots or physical devices available with the ability for partitions to have dedicated I/O, virtual I/O, or both
- ▶ Maximizing physical resource use on the system

Installing and customizing the Virtual I/O Server

You install the Virtual I/O Server partition from a special mkxsysb CD that is shipped with the Advanced POWER Virtualization feature.

Creating the VIO Server LPAR

The VIO Server is installed into an LPAR that can be created using the Partition wizard. As shown in Figure 3-11 on page 55, you only need to select VIO Server as the partition environment, and not AIX.

When creating the LPAR that should run the VIO Server, keep the following considerations in mind:

- ▶ When selecting memory values for the Virtual I/O Server LPAR, select the maximum value with care.

If you select a large number, such as 128 GB, you will pin a significant amount of memory. The hypervisor firmware will reserve 1/64 of the value entered as the maximum value in the hypervisor firmware system memory. Therefore, if you select 128 GB, you will reserve 2 GB of memory in the hypervisor system memory for a value you may never need. The hypervisor memory used can be obtained in the IBM System Planning Tool (SPT).

- ▶ When you select the number of CPUs, use a realistic estimate for CPUs.

If you have a two-CPU workload that may expand up to four CPUs, do not enter 12 CPUs. Unless your production workload validates a smaller value, start with an allocation of at least a whole CPU for the Virtual I/O Server if you plan on having high network traffic. In general, network traffic increases CPU utilization. Disk traffic does not, in general, require the same amount of CPU because the I/Os are queued to slower devices.

- ▶ If you want to create a dual Virtual I/O Server scenario with Shared Ethernet Adapter failover, then on the primary Virtual I/O Server select the value 1 in the Trunk priority panel, and use the value 2 on the standby Virtual I/O. Then create the control path between the two Virtual I/O Servers that is used by the Shared Ethernet Adapter failover.
- ▶ Choose a self-explanatory name for the VIO Server and the LPAR. (In our case we added the suffix `vio`, as in `wasp5l_vio`.)

In Example 3-7, we used the command `lshwres` on the HMC command line interface to display the default settings for memory and processors that we used for the VIO Server.

Example 3-7 Initial memory and processor settings for the VIO Server

```
#lshwres -r mem -m p5+-9133-55A-SN10D1FAG --level lpar --filter
"lpar_names=wasp5l_vio"
lpar_name=wasp5l_vio,lpar_id=1,curr_min_mem=128,curr_mem=512,curr_max_m
em=768,pend_min_mem=128,pend_mem=512,pend_max_mem=768,run_min_mem=128,r
un_mem=512
#lshwres -r proc -m p5+-9133-55A-SN10D1FAG --level lpar --filter
"lpar_names=wasp5l_vio"
lpar_name=wasp5l_vio,lpar_id=1,curr_shared_proc_pool_id=0,curr_proc_mod
e=shared,curr_min_proc_units=0.1,curr_proc_units=0.1,curr_max_proc_unit
s=1.0,curr_min_procs=1,curr_procs=1,curr_max_procs=2,curr_sharing_mode=
uncap,curr_uncap_weight=128,pend_shared_proc_pool_id=0,pend_proc_mode=s
hared,pend_min_proc_units=0.1,pend_proc_units=0.1,pend_max_proc_units=1
```

```
.0,pend_min_procs=1,pend_procs=1,pend_max_procs=2,pend_sharing_mode=uncap,pend_uncap_weight=128,run_proc_units=0.1,run_procs=1,run_uncap_weight=128
#
```

The same information can be retrieved using the HMC GUI; see Figure 3-15.

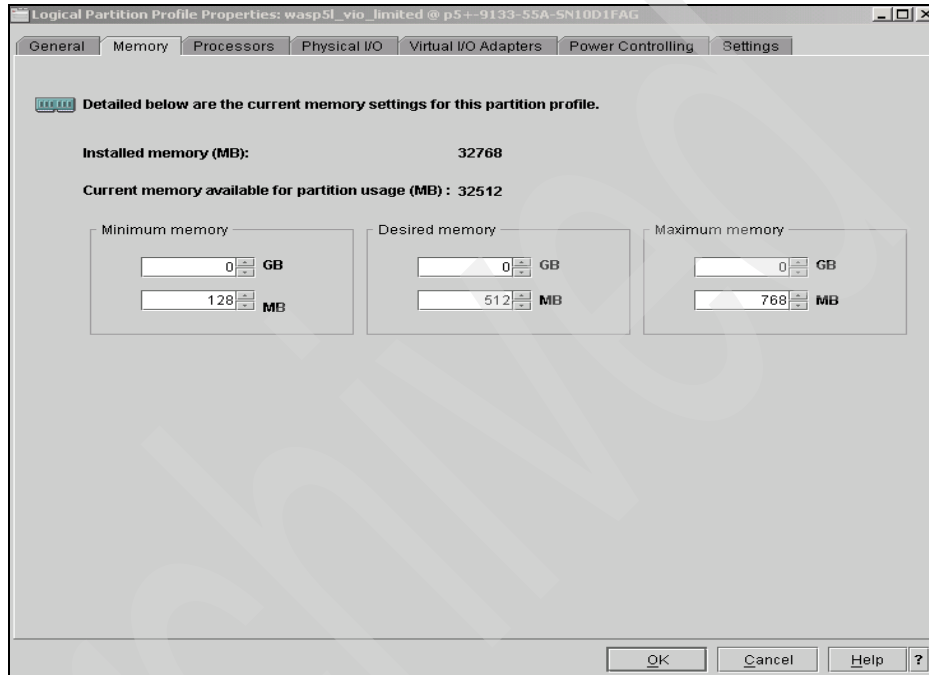


Figure 3-15 Initial memory settings for the VIO Server

When creating a partition, the default values are predefined to only 128 MB, which is insufficient for installing the VIO Server. Therefore, in our case we used 512 MB as the desired value. Set the Minimum memory value to at least equal the desired value, in order to ensure VIO Server operation.

Having a guaranteed 0.1 of a CPU can sustain daily network usage. But by using the uncapped CPU resources, we can allow the VIO Server to grow to 1.0 CPUs, if required, using spare CPU cycles from the CPU pool. The values used during our setup are shown in Figure 3-16 on page 68.

For more detailed information about how to determine the proper amount of memory and CPU for your VIO Server, see Chapter 5 of *IBM System p Advanced POWER Virtualization (PowerVM) Best Practices*, REDP-4194.

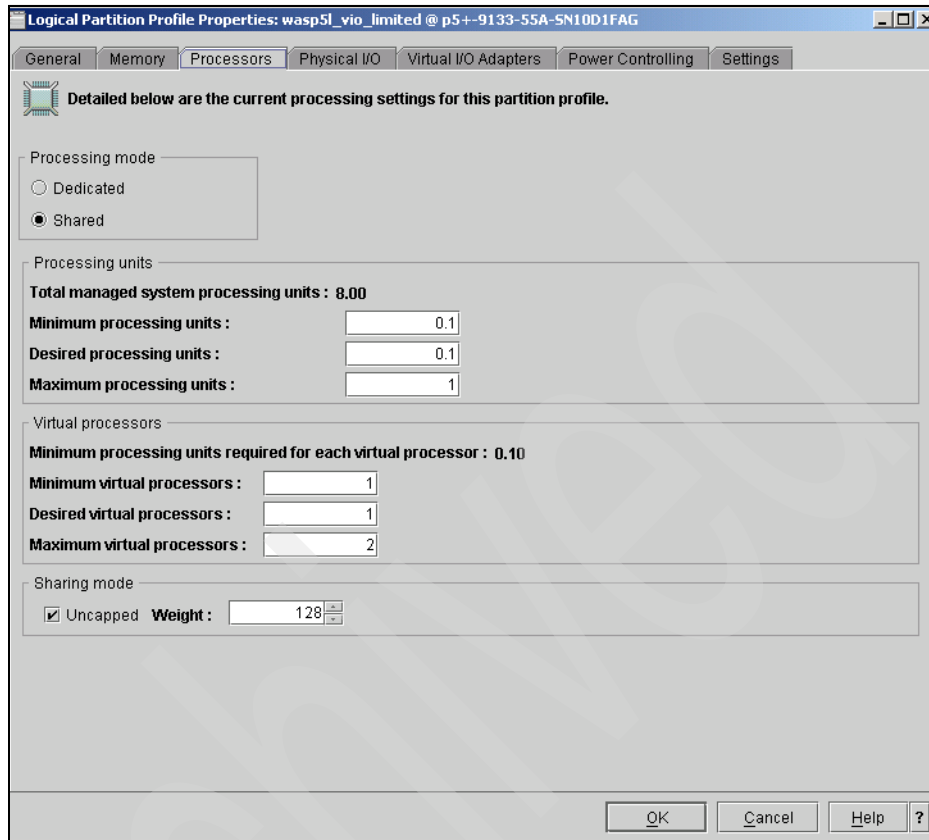


Figure 3-16 VIO initial PROCS values used

The information can be retrieved from the HMC command line login by using the **lshwres** command, as shown in Example 3-8.

Example 3-8 lshwres output for VIO Server processor settings

```
wasp5l@riogrande:~> lshwres -r proc -m p5+-9133-55A-SN10D1FAG --level
lpar --filter "lpar_names=wasp5l_vio"
lpar_name=wasp5l_vio,lpar_id=1,curr_shared_proc_pool_id=0,curr_proc_mod
e=shared,curr_min_proc_units=0.1,curr_proc_units=0.1,curr_max_proc_unit
s=1.0,curr_min_procs=1,curr_procs=1,curr_max_procs=2,curr_sharing_mode=
uncap,curr_uncap_weight=128,pend_shared_proc_pool_id=0,pend_proc_mode=s
hared,pend_min_proc_units=0.1,pend_proc_units=0.1,pend_max_proc_units=1
.0,pend_min_procs=1,pend_procs=1,pend_max_procs=2,pend_sharing_mode=unc
ap,pend_uncap_weight=128,run_proc_units=0.1,run_procs=1,run_uncap_weigh
t=128
```


At this point, we were able to install the VIO Server.

Installing the VIO Server

You can install the Virtual I/O Server by using one of the following methods:

- ▶ Media (assign the DVD-ROM drive to the partition and boot from the media)
- ▶ HMC (insert the media in the DVD-ROM drive on the HMC and use the **installios** command)
- ▶ NIM (insert the media in the DVD-ROM drive on the NIM Master and use the **installios** command)

The installation itself does not require manual or further input.

Note that the Virtual I/O Server is not accessible as a standard partition. Administrative access to the Virtual I/O Server partition is only possible as the user `padmin`, not as the root user. After logging in using a virtual terminal, the user `padmin` gets a restricted shell, which is not escapable, called the command line interface.

Before you can start any further configuration, you need to accept the license agreement by entering the **license -accept** command on the Virtual I/O Server restricted shell.

Setting up the VIO Server

Normally, setting up Virtual I/O is done alongside creating partitions and making virtual devices available to the partitions. So the process involves actions that need to be performed on the HMC, and actions to be performed on the VIO Server side. For this reason, planning and designing your layout up front is crucial to implementing a successful configuration.

The following steps have to be taken using the HMC interface:

- ▶ Creating a virtual Ethernet adapter for the Virtual I/O Server
- ▶ Creating virtual SCSI server adapters
- ▶ Creating client partitions
- ▶ Creating virtual Ethernet adapters for client partitions
- ▶ Creating the virtual SCSI adapter for client partitions

The following steps have to be taken using the VIO Server command line interface:

- ▶ Defining volume groups and logical volumes
- ▶ Creating a Shared Ethernet Adapter
- ▶ Installing client partition AIX 5L

For detailed explanations about each of these steps, refer to the IBM Redbooks publication *PowerVM Virtualization on IBM System p: Introduction and Configuration Fourth Edition*, SG24-7940.

Setting up virtual devices

This section explains how to set up virtual devices, within the VIO Server, that need to be connected to the Partition Profile.

Naming conventions

In addition to using a tracking tool such as a spreadsheet, a useful naming convention is key to managing this information. One strategy for reducing the amount of data that must be tracked is to make device names and slots match on the virtual I/O client and server wherever possible.

This convention could include corresponding volume group, logical volume, and virtual target device names. Integrating the virtual I/O client host name into the virtual target device name can simplify tracking on the server.

Device slot numbers

After naming conventions have been established, slot numbering conventions should also be established for the virtual I/O adapters.

Slot numbers are shared between virtual storage and virtual network devices. In complex systems there will tend to be far more storage devices than network devices because each virtual SCSI device can only communicate with one server or client. We recommend that you reserve the slot numbers through 20 for network devices on all LPARs, to keep the network and storage devices grouped together.

Management can be simplified by keeping slot numbers consistent between the virtual I/O client and server. However, when partitions are moved from one server to another, this may not be possible.

In environments with only one VIO Server, storage adapters should be added incrementally starting with slot 21 and higher. When clients are attached to two VIO Servers, the adapter slot numbers should be alternated from one VIO Server to the other. The first VIO Server should use odd numbered slots starting at 21, and the second VIO Server should use even numbered slots starting at 22. In a two-server scenario, slots should be allocated in pairs, with each client using two adjacent slots such as 21 and 22, or 33 and 34.

The maximum number of virtual adapter slots per LPAR should be increased above the default value of ten when you create an LPAR. The appropriate number for your environment depends on the number of LPARs and adapters expected on each system. Each unused virtual adapter slot consumes a small

amount of memory, so the allocation should be balanced. Use the System Planning Tool, which is available from the following URL, to plan memory requirements for your system configuration:

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

Because VSCSI connections operate at memory speed, there is generally no performance gain from adding multiple adapters between a VIO Server and client. Each adapter pair can handle large numbers of target devices, or disks. In our case, we used the assignment definition listed in Table 3-5.

Table 3-5 Server and Client Slot ID definitions

Partition name	Server Slot ID	Client Slot ID
de_lpar	31	32
br_lpar	41	42
uk_lpar	51	52
us_lpar	61	62
pt_lpar	71	72
pl_lpar	81	82

We used a value of 100 as shown in Figure 3-17 on page 72 for the VIO Server Profile using HMC GUI.

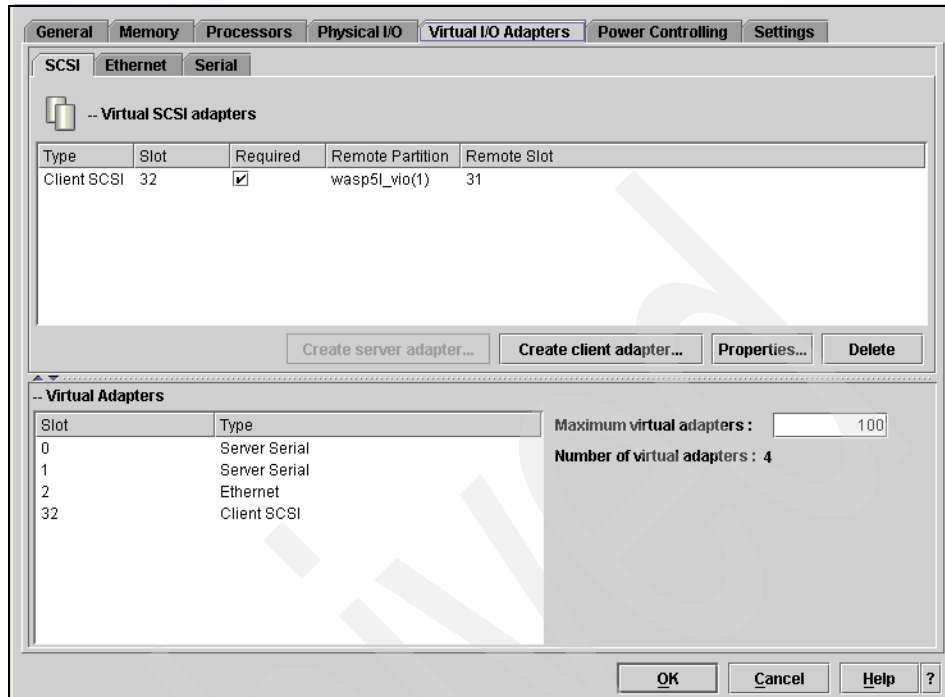


Figure 3-17 Maximum virtual adapters setting

Figure 3-18 on page 74 shows the final VIO Partition Properties after the physical and virtual disks are all mapped together.

Managing and exporting physical storage on the VIO Server

The Virtual I/O Server presents disk storage to virtual I/O clients (VIOCs) as virtual SCSI disks. These virtual disks must be mapped to physical storage by the VIO Server. There are three different ways to perform this mapping, each with its own advantages:

- ▶ Physical volumes
- ▶ Logical volumes
- ▶ Storage pools

The general rule for selecting between these options is that disk devices being accessed through a SAN should be exported as physical volumes, with storage allocation managed in the SAN. Internal and SCSI attached disk devices should be exported with either logical volumes or storage pools so that storage can be located within the server.

A single volume group should not contain logical volumes used by virtual I/O clients and logical volumes used by the VIO Server operating system. Keep VIO

Server file systems within the rootvg, and use other volume groups to host logical volumes for virtual I/O expanding the size of virtual storage devices.

When exporting logical volumes to clients, the mapping of individual logical volumes to virtual I/O clients is maintained on the VIO Server. The additional level of abstraction provided by the logical volume manager makes it important to track the relationship between physical disk devices and virtual I/O clients.

One strategy for reducing the amount of data that must be tracked is to make device names and slots match on the virtual I/O client and server wherever possible. This could include corresponding volume group, logical volume, and virtual target device names. Integrating the virtual I/O client host name into the virtual target device name can simplify tracking on the server.

After naming conventions have been established, slot numbering conventions should also be established for the virtual I/O adapters. Slot numbers are shared between virtual storage and virtual network devices.

When planning for the number of virtual I/O slots on your LPAR, the maximum number of virtual adapter slots available on a partition is set by the partition's profile. To change this profile, shut down the LPAR. We recommend leaving plenty of room for expansion when setting the maximum number of slots so that new virtual I/O clients can be added without shutting down the LPAR or VIO Server partition.

Despite the best intentions in record keeping, it sometimes becomes necessary to manually trace a client virtual disk back to the physical hardware. The IBM Systems Hardware Information Center contains a guide to tracing virtual disks. At the following URL, search for "Mapping virtual disks to physical disks". Be sure to include the quotes when you enter the search string.

http://publib.boulder.ibm.com/infocenter/systems/index.jsp?topic=/iphb1/iphb1_vios_managing_mapping.htm

Depending on which method you choose, you may need to track the following information:

- ▶ Virtual I/O Server
 - Server host name
 - Physical disk location
 - Physical adapter device name
 - Physical hdisk device name
 - Volume group or storage pool name1
 - Logical volume or storage pool backing device name1
 - VSCSI adapter slot
 - VSCSI adapter device name
 - Virtual target device

- Virtual I/O Client
 - Client host name
 - VSCSI adapter slot
 - VSCSI adapter device name
 - Virtual hdisk device name

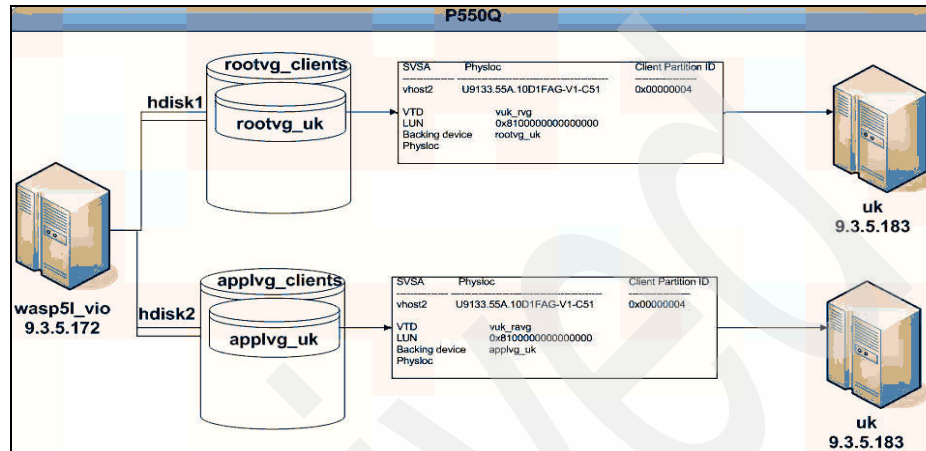


Figure 3-18 Exported VIO disk mapping

Figure 3-18 illustrates mapping between physical storage, VIO Server and client. On the VIO Server, hdisk2 has been assigned to the volume group applvg_clients. On this volume group, we created a logical volume applvg_uk. This logical volume is connected as a virtual disk to the LPAR uk_lpar.

When we logged on to the partition uk.itsc.austin.ibm.com, AIX mapped the virtual device as hdisk1, which was assigned to the volume group appl_vg.

Example 3-9 displays the final result of using the command **lspv** on node uk executed through a dsh session of the Management Server madrid.itsc.austin.ibm.com.

Example 3-9 lspv on system uk with the second virtual hdisk1

```
[2:root@MADRID:]/home/root # dsh -n uk "lspv"
uk.itsc.austin.ibm.com: hdisk0          00c4790e281766f5
rootvg                  active
uk.itsc.austin.ibm.com: hdisk1          00c4790ecd59fa72
appl_vg                 active
```

Creating the boot volume for the VIO SCSI client partitions

In order to assign a boot volume to a partition that uses Virtual SCSI, you first need to create the volumes (physical and logical) on the VIO Server, and then create the virtual device for the client partition. In this section we demonstrate these tasks, using the definition shown in Figure 3-18 on page 74.

First, we executed the `lspv` command on the VIO Server. This showed that the devices `hdisk1` and `hdisk2` were available but unconfigured; see Example 3-10.

Example 3-10 lspv output default vios output

\$ lspv		
NAME	PVID	VG
STATUS		
hdisk0	000b7bac80a00769	rootvg
active		
hdisk1	none	None
hdisk2	none	None
\$		

In our case, we want `hdisk1` to hold all logical volumes and act as the boot volume, and `hdisk2` to become our shared volume group. We created the volume group named `rootvg_clients` on `hdisk1`, as shown in Example 3-11.

Example 3-11 Using the mkvg command to create rootvg_clients volume group

\$ mkvg -f -vg rootvg_clients hdisk1			
rootvg_clients			
\$ lspv			
NAME	PVID	VG	STATUS
hdisk0	000b7bac80a00769	rootvg	active
hdisk1	000b7bacc2fc3c83	rootvg_clients	active
hdisk2	none	None	
\$			

The newly created volume group can now be used to hold all logical volumes. Next, we created the logical volume `rootvg_uk`, which will become the root volume group on partition `uk`; see Example 3-12.

Example 3-12 Using the mklv command to create the rootvg_uk

\$ mklv -lv rootvg_uk rootvg_clients 20G	
rootvg_uk	
\$	
\$ lsvg -lv rootvg_clients	
rootvg_clients:	

LV NAME	TYPE	LPs	PPs	PVs	LV STATE	MOUNT
POINT						
rootvg_uk	jfs	160	160	1	closed/syncd	N/A
\$						

We created the virtual device and vuk_rvg using the **mkvdev** command; see Example 3-13.

Example 3-13 Using the mkvdev command to create virtual device vuk_rvg

```
$mkvdev -vdev rootvg_uk -vadapter vhost0 -dev vuk_rvg
$vuk_rvg Available
```

Example 3-14 and Example 3-15 on page 77 illustrate the relationship between the Virtual SCSI disk on the VIO Server and the client.

On the VIO Server we used the command **lsmmap** to display the mapping between the physical, logical, and virtual devices; see Example 3-14.

Example 3-14 Virtual device map for node uk on VIO

```
$lsmmap -vadapter vhost2
SVSA          Physloc Client Partition ID
-----
vhost2        U9133.55A.10D1FAG-V1-C51          0x00000004

VTD           vuk_rvg
LUN           0x8100000000000000
Backing device rootvg_uk
Physloc
#
```

Notice that the logical volume rootvg_uk device is mapped as vuk_rvg through the server virtual adapter (SVSA) vhost2, which has the physical location code U9133.55A.10D1FAG-V1-C51. (This is also shown in Figure 3-21 on page 81.)

Next, we created a listing of the vital product data by using the commands **lscfg** and **lspv** on node uk.itsc.austin.ibm.com. As you can see in Example 3-15 on page 77, hdisk0 is mapped to the physical location U9133.55A.10D1FAG-V4-C52.

Also note that the locations U9133.55A.10D1FAG-V1-C51 and U9133.55A.10D1FAG-V4-C52 reflect to the virtual devices slot numbers as defined in “Device slot numbers” on page 70 and Table 3-5 on page 71.

Example 3-15 Virtual device configuration on node uk

```
#dsh -n uk "lscfg -vl hdisk0"
uk.itsc.austin.ibm.com:   hdisk0
U9133.55A.10D1FAG-V4-C52-T1-L810000000000 Virtual SCSI Disk Drive
#dsh -n uk lspv
uk.itsc.austin.ibm.com: hdisk0          000b7bacc8a01fdf
rootvg                  active
```

At this point, we finalized the preparation for our partitions before installing the base operating system.

Networking on the VIO Server

The VIO Server can be accessed from the HMC by using a secure private HMC-to-Service Processor network to open a console session. This makes a dedicated network address on the VIO Server for administration optional. However, if the VIO Server does not appear on any network at all, dynamic resource allocation will not be enabled for the VIO Server because there is no way to connect to it.

One key to managing a virtual environment is keeping track of which virtual objects correspond to which physical objects. In the network area, this can involve physical and virtual network adapters and VLANs that span hosts and switches.

Depending on whether you choose to use 802.1Q tagged VLANs, you may need to track the following information:

- ▶ Virtual I/O Server
 - Server host name
 - Physical adapter device name
 - Switch port
 - SEA adapter device name
 - Virtual adapter device name
 - Virtual adapter slot number
 - Port virtual LAN ID (in tagged and untagged usages)
 - Additional virtual LAN IDs
- ▶ Virtual I/O client
 - Client host name
 - Virtual adapter device name
 - Virtual adapter slot number
 - Port Virtual LAN ID (in tagged and untagged usages)
 - Additional virtual LAN IDs

Slot numbers are shared between virtual storage and virtual network devices. In complex systems there will tend to be far more storage devices than network devices, because each virtual SCSI device can only communicate with one server or client. We recommend that you reserve slot numbers through 20 for network devices on all LPARs in order to keep the network devices grouped together; refer to “Device slot numbers” on page 70 for details.

Setting up a Shared Ethernet Adapter (SEA) on the VIO Server

Virtual devices must be created on the Partition profile level through the HMC. We changed the current Partition profile using default settings, as shown in Figure 3-19.

Adapter settings

Slot : * 2

Virtual LAN ID : * 1

☒ Access external network

Trunk priority : 1

☐ IEEE 802.1Q compatible adapter

Maximum number of virtual LAN IDs: 20

Number of VLAN IDs : 1

Additional virtual LAN ID to add: 1

Additional VLAN IDs :

Add Remove

System VLANs

Slot	Virtual Ada...	Bridge Adapter
------	----------------	----------------

OK Cancel Help ?

Figure 3-19 Virtual Ethernet configuration for VIO Server using the HMC

Because the change has been applied to the current partition profile, a reboot of the VIO Server is required.

In Example 3-16 on page 79 you can see that the virtual adapter ent2 is mapped to the physical adapter ent0. Also note that the second physical adapter ent1 is not in use at this point. You need to be in the from the oem_setup_env session on the VIO Server to be able to execute the command.

Example 3-16 VIO initial virtual Ethernet Adapter state

```
# lscfg -l ent\*
ent2          U9133.55A.10D1FAG-V1-C2-T1  Virtual I/O Ethernet
Adapter (1-lan)
ent0          U787B.001.DNWB206-P1-T9      2-Port 10/100/1000
Base-TX PCI-X Adapter (14108902)
ent1          U787B.001.DNWB206-P1-T10   2-Port 10/100/1000
Base-TX PCI-X Adapter (14108902)
```

Next, we created the SEA, using the **mkvdev -sea** option for the virtual adapter **ent2**. We used the command **lsdev** to verify that the adapter has been created; see Example 3-17.

Example 3-17 Virtual Ethernet ent2 status

```
$mkvdev -sea ent0 -vadapter ent2 -default ent2 -defaultid 2
$lsdev -dev ent2
name          status
description
ent2          Available  Virtual I/O Ethernet Adapter (1-lan)
```

We created the Ethernet TCPIP Interface, which enables TCP/IP traffic. The Virtual Interface Name generated by the command **mkvdev** must be used. In our case, it was interface **en3**; see Example 3-18.

Example 3-18 Running mktcip on VIO Virtual Ethernet

```
$mktcpip -hostname wasp5l_vio -inetaddr 9.3.5.170 -interface en3
-netmask 255.255.255.0 -gateway 9.3.5.41
$lsdev -dev en3
name          status
description
en3           Available  Standard Ethernet Network Interface
```

Now we were ready to use the Shared Virtual Ethernet Interface, which could be used by other LPARs without the need to assign more physical Ethernet adapters to each of them. The Virtual Adapter must be created in the Partition Profile, as shown in Figure 3-20 on page 80.

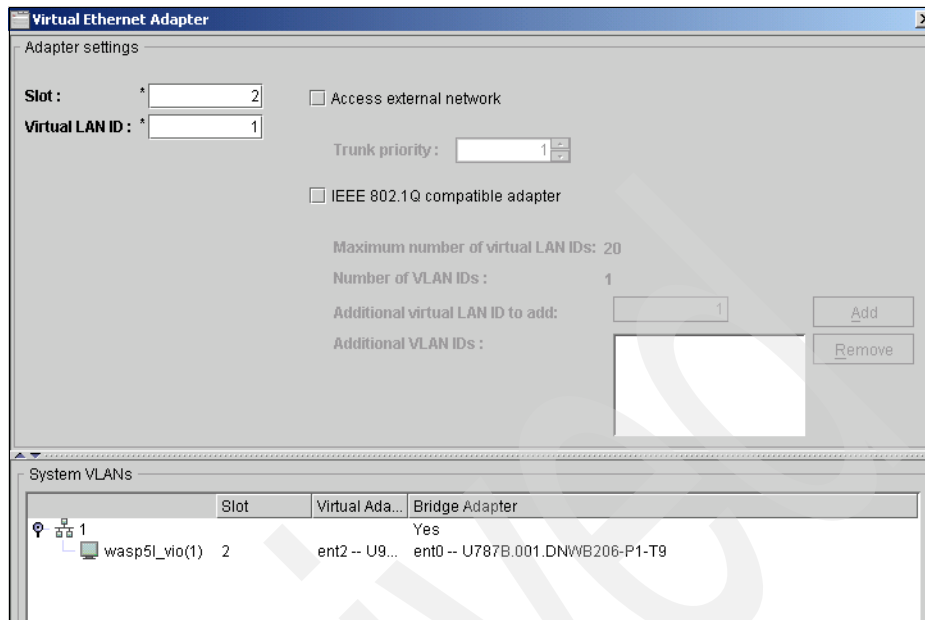


Figure 3-20 Virtual Ethernet configuration for client partition using the HMC

Mapping the virtual devices into the Client Partition Profile

The virtual SCSI Disk, as well as the Virtual Ethernet Adapter, must be mapped into the Client Partition Profile on the HMC after the setup has been finalized with the VIO Server. The final result can be checked on the VIO Server login by using the `lsdev` command, as shown in Example 3-19.

Example 3-19 `lsdev -virtual` on the VIO Server

```
$ lsdev -virtual
name          status
description
ent2          Available Virtual I/O Ethernet Adapter (1-lan)
vhost0        Available Virtual SCSI Server Adapter
vhost1        Available Virtual SCSI Server Adapter
vhost2        Available Virtual SCSI Server Adapter
vhost3        Available Virtual SCSI Server Adapter
vhost4        Available Virtual SCSI Server Adapter
vhost5        Available Virtual SCSI Server Adapter
vsa0          Available LPAR Virtual Serial Adapter
vbr_rvg       Available Virtual Target Device - Logical Volume
vde_rvg       Available Virtual Target Device - Logical Volume
vpl_rvg       Available Virtual Target Device - Logical Volume
vpt_rvg       Available Virtual Target Device - Logical Volume
```

vuk_rvg	Available	Virtual Target Device - Logical Volume
vus_rvg	Available	Virtual Target Device - Logical Volume
ent3	Available	Shared Ethernet Adapter

We used the HMC GUI Interface to create the link between the VIO virtual devices and the client partitions. As discussed in “Device slot numbers” on page 70, this illustrates the importance of using slot number assignments. Figure 3-21 shows the virtual SCSI devices mapped to clients.

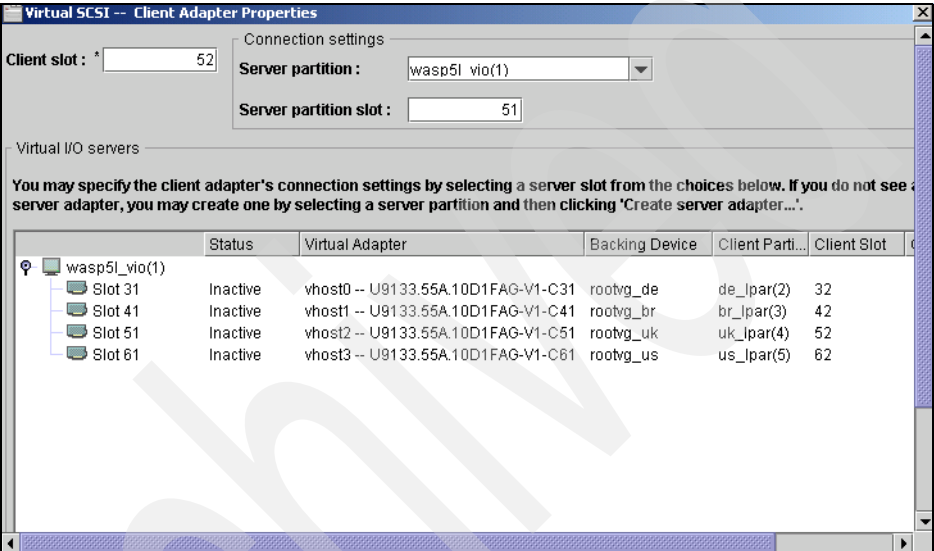


Figure 3-21 Virtual SCSI Disk configuration for client partition using the HMC

The shared virtual Ethernet Adapter is mapped to each client partition profile as shown in Figure 3-22 on page 82

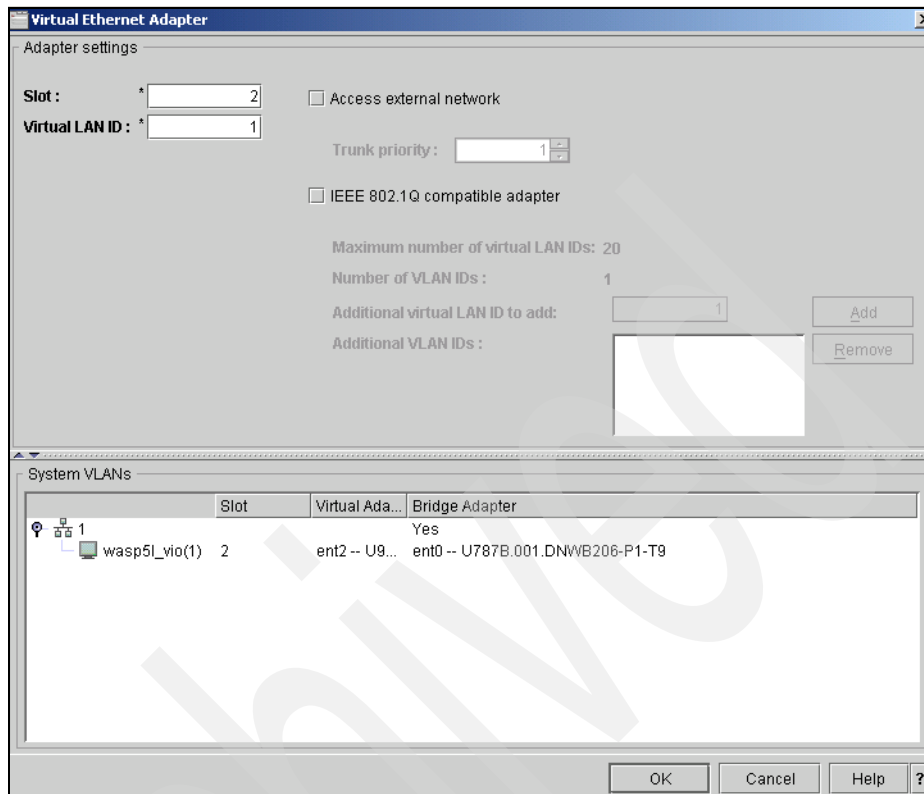


Figure 3-22 Virtual Ethernet configuration for client partition using the HMC

Figure 3-23 on page 83 shows the Property relationship in the Partition Profile for the VIO Server between the Server Adapter and the Client Adapter.

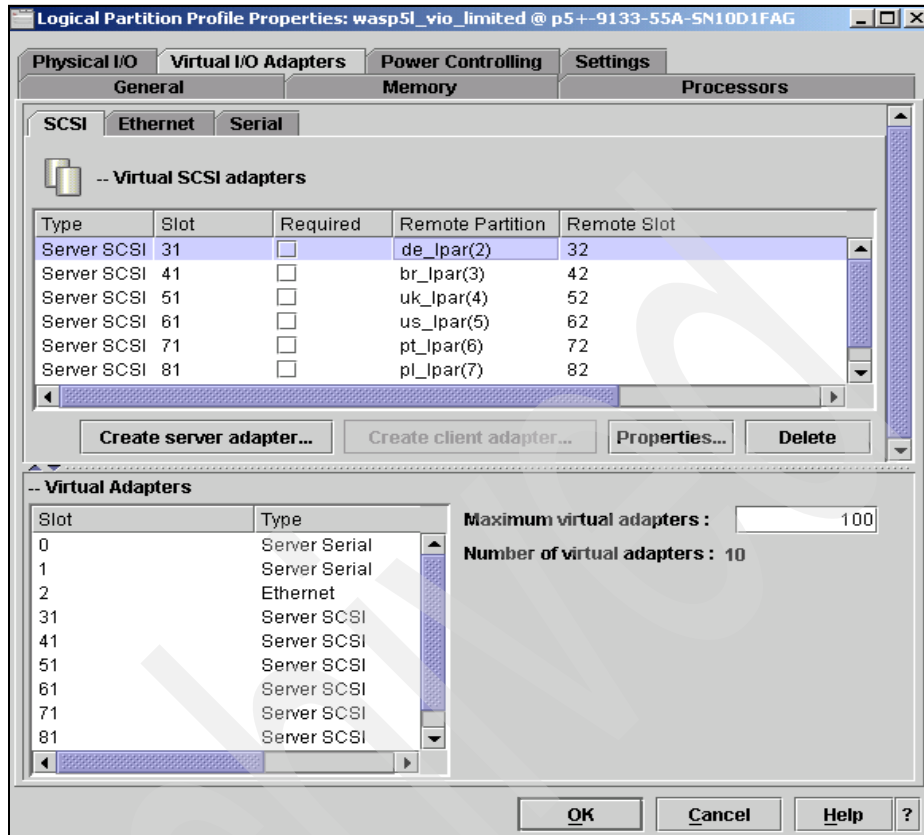


Figure 3-23 VSCSI setup for DLPAR UK in VIO Server

Backup and availability of the VIO Server

The Virtual I/O Server partition is a critical partition; it always needs to be available for client partitions. If you need to reboot or recover, all partitions deployed through VIO Server will be unavailable. Therefore, backup and availability considerations are important.

Backup

Back up the Virtual I/O Server regularly using the **backupios** command to create an installable image of the root volume group onto either a bootable tape or a multivolume CD or DVD. The creation of an installable NIM image on a file system is provided as well.

Availability

In addition, you can use the **mirrorios** command to mirror all logical volumes on the rootvg volume group of the VIO Server on a physical drive, which must already be member of the rootvg volume group.

Here are two main reasons to implement the mirroring of rootvg:

- ▶ To provide continuous operation of the VIO Server operating system if a disk that is part of the operating system fails, and an active mirrored copy of the operating system exists on another disk.
- ▶ To be able to boot more than one disk of the rootvg if another boot disk fails. In some cases, the ability to boot from an alternate disk may require some user interaction.

Example 3-20 illustrates how to add a hard disk named hdisk1 on the VIO Server into the rootvg, create the mirror, and show the new bootlist with both boot devices.

Example 3-20 mirrorios command execution

```
$bootlist -mode normal
hdisk0 blv=hd5
$extendvg rootvg hdisk
$lsnv
NAME                PVID                VG
STATUS
hdisk0               00cba77f96c080df    rootvg
active
hdisk1               00cba77f0636e164    rootvg
active

$mirrorios -f hdisk
$bootlist -mode normal
hdisk1 blv=hd5
hdisk0 blv=hd5
```

3.2 Provisioning

The automation of typical data center tasks, such as software installation or network configuration, is crucial for faster and easier data center management. Therefore, the provisioning process involves many steps. However, after these steps are completed in your pSeries environment, you can provision and manage your environment more effectively than before.

Task automation in IBM takes the form of workflows that automate a manual process, and it includes the IT expertise needed to perform these tasks. The dynamic execution of workflows is the core of systems provisioning. Systems provisioning is one of the IBM Virtualization Engine™ systems services.

Systems provisioning introduces the concept of pools of IBM eServer™ resources that are shared between different workloads. The resources are dynamically assigned to workloads that need capacity. Capacity can be given back to the pool if the workload no longer needs it. Little or no human interaction is required. Typical activities include installation of operating systems, booting machines over a network, device configuration, and configurations for networks, such as virtual LANs (VLANs).

3.2.1 Methods

Provisioning occurs at both the hardware level and the software level, as explained here.

Hardware-based methods:

- ▶ Hardware Management Console (HMC)
- ▶ Dynamic Logical Partition
- ▶ Micro Partitioning
- ▶ Virtual I/O (VIO)
- ▶ Capacity on Demand (CoD)

Software-based methods:

- ▶ Cluster System Manager (CSM)
- ▶ Network Installation Manager (NIM)
- ▶ Reliable Scalable Cluster Technology (RSCT)
- ▶ Workload Manager (WLM)
- ▶ Partition Load Manager (PLM)

3.2.2 Provisioning at the operating system level

There are various methods available to install the operating system. We recommend that you install the operating system automatically by using the NIM Master server, because it enables you to maintain a single installation and update source, and provides the flexibility to choose between different installation methods. Keep in mind that your network environment must be defined and working correctly before installing AIX 5L automatically.

In addition, we chose to use Cluster System Manager (CSM) to manage our WebSphere LPARs because it provides a single access and control point from a

systems management perspective, as well as the flexibility to grow with an enterprise.

Cluster System Manager

Cluster System Manager (CSM) provides a distributed systems management solution for maintaining clusters of AIX and Linux nodes. CSM has a client server architecture. It utilizes the Resource Management and Control (RMC) part of RSCT to manage pSeries servers and LPARs. The functionality of CSM includes:

- ▶ Installing and updating software on nodes
- ▶ Distributed command execution
- ▶ Hardware control
- ▶ File synchronization across managed nodes
- ▶ Monitoring resources in the cluster

CSM uses NIM for software installation and update. It provides commands to set up the NIM environment and create machine and other resources. However, it does not prevent independent use of NIM. After CSM setup and cluster configuration, you can still manage the software installation and maintenance of machines that are not part of the CSM cluster.

We suggest that you create one VLAN for the CSM management server, managed devices, and hardware control points, and a separate VLAN for the CSM management server and cluster nodes.

Here is the recommended configuration for system management in CSM:

- ▶ **Management VLAN**

Hardware control commands such as `rpower` and `rconsole` are run on the management server, and they communicate to nodes through the management VLAN. The management VLAN connects the management server to the cluster hardware through an Ethernet connection.

For optimal security, the management VLAN must be restricted to hardware control points, remote console servers, the management server, and root users. Routing between the management VLAN and cluster or public VLANs could compromise security on the management VLAN.

- ▶ **Cluster VLAN**

The cluster VLAN connects nodes to each other and to the management server through an Ethernet connection. Installation and CSM administration tasks such as running the `dsh` command are done on the cluster VLAN. Host names and attribute values for nodes on the cluster VLAN are stored in the CSM database.

► Public VLAN

The public VLAN connects the cluster nodes and management server to the site network. Applications are accessed and run on cluster nodes over the public VLAN. The public VLAN can be connected to nodes through a second Ethernet adapter in each node, or by routing to each node through the Ethernet switch.

Note that using a physically separate CSM management server is safer than using an LPAR management server that is part of a Central Electronics Complex (CEC) that can fail during a hardware or power failure.

For more detailed information about CSM, refer to IBM Redbooks publication *Cluster Systems Management Cookbook for pSeries*, SG24-6859.

Creating a CSM cluster

Use the following basic steps to create a CSM cluster:

1. Set up the management server.
2. Set up one or more install servers (optional).
3. Define the nodes in the cluster.
4. Define non-node devices to the cluster (optional).
5. Install the nodes of the cluster (optional).
6. Add the nodes to the cluster. (You can add AIX, Linux, or both AIX and Linux nodes.)

All tasks can be performed from the WebSM client. Figure 3-24 on page 88 displays the Web-based System Manager Master Server Overview; all the Task items listed on the left side can be performed on the CSM Master.

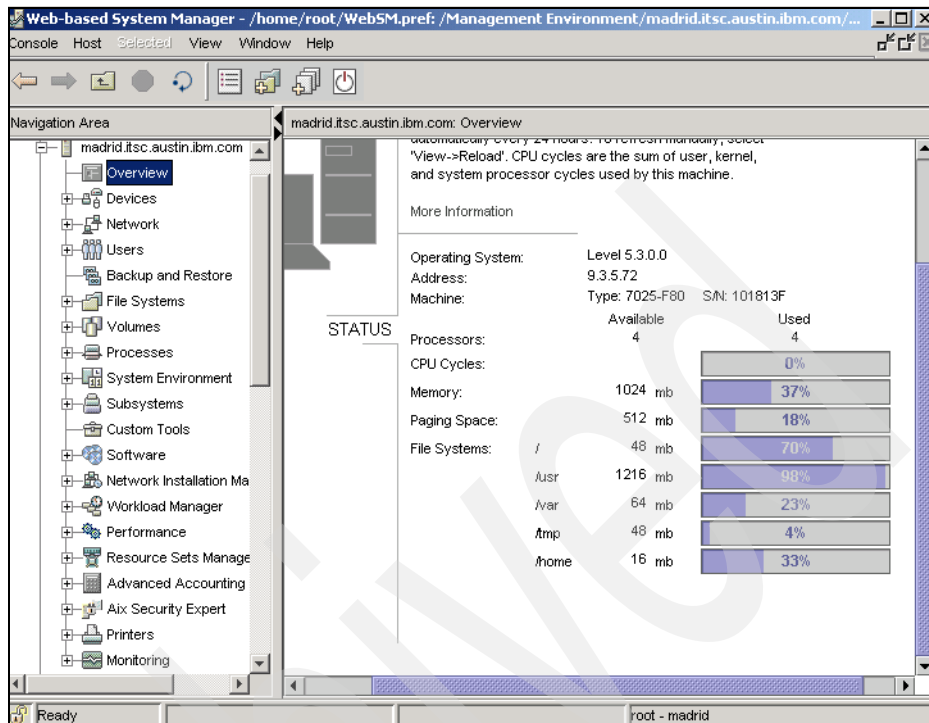


Figure 3-24 WebSM Master Server Overview - tasks listing

Adding HMC managed partitions as nodes to a cluster

To add HMC controlled p5 partitions, then in addition to Step 3 in “Creating a CSM cluster” on page 87, define the nodes in the cluster as follows:

1. To enable CSM remote console function for HMC-attached pSeries nodes, use the command **systemid** to store the user ID and password for remote hardware access and execution. You can verify that the CSM management server can access the HMC by executing the command **rpower -a query**.
2. Create a CSM node definition file called **nodedef** file and define the CSM node.

The creation of the **nodedef** file enables you to define the HMC as the remote console configuration for this partition using **ConsoleMethod=hmc**. We created a file named **/exports/systemfiles/br.node.csm** with the following contents:

```
br:
ConsoleMethod=hmc
ConsoleServerName=riogrande
```

```
HWControlNodeId=br_lparprod
```

```
HWControlPoint=riogrande
```

```
LParID=003
```

```
PowerMethod=hmc
```

```
InstallAdapterDuplex=auto
```

```
InstallAdapterSpeed=auto
```

At this point, partition br_lparprod can be defined as node br into the CSM database by using the command **definenode**:

```
definenode -f /exports/systemfiles/br.node.csm
```

To check the power control from the CSM Management Server, use the **rpower** command for the new created node. In our case, it showed that the power status was off, as expected:

```
#rpower -n pt query
```

```
pt.itsc.austin.ibm.com off
```

3. Acquire Network Adapter information from the node.

In our case, because we want to use NIM to install our partitions, we needed to know the Ethernet adapter information. You can obtain this information either manually by using the SMS Menu, or by executing the CSM command **getadapters**. You can use the command to collect the data for all nodes by using the **-a** option, or for single nodes only by using the **-n** option.

The following example shows the command used to collect the data for all nodes and write them into a file:

```
getadapters -a -z /exports/systemfiles/p550q_lpar_adapters
```

The node stanzas in this file can be adapted for your needs. In our case, the final content in the /exports/systemfiles/p550q_lpar_adapters file appeared as follows:

```
###CSM_ADAPTERS_STANZA_FILE###--do not remove this line
```

```
#---Stanza Summary-----
```

```
#   Date: Tue Sep 19 17:12:11 CDT 2006
```

```
#   Stanzas Added: 3
```

```
#---End Of Summary-----
```

```
de.itsc.austin.ibm.com:
```

```
    MAC_address=922430002002
```

```
    adapter_type=ent
```

```
cable_type=N/A
install_gateway=9.3.5.41
location=U9133.55A.10D1FAG-V2-C2-T1
machine_type=install
netaddr=9.3.5.187
interface_type=en
subnet_mask=255.255.255.0
adapter_duplex=auto
adapter_speed=auto
```

To view the entire file we used, refer to “CSM adapter definition file: p550q_lpar_adapters” on page 471.

4. Update the CSM database and write the node definitions.

Perform this task by using the command **getadapters** with the **-w** option, as shown:

```
getadapters -w -f /exports/systemfiles/p550q_lpar_adapters
```

The command **lsnodes** can be used to verify the entries in the CSM database. For example, to list the status of all nodes use the **lsnode -p** command:

```
# lsnode -p
br: 1 (alive)
brazos: 1 (alive)
de: 1 (alive)
```

5. Update the NIM database with the CSM node information by using commands **csm2nimnodes** and **csmsetupnim**, as explained here:

► **csm2nimnodes**

This command creates or updates NIM machine definitions corresponding to CSM for AIX node definitions. The command can be run from an AIX or Linux management server. It uses CSM database information and command line input to run the appropriate NIM commands. The command creates NIM machine definitions corresponding to CSM cluster nodes; it does *not* provide all of the options available with standard NIM commands.

► **csmsetupnim**

This command sets up CSM customization scripts to automatically install and configure CSM when AIX is installed on a node. This command must be run *after* the NIM client definitions have been created. The command assumes that the CSM node names match the NIM client names.

Provisionnode script for advanced installations

The sample script provisionnode provides an example of how to provision a node based on a predefined node “profile”. The provisionnode sample script and its README can be found at /opt/csm/samples/install. You can customize the script for your own needs.

Script provisionnode runs on a CSM management server to automatically detect and add nodes into a cluster. For details, refer to *CSM for AIX 5L and Linux V1.5 Planning and Installation Guide*, which is available at the following address:

<http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/topic/com.ibm.cluster.csm.doc/csm15/am7i112019.html?>

CSM node reachability, power status, and network interface status

You can check cluster status by using the command **csmstat**, as shown in Example 3-21. Notice that all CSM cluster nodes have the status on.

Example 3-21 Output of csmstat command for entire cluster

#csmstat			

Hostname	HWControlPoint	Status	PowerStatus
Network-Interfaces			

br.itsc.austin.i~	riogrande	on	on
en0-Online			
brazos.itsc.aust~	riogrande	on	on
en0-Online			
de.itsc.austin.i~	riogrande	on	on
en0-Online			
guadalupe.itsc.a~	riogrande	on	on
en0-Online			
pt.itsc.austin.i~	riogrande	on	on
en0-Online			
trinity.itsc.aus~	riogrande	on	on
en0-Online			
uk.itsc.austin.i~	riogrande	on	on
en0-Online			
us.itsc.austin.i~	riogrande	on	on
en0-Online			

The **csmstat** command gathers node reachability, power status and network interface status for one or more nodes and displays the output. The default

ordering for output is by host name. If there are multiple hardware control points for node (for example, multiple Hardware Management Consoles), then the first hardware control point in the list is shown.

Copy and store vital CSM data using csmbackup

The **csmbackup** command copies vital CSM data from the management server and stores the data in the directory specified by the **-d** flag. The command backs up a CSM management server in case of a hardware problem, or if the management server is being changed to another machine. For example, the following command saves all current CSM data into the directory `/backup/CSMServerbackup`:

```
csmbackup -d /backup/CSMServerbackup
```

Distributed Shell (dsh) execution

The **dsh** command runs commands concurrently on remote targets, that is, on nodes, hardware devices, or both. Targets can be selected from multiple contexts. A *context* is a target database that contains node and device definitions, such as the CSM database. The **dsh** command issues a remote shell command for each target specified and returns the output from all targets, formatted so that command results from all nodes can be managed.

The **dsh** command is a powerful utility for managing and executing common tasks in one step from a central server on all, or selected, managed nodes. For an example illustrating the stop and start of WebSphere from the CSM Management Server, refer to Example 4-12 on page 139.

You can also gather information about resources allocated to all partitions, and then generate a general report repository. As shown in Example 3-22, we ran the command **lsattr -El mem0** on all nodes (br,de,pl,pt,uk,us) and wrote the dsh execution report into directory `/tmp/Nodes` on the CSM Management Server.

Example 3-22 dsh execution to generate report

```
#dsh -n br,de,pl,pt,uk,us --report /tmp/Nodes --report-name  
MemasofOct2006 "lsattr -El mem0"
```

A report for this command has been generated and can be found on the Managing Machine at `/tmp/Nodes/MemasofOct2006.0001`.

The **--report** option enables report generation and specifies the path to the directory where reports are saved. All output from each host is saved to individual output and error files, and a report is generated. All output from each host is saved to individual output and error files. Summary HTML and XML report files are created, in addition to an XML results file, as shown in Figure 3-25 on page 94.

Distributed Copy (dcp) execution

The **dcp** command concurrently copies files to or from remote target nodes, hardware devices, or both. Targets can be selected from multiple contexts. The **dcp** command is a CSM Distributed Shell Utility. The configuration and environmental settings for **dsh** impact the behavior of **dcp**.

In illustrated in Example 3-24, we used the **dcp** command to transfer the report that was generated by **dsh** in Example 3-22 on page 92 onto a node that has an IBM HTTP server running.

First we determined where an HTTP server was up and running, as shown in Example 3-23.

Example 3-23 dsh execution to check for a running process

```
#dsh -n br,de,pl,pt,uk,us "ps -ef |grep HTTPServer |grep -v  
DSH_TARGET_ |grep -v grep  
pt.itsc.austin.ibm.com:      root 327778      1   0   Oct 05      -  
0:02 /usr/IBM/HTTPServer/bin/httpd -d /usr/IBM/HTTPServer -k start
```

The IBM HTTP server was up and running, so we copied the files using the **dcp** recursive option **-R** onto node **pt**. To verify that the files are on the remote node **pt**, we generated a directory listing by using the **dsh** command with the **ls** command, as shown in Example 3-24.

Example 3-24 dcp command to copy files

```
#dcp -n pt -R /tmp/Nodes /usr/IBM/HTTPServer/htdocs/en_US/Nodes  
#dsh -n pt "ls /usr/IBM/HTTPServer/htdocs/en_US/Nodes/*"  
pt.itsc.austin.ibm.com: MemasofOct2006.0001  
pt.itsc.austin.ibm.com: index.html  
pt.itsc.austin.ibm.com: listOfReports.html
```

The result can be viewed in a Web browser, as shown in Figure 3-25 on page 94.

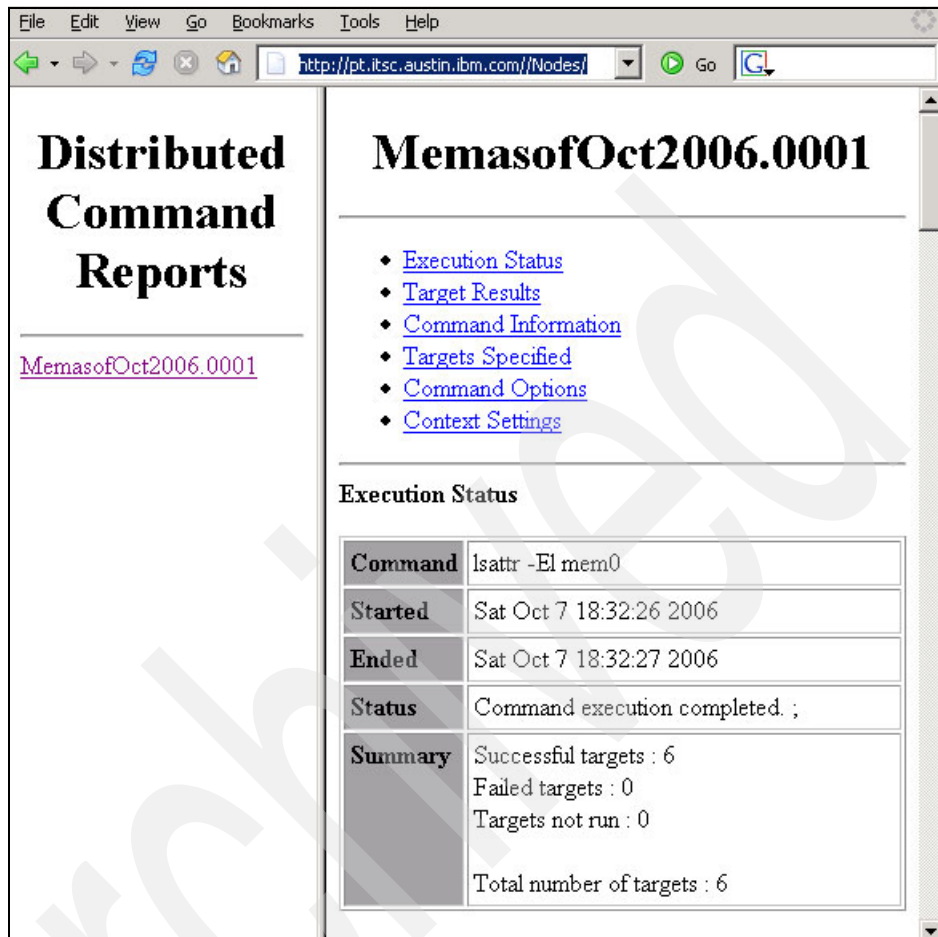


Figure 3-25 dsh execution report in Web browser

Using dsh and dcp to create WebSphere backups

You can also run WebSphere backups of the cell on a regular basis. WebSphere Application Server provides the command line tool **backupConfig.sh** for this purpose. This tool is located in the bin directories for both WebSphere Application Server and the Network Deployment run-time.

Example 3-25 on page 95 shows the script `backupConfig.sh` being executed on node `pt` using the absolute pathname to the WebSphere bin directory. To prevent changes from being made while the backup is running, we did *not* use the option **-nostop**. As a result, we needed to restart the server after **backupConfig.sh** finished.

Example 3-25 dsh command to run backupConfig.sh

```
#dsh -n pt -s "/usr/IBM/WebSphere/AppServer/bin/backupConfig.sh"
pt.itsc.austin.ibm.com: ADMU0116I: Tool information is being logged in
file
pt.itsc.austin.ibm.com:
/usr/IBM/WebSphere/AppServer/profiles/Dmgr/logs/backupConfig.log
pt.itsc.austin.ibm.com: ADMU0128I: Starting tool with the Dmgr profile
pt.itsc.austin.ibm.com: ADMU5001I: Backing up config directory
pt.itsc.austin.ibm.com:
/usr/IBM/WebSphere/AppServer/profiles/Dmgr/config to file
pt.itsc.austin.ibm.com:
/home/root/WebSphereConfig_2006-10-07.zip
pt.itsc.austin.ibm.com: ADMU0505I: Servers found in configuration:
pt.itsc.austin.ibm.com: ADMU0506I: Server name: dmgr
pt.itsc.austin.ibm.com: ADMU2010I: Stopping all server processes for
node ITSOProdCellManager
pt.itsc.austin.ibm.com: ADMU0510I: Server dmgr is now STOPPED
pt.itsc.austin.ibm.com:
.....
pt.itsc.austin.ibm.com: ADMU5002I: 456 files successfully backed up
#dsh -s -n pt "/exports/systemfiles/startup/rc.was"
pt.itsc.austin.ibm.com: ADMU0116I: Tool information is being logged in
file
pt.itsc.austin.ibm.com:
/usr/IBM/WebSphere/AppServer/profiles/Dmgr/logs/dmgr/startServer.log
pt.itsc.austin.ibm.com: ADMU0128I: Starting tool with the Dmgr profile
pt.itsc.austin.ibm.com: ADMU3100I: Reading configuration for server:
dmgr
pt.itsc.austin.ibm.com: ADMU3200I: Server launched. Waiting for
initialization status.
pt.itsc.austin.ibm.com: ADMU3000I: Server dmgr open for e-business;
process id is 389206
```

To keep a copy of the WebSphere backups, use the **dcp** command again to copy files into a backup directory on the CSM Management server. You must use the **dcp** command option **-P** to pull the files from the client node to the management node, as shown in Example 3-26.

Example 3-26 dcp command to pull WebSphere Application Server backupconfig File

```
#dcp -n pt -P /home/root/WebSphereConfig_2006-10-07.zip
/backup/wasconfigbackups
#ls -ltr /backup/wasconfigbackups
```

Network Installation Manager

Network Installation Manager (NIM) provides remote installation of the operating system and manages software updates, and it can be configured to install and update third-party applications. Although both the NIM server and client filesets are part of the operating system, a separate NIM server has to be configured which will keep the configuration data and the installable product filesets. This server can provision several clients.

An `image_data` resource contains information about the physical disks and file systems configured for rootvg at installation time. Customizing this resource allows you to predefine settings for provisioned systems. This can be important in the case of cloned installation and storage virtualization.

You can set up NIM initially by using Basic NIM Environment (Easy Startup) or Advanced Configuration. You can start by using Easy Startup and later migrate to Advanced Configuration.

The Easy Startup (EZNIM) feature helps system administrators by organizing commonly-used NIM operations and simplifying frequently-used advanced NIM operations.

Features of SMIT EZNIM include:

- ▶ Task-oriented menus
- ▶ Automatic resource naming that includes the level of the software used to create NIM resources
- ▶ A review of steps that will take place before executing a task, whenever possible

Use the SMIT `eznim` fast path to open the EZNIM main menu. If the NIM environment has *not* been set up on your system, the EZNIM main menu displays the following options:

- ▶ Configure as a NIM Master
- ▶ Configure as a NIM Client

Keep these guidelines in mind when implementing NIM:

- ▶ The NIM master must always be at the highest level of the AIX release and maintenance level that you are going to install.
- ▶ You need a network connection between the NIM master and clients.
- ▶ To speed up NIM installations or other NIM operations, it may be helpful to separate the NIM network from the public network.

- ▶ We recommend that you create a separate volume group to be used for NIM operations on the NIM master.
- ▶ The NIM master and NIM client Domain Name Service (DNS) host names must be consistently resolvable.
- ▶ Back up your NIM Database on a frequent basis either using SMIT fastpath `nim_backup_db` or scheduled backups. The `nim_master_recover` command can then restore and update the NIM database from a backup tar file, or update the database from a `mksysb`.

If you are restoring a `mksysb` of your primary NIM master, then you can update the NIM database restored from the `mksysb`. In that case, you will not need the `nimdb.backup` tar file.

- ▶ If you use Cluster System Manager (CSM), then NIM operation can be easily applied using the WebSM Interface; however, the nodes must be defined in CSM first.

Tuning the NIM client-request processing

The multithreaded option on the `nimesis` daemon provides better handling of the volume of client information change requests and client state changes. Without the use of the multithreaded option, the NIM master can become overloaded by activity on the NIM database and the number of active processes, resulting in simultaneous failures during the installation of a large number of client machines.

The number of threads assigned to this daemon determines how many simultaneous NIM client requests can be handled in the NIM environment. Because most of the NIM client requests are processed rapidly, it is not necessary to have one thread for every client installing. The number of threads needed to support the activities in a NIM environment is dependent upon several items. The following should be considered when determining the number of threads:

- ▶ Number of clients that will be operated on at the same time
- ▶ Processing capacity of the NIM master machine
- ▶ What type of operations are planned

In general, one thread can support two to four clients that are installing BOS at the same time. The settings can be changed through CSM WebSM access Menu **Network Installation Manager -> Advanced Configuration -> Tune Client Requests...**, as shown in Figure 3-26 on page 98. Use the dialog to complete the task.

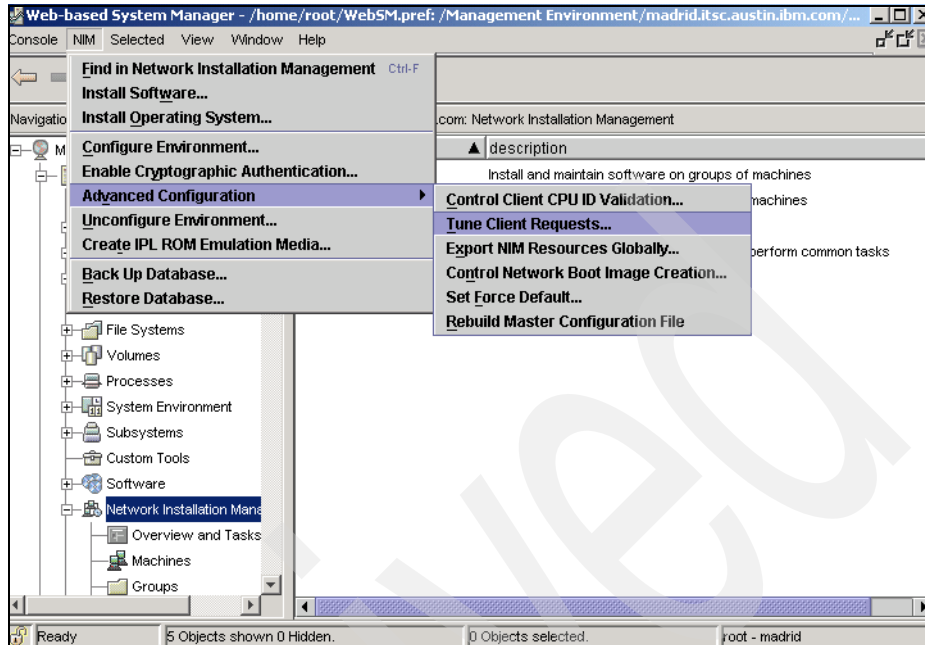


Figure 3-26 CSM NIM Tune Client Request GUI

Set the new value to 64; see Figure 3-27.

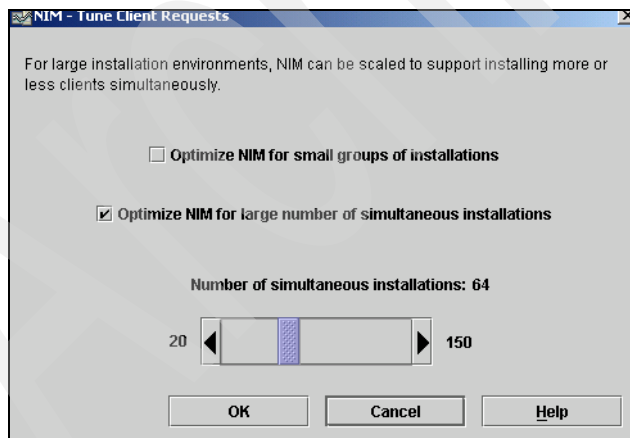


Figure 3-27 CSM NIM Tune Client Request GUI change

Using AIX Network Installation Manager and CSM

You use AIX Network Installation Manager (NIM), CSM, and hardware commands to install AIX on the cluster nodes. NIM enables a cluster

administrator to centrally manage the installation and configuration of AIX and optional software on machines within a network environment. In a CSM cluster, you may install and configure NIM on an AIX management server or on one or more AIX install servers.

Installing AIX and CSM on cluster nodes

Follow these steps to install AIX onto CSM Cluster nodes:

1. Verify the node definitions.
2. Create CSM node groups (optional).
3. Validate hardware control (required for hardware control).
4. Get network adapter information.
5. Set up Network Installation Manager (NIM).
6. Create additional NIM network definitions and routes (optional).
7. Create NIM machine definitions.
8. Create NIM machine groups (optional).
9. Prepare customization scripts (optional).
10. Prepare for secondary adapter configuration (optional).
11. Set up cluster configuration (optional).
12. Verify authentication methods for NIM (optional).
13. Prepare NIM to add the nodes.
14. Add OpenSSH and OpenSSL software (optional).
15. Add Kerberos client software (optional).
16. Add the nodes to the cluster.
17. Initiate a network installation of the nodes.
18. Monitor and verify the installation.
19. Enable Kerberos Version 5 remote commands (optional).
20. CSM post-installation tasks - this includes the following steps:
 - a. Getting started with the newly installed cluster.
 - b. Enabling remote commands to use Kerberos Version 5 authentication (optional).
 - c. Understanding the installation and configuration log files - for details about this step, refer to the following address:

http://publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp?topic=/com.ibm.help.csm.doc/csm_books/csm_admin/am7ad13014.html

Maintaining NIM lpp_sources for each AIX release

One requirement of the Network Installation Manager (NIM) is that the AIX level of an lpp_source and the Shared Product Object Tree (SPOT) have to be the same as the AIX level in system backup to be restored or the node to be booted into maintenance. Trying to install a mksysb taken from a system running AIX 5.3ML05 using lpp_source and SPOT at AIX 5.3ML04 Level likely will fail.

In an environment with client nodes running at different levels of an AIX release, an lpp_source and a SPOT for each of these levels must be available to be able to restore client nodes backups or perform maintenance boots of the client nodes.

For our example with the different AIX 5.3 level, there is one lpp_source with the AIX 5.3 ML04 installation images at /export/lpp_source/lpp_source53ML04.

To create a new lpp_source for AIX 5.3 ML05 at directory /export/lpp_source/lpp_source53ML05, we can simply copy the AIX 5.3 ML04 lpp_source to the AIX 5.3 ML05 location and apply the ML05 updates to it. This requires more disk space, but ensures the integrity of the installation source.

Next steps are to create the new NIM lpp_source and SPOT resources for the new level AIX 5.3 ML05 and update the target partitions.

Installing the Virtual I/O Server using NIM

You can use the following procedures to install the Virtual I/O (VIO) server into environments managed by the HMC or Integrated Virtualization Manager using Network Installation Management (NIM).

You need the following files before beginning this procedure. These files are located on the Virtual I/O Server installation media:

- ▶ nimol/ioserver_res/mksysb (the mksysb image)

In addition, the following system requirements must be met:

- ▶ The NIM server with AIX 5.3 with 5300-03 or higher and a file system with at least 700 MB free space available
- ▶ A logical partition of type Virtual I/O Server containing an Ethernet adapter connected to an active network for installing the Virtual I/O Server
- ▶ A storage controller containing at least 16 GB of disk space allocated to the Virtual I/O Server partition

After the prerequisites have been met, you can also install a Virtual I/O Server or an Integrated Virtualization Manager through the SMIT interface. Run **smitty installios** to get access to the SMIT interface to the **installios** command.

The installios setup process creates the following NIM resources to start the installation:

bosinst_data, installp_bundle, lpp_source, mksysb, resolv_conf, SPOT, Client definition.

You need to know the following information as defined within the HMC environment:

HMC Name, Managed System Name, Partition Name, Partition Profile Name.

The full **installios** command is shown here:

```
/usr/sbin/installios -d'cd0' -h'riogrande.itsc.austin.ibm.com' -s  
'p5+-9133-55A-SN10D1FAG' -p'wasp5l_vio' -r'wasp5l_vio_limited'  
-i'9.3.5.170' -S'255.255.255.0' -g'9.3.5.41' -P'100' -D'full'  
-l'en_US' -N'
```

If you are installing the Virtual I/O Server logical partition, and if Secure Shell (SSH) and credentials have been configured on the NIM master, then the partition is network-booted from the Hardware Management Console (HMC) to begin the installation.

Installing the WebSphere partitions using NIM

Performing a NIM mksysb installation is faster than performing a NIM Runtime (rte) installation. And, using mksysb, you can optionally include other installed software. A mksysb image is the system backup image created by the AIX **mksysb** command. You can use this image to install other machines or to restore the machine that was the source of the mksysb.

1. After you complete all steps listed in “Installing AIX and CSM on cluster nodes” on page 99, you can install AIX on the nodes.
2. We suggest that you install it on the node with the basic operating system to create a Master image (also known as mksysb image).

Issue NIM bos_inst operation with the source attribute set to rte for one node to be installed. The command for node trinity is shown here:

```
csmsetupnim -n trinity  
nim -o bos_inst -a source=rte -a lpp_source=lpp_source53ML05 -a  
spot=spot535 trinity
```

3. After the installation, configure and prepare the operating system to run WebSphere. You can, for example, preinstall the AIX ToolBox filesets, add the default user needed for your special needs, and preset NFS remote mounts. (Refer to Chapter 4, “AIX configuration” on page 127 for details about what we applied, in our case, to the AIX operating system.)
4. Use the installed node to create a NIM mksysb resource and a mksysb image. You can perform this task by using a single NIM command, as shown here:

```
nim -o define -t mksysb -a server=master -a \  
location=/export/mksysb/AIX53ML05_WebSphere Application Server  
61_Base -a mk_image=yes -a \ source=trinity WebSphere Application  
Server 61AIX535_mksysb
```

The created NIM resource can be verified.

5. Use this mksysb image and NIM resource to install AIX on all new partitions.

Before installing the new node, issue the command **csmsetupnim** to set up the CSM customization scripts to the automatically update command to exchange HBA public keys and update the Trusted Host List after AIX is installed on a node.

Issue the NIM bos_inst operation with the source attribute set to mksysb for one or a group of nodes to be installed. The commands for node us is shown here:

```
csmsetupnim -n us
nim -o bos_inst -a source=mksysb -a mksysb=WebSphere Application
Server 61AIX535_mksysb -a spot=spot535 us
```

Issue the command **lsnim -c resources us** to verify that all resources have been allocated and that the Node NIM status is as required. The command output is shown here:

```
osprereboot      script
WebSphere Application Server 61AIX535_mksysb mksysb
spot535          spot
boot             boot
```

You can check the NIM status for the object machine by using the command **lsnim**. In our case, because we were only interested in the value of the NIM attribute Cstate, we used the command **lsnim -l us | grep Cstate** to check the output, as shown here:

```
Cstate          = BOS installation has been enabled
```

6. Initiate the node installation.

To initiate node installation, use **netboot** command. The command we used in our case is shown here:

```
netboot -n us
```

The progress is written to the CSM logfile `/var/log/csm/netboot`. In addition, you can use the **rconsole** command to connect to the partition from the CSM management server.

The **rconsole** command provides remote console support for nodes and devices in a cluster. The command uses the CSM database to determine the nodes and devices and their console access information. It provides similar functionality to the HMC Virtual Terminal without needing to have a HMC GUI session open.

3.2.3 Accessing the CSM through WebSM

After you start the WebSM Client as described in “Remote access to the HMC GUI” on page 39, you must provide the CSM Management Server host name instead of the HMC host name.

Figure 3-28 shows the CSM Management Server hostname `madrid.itsc.austin.ibm.com` that we provided in our case.

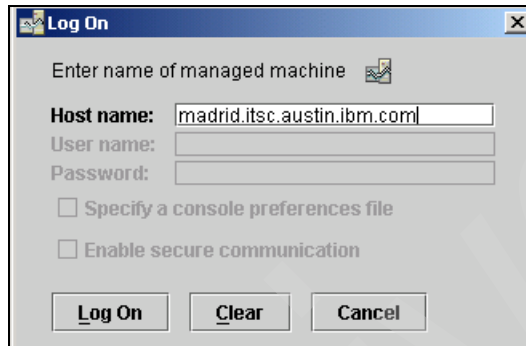


Figure 3-28 WebSM Host name for CSM Managed Server Login panel

After the server has been contacted, you have to provide a valid AIX user and word combination to access the CSM WebSM Interface. After the WebSM CSM Interface is loaded, a panel similar to Figure 3-29 on page 104 displays.

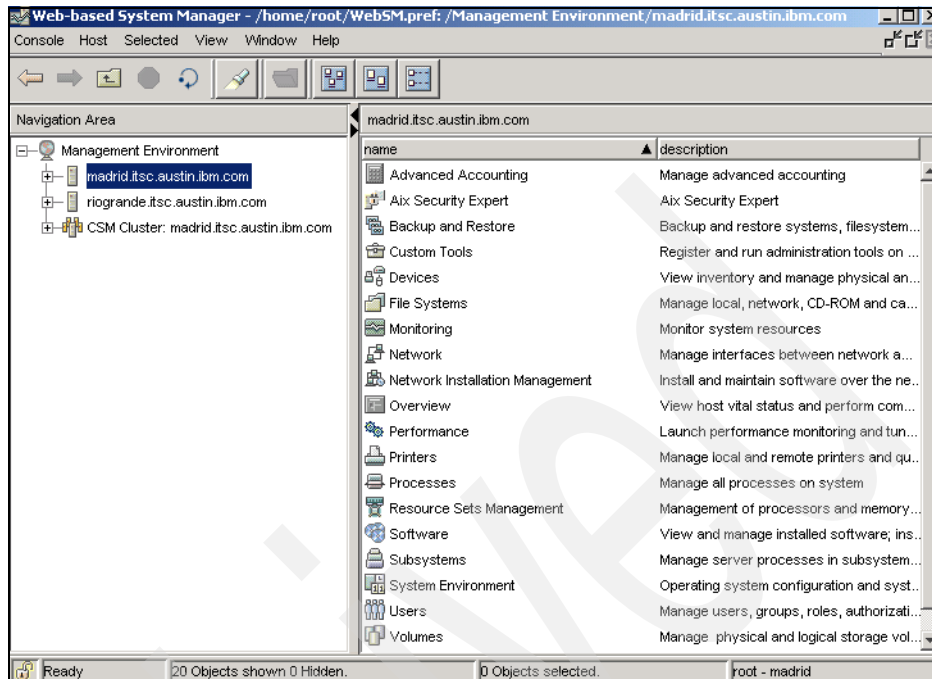


Figure 3-29 WebSm CSM Cluster root view

Note: Figure 3-29 on page 104 shows that access to the HMC is possible through CSM WebSM.

Accessing a node through CSM WebSM access

WebSM CSM enables you to manage CSM cluster nodes through CSM after the CSM Management Server has been accessed as described in 3.2.3, “Accessing the CSM through WebSM” on page 103.

In our case, we selected the node brazos.itsc.austin.ibm.com in the cluster view, as shown in Figure 3-30 on page 105:

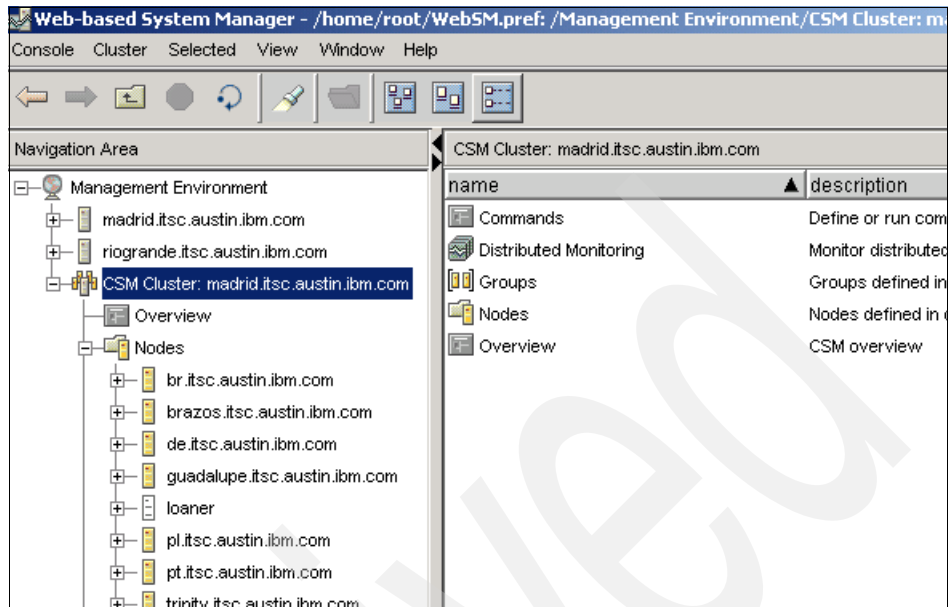


Figure 3-30 WebSM CSM node view managed server

The Log On panel displayed in Figure 3-31 shows that Host name `brazos.itsc.austin.ibm.com` was preselected.

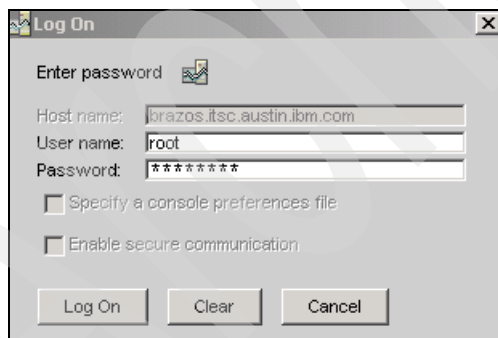


Figure 3-31 WebSM User for CSM node Log On panel

Type in the required information and press Log On to access the WebSM-based System Management view for node `brazos`, as shown in Figure 3-32 on page 106.

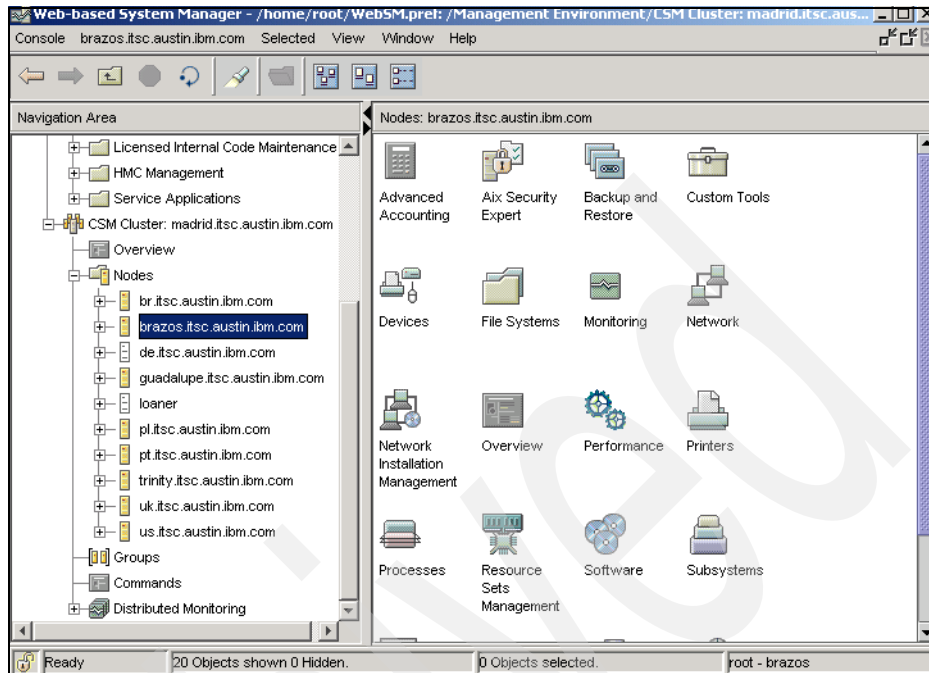


Figure 3-32 WebSM CSM root view - node brazos.itsc.austin.ibm.com

In theory, we now had total control over the system, exactly as if we connected directly to the WebSM local server or via a command line interface.

Reliable Scalable Cluster Technology

Reliable Scalable Cluster Technology (RSCT) plays key role in provisioning for pSeries. You can use some of its functions to make the system more stable. For more information about RSCT and its components, refer to *IBM Reliable Scalable Cluster Technology Administration Guide*, at the following address:

<http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.rsct.doc/rsctbooks.html>

System events generated by the operating system can be monitored by the RSCT Resource Monitoring and Control (RMC) subsystem. For example, you can define an RMC monitor to control the size of the WebSphere filesystem, which is by default /usr/IBM or (if used), the filesystem that holds the WebSphere System and server log files.

Under normal operation mode, we consider that this filesystem should never be 100% full. When the size of the filesystem reaches 90%, RMC automatically runs a script that extends the filesystem by the size of one LPAR and creates a

notification that can be, for example, an e-mail to the administrator or a monitoring application, as well as an SNMP trap.

The basic flow of the monitor creation is as follows:

1. Create the condition.
2. Create the response.
3. Link the response with the condition.
4. Start the monitor.

AIX delivers a comprehensive set of predefined RSCT Events that can be monitored. Use the command **lscondition** to generate a list of all available conditions. Use the command **lsresponse** to list the configured responses.

Because persistent attributes are generally unchanging, you will usually monitor a dynamic attribute. If none of the dynamic attributes provided by the resource managers contains the value you want to monitor, you can create a sensor that includes a command to be run by RMC to retrieve the value you want to monitor.

Resource Monitoring and Control subsystem

The Resource Monitoring and Control (RMC) subsystem monitors and queries system resources. The RMC daemon manages an RMC session and recovers from communications problems. The RMC subsystem is used by its clients to monitor the state of system resources and to send commands to resource managers. The RMC subsystem acts as a broker between the client processes that use it and the resource manager processes that control resources.

A resource manager is a standalone daemon that maps resource and resource class abstractions into calls and commands for one or more specific types of resources. A resource manager contains definitions of all resource classes that the resource manager supports. A resource class definition includes a description of all attributes, actions, and other characteristics of a resource class. These resource classes are accessible and their attributes can be manipulated through IBM Web-based System Manager or through the command line.

Basic configuration of RSCT/RMC

RSCT can be configured through the WebSM-based interface or command line interface. Figure 3-33 on page 108 shows the condition available on node br.itsc.austin.ibm.com, which we opened using the CSM WebSM Cluster Interface.

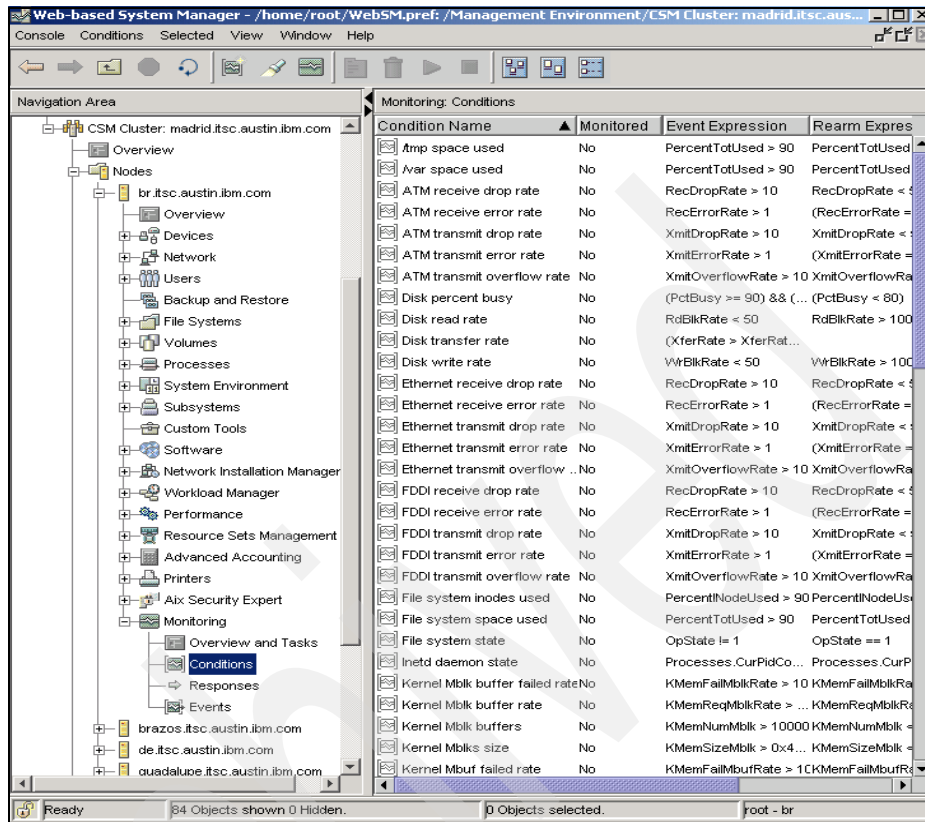


Figure 3-33 RSCT Monitoring conditions for node br.itsc.austin.ibm.com

If you intend to use RSCT monitoring on a cluster level, you can use Distributed Monitoring Access as shown in Figure 3-34 on page 109.

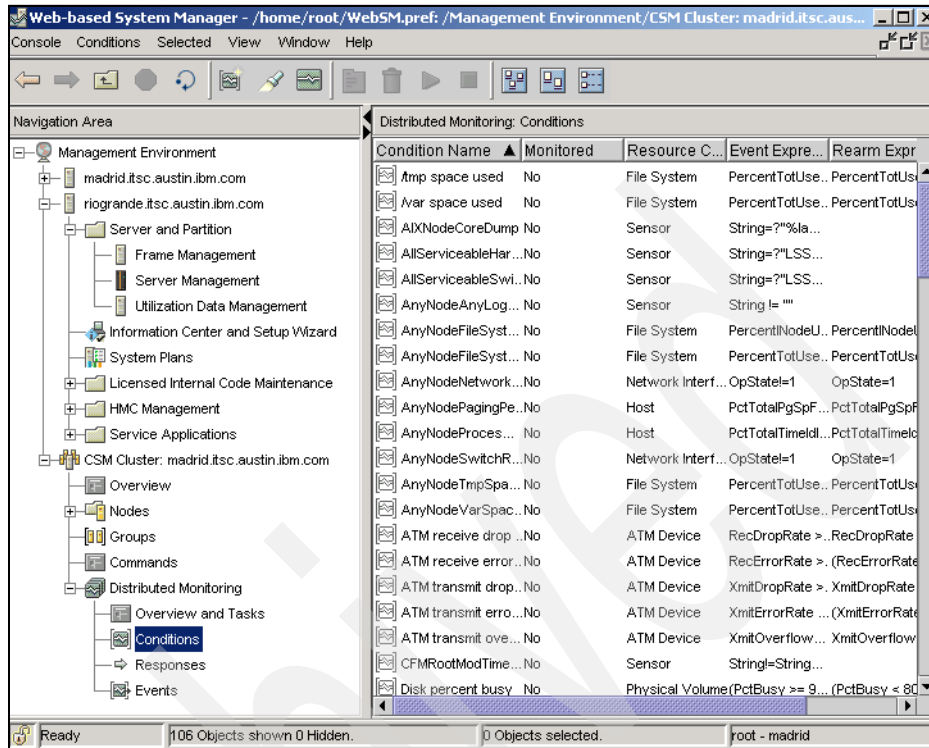


Figure 3-34 RSCT monitoring conditions for the cluster

Events triggered by one of monitors can be checked through WebSM CSM cluster access, as shown in Figure 3-35 on page 110. However, there are also other options available to manage events.

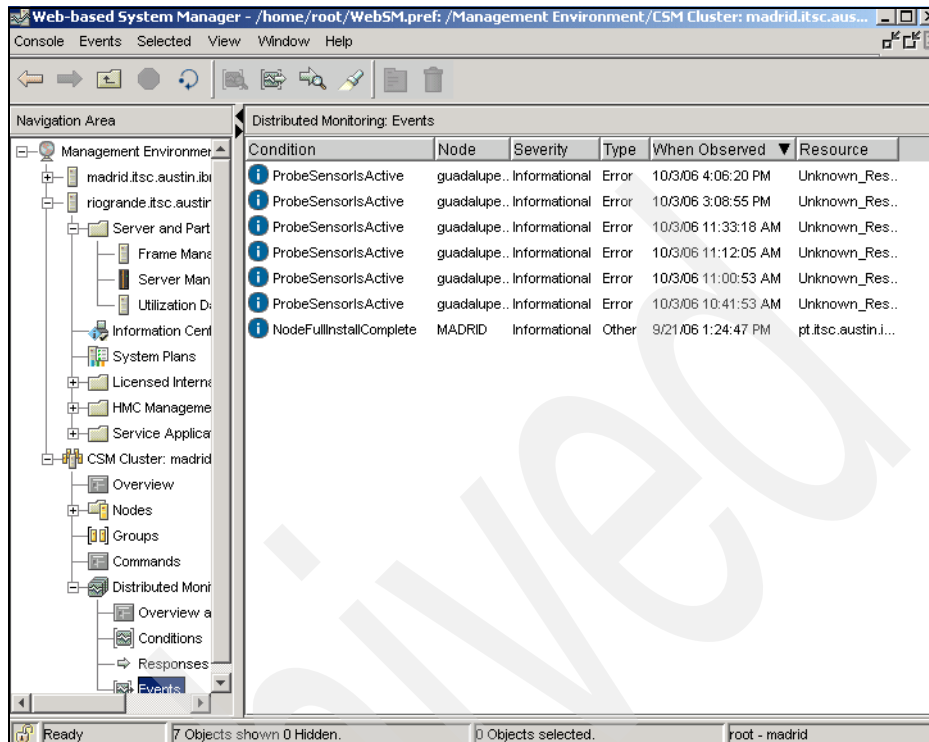


Figure 3-35 RSCT Monitoring Events for the cluster

Locked conditions and responses

The following Event Response Resource Manager (ERRM) conditions, responses, and associations are locked and are not intended to be changed.

However, you can unlock these resources if changing them is absolutely necessary.

Locked conditions:

- ▶ NodeFullInstallComplete
- ▶ NodeManaged
- ▶ NodeChanged
- ▶ AllServiceableHardwareEvents
- ▶ AllServiceableSwitchEvents

Locked responses:

- ▶ RunCFMToNode
- ▶ GatherSSHHostKeys
- ▶ rconsoleUpdateResponse

- ▶ SetupSSHAndRunCFM
- ▶ AuditLogServiceableEvents

Locked associations (condition-response pairs):

- ▶ NodeChanged-rconsoleUpdateResponse

Setting up an efficient monitoring system

Following these steps will help you to set up an efficient monitoring system:

1. Review the predefined conditions of your interests.

Use them as they are, customize them to fit your configurations, or use them as templates to create your own.

2. Review the predefined responses.

Customize them to suit your environment and your working schedule. For example, the predefined response Critical notifications is predefined with three actions:

- a. Log events to /tmp/criticalEvents.
- b. Send an e-mail to root.
- c. Broadcast the message to all logged-in users when an event or rearm event occurs.

You may modify the response, for example, to log events to a different file when events occur; e-mail you during non-working hours, and add a new action to page you only during working hours. With such a setup, different notification mechanisms can be automatically switched, based on your working schedule.

3. Reuse the responses for conditions.

For example, you can customize the three severity responses (Critical notifications; Warning notifications; and Informational notifications) to take actions in response to events of different severities, and associate the responses to the conditions of respective severities. With only three notification responses, you can be notified of all events with respective notification mechanisms based on their urgencies.

4. After monitoring is set up, your system continues being monitored whether your Web-based System Manager session is running or not.

To determine the system status, you may bring up a Web-based System Manager session and view the Events plug-in, or simply use the **lsaudrec** command from the command line interface to view the audit log, as shown in Example 3-27 on page 112.

Example 3-27 Sample output of command *lsaudrec*

```
[0:root@MADRID:]/home/root # lsaudrec
10/03/06 16:06:20      ERRM Error      Error detected during monitoring
for ProbeSensorIsActive on guadalupe.itsc.austin.ibm.com.
10/03/06 16:08:08      ERRM Info      Monitoring of condition
ProbeSensorIsActive resumed after the Resource Manager recovered from a
failure on guadalupe.itsc.austin.ibm.com.
```

The same information can be retrieved using WebSM CSM access, as shown in Figure 3-36.

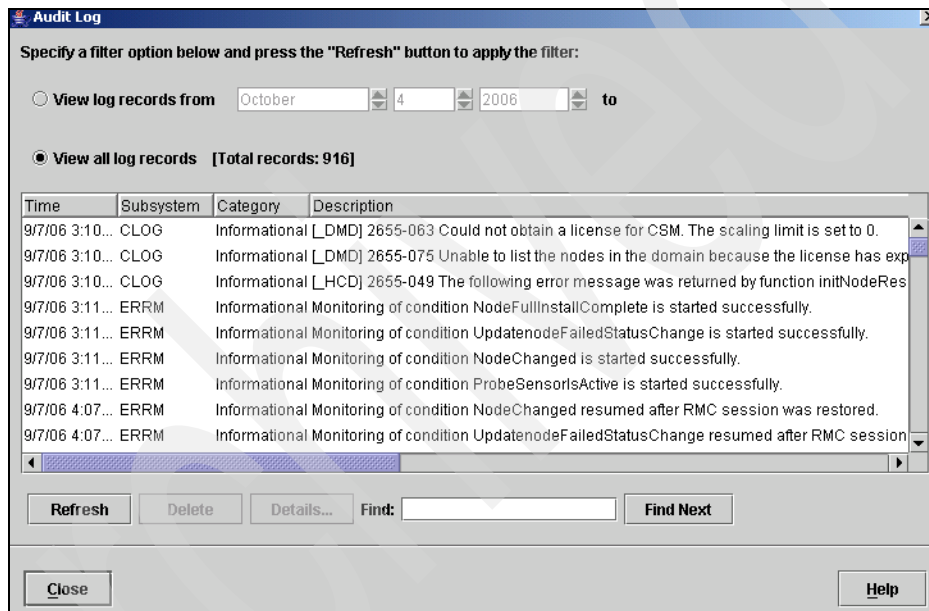


Figure 3-36 Output of Audit Log from WebSM for the cluster

Setting up monitoring for the WebSphere filesystem

In general you need to perform the following steps to set up an RSCT Monitor:

1. Define the RMC condition.
2. Define the RMC response.
3. Link the condition with the response.
4. Start the monitor.
5. Start the execution.

In this section, we use these steps to demonstrate how to define an RMC monitor to control the size of the WebSphere filesystem.

In our case, we assumed that the default filesystem /usr/IBM was used on the installation. We wanted to check the size of the filesystem with a threshold set to 90% filling ratio.

1. Define the RMC condition.

We issued the command **mkcondition** to create the /usr/IBM space used condition:

```
mkcondition -r "IBM.FileSystem" -e "PercentTotUsed > 90" -d
"/usr/IBM 90% full" -s "Name == \"/usr/IBM\""" -S c "/usr/IBM space
used"
```

We checked the condition and status by using the **lscondition** command; Example 3-28 displays the output.

Example 3-28 lscondition output for mkcondition

```
#lscondition -a |grep "/usr/IBM"
"/usr/IBM space used"          "Not monitored"
#lscondition "/usr/IBM space used"
Displaying condition information:
```

condition 1:

```
      Name           = "/usr/IBM space used"
      MonitorStatus   = "Not monitored"
      ResourceClass   = "IBM.FileSystem"
      EventExpression  = "PercentTotUsed > 90"
      EventDescription = "/usr/IBM 90% full"
      RearmExpression  = ""
      RearmDescription = ""
      SelectionString  = "Name == \"/usr/IBM\"""
      Severity        = "c"
      NodeNames       = {}
      MgtScope        = "1"
```

#

2. Define the RMC response.

We used the **mkresponse** command to create the response with name "/usr/IBM Filesystem full" that has an action named "E-mail root":

```
mkresponse -n "E-mail root" -s "/usr/sbin/rsct/bin/notifyevent root"
-e b "/usr/IBM Filesystem full"
```

We checked the condition and status by using the **lsresponse** command. Example 3-29 on page 114 displays the output.

Example 3-29 lsresponse output for mkresponse

```
#lsresponse -a
Displaying response information:
ResponseName
"/usr/IBM Filesystem full"
#lsresponse "/usr/IBM Filesystem full"
Displaying response information:

ResponseName    = "/usr/IBM Filesystem full"
Action          = "E-mail root"
DaysOfWeek      = 1-7
TimeOfDay       = 0000-2400
ActionScript    = "/usr/sbin/rsct/bin/notifyevent root"
ReturnCode      = 0
CheckReturnCode = "n"
EventType       = "b"
StandardOut     = "n"
EnvironmentVars = ""
UndefRes        = "n"
```

3. Link the condition with the response.

We issued the **mkcondresp** command to link the condition and the response:

```
mkcondresp "/usr/IBM space used" "/usr/IBM Filesystem full"
```

We checked the condition and status by using the **lscondresp** command.
Example 3-30 displays the output.

Example 3-30 lscondresp output for mkcondresp

```
#lscondresp
Displaying condition with response information:
Condition      Response      State
"/usr/IBM space used" "/usr/IBM Filesystem full" "Not active"
#lscondresp "/usr/IBM space used" "/usr/IBM Filesystem full"
Displaying condition with response information:

condition-response link 1:
Condition = "/usr/IBM space used"
Response  = "/usr/IBM Filesystem full"
State     = "Not active"
```

4. Start the monitor.

We started monitoring by issuing the **startcondresp** command:

```
startcondresp "/usr/IBM space used" "/usr/IBM Filesystem full"
```

We verified that the monitor was operational by using the **lscondresp** command. Example 3-31 displays the output.

Example 3-31 lscondresp output for startcondresp

```
#lscondresp
Displaying condition with response information:
Condition      Response      State
"/usr/IBM space used" "/usr/IBM Filesystem full" "Active"
lscondresp "/usr/IBM space used" "/usr/IBM Filesystem full"
Displaying condition with response information:

condition-response link 1:
    Condition = "/usr/IBM space used"
    Response  = "/usr/IBM Filesystem full"
    State     = "Active"
```

In addition, the status of the condition named `"/usr/IBM space used"` changed from `"Not monitored"` to `"Monitored"`; you can verify this by using the **lscondition** command.

The same information can be retrieved using the WebSM CSM access, as shown in Figure 3-37 on page 116. In addition, you can see that the value for Management Scope is `Local Machine`.

You can create conditions for which the Management Scope is `Domain`. These conditions will then be applied to the whole domain.

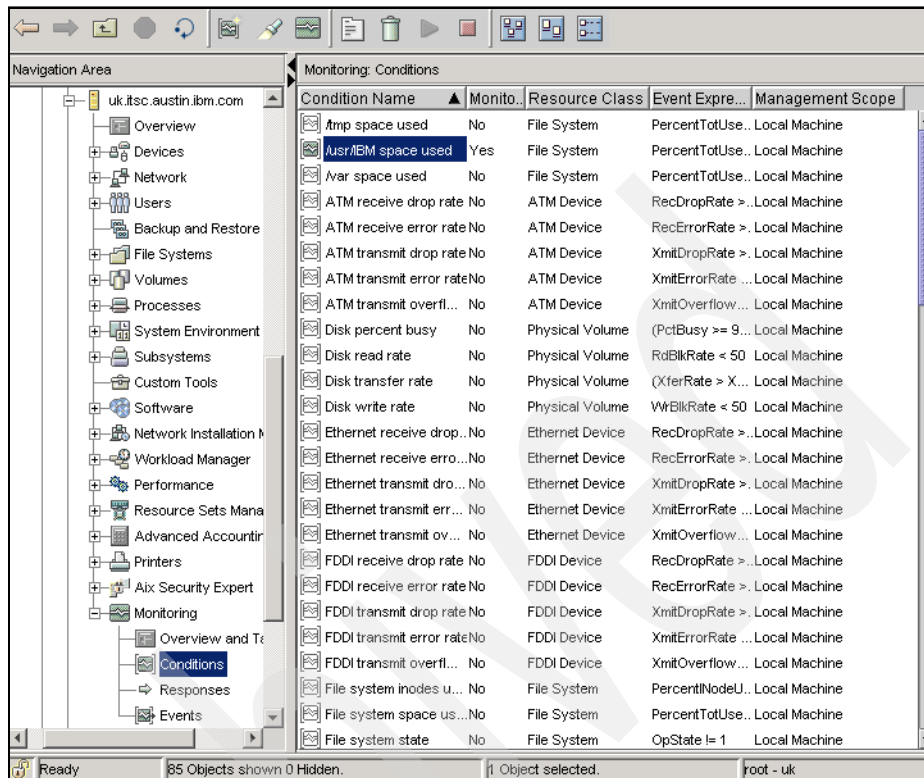


Figure 3-37 RMC Condition Status for Node

5. Start the execution.

After the condition threshold was reached as defined, the action was triggered and an e-mail was initiated to the defined user; see Example 3-32.

Example 3-32 Sample RCST e-mail for /usr/IBM space used

```
[0:root@uk:]/usr/IBM # mail
Mail [5.2 UCB] [AIX 5.X] Type ? for help.
"/var/spool/mail/root": 2 messages 2 new
>N 1 root          Wed Oct  4 17:09 28/723  "/usr/IBM space used"
essage 1:
From root Wed Oct  4 17:09:31 2006
Date: Wed, 4 Oct 2006 17:09:31 -0500
From: root
To: root
Subject: /usr/IBM space used
```

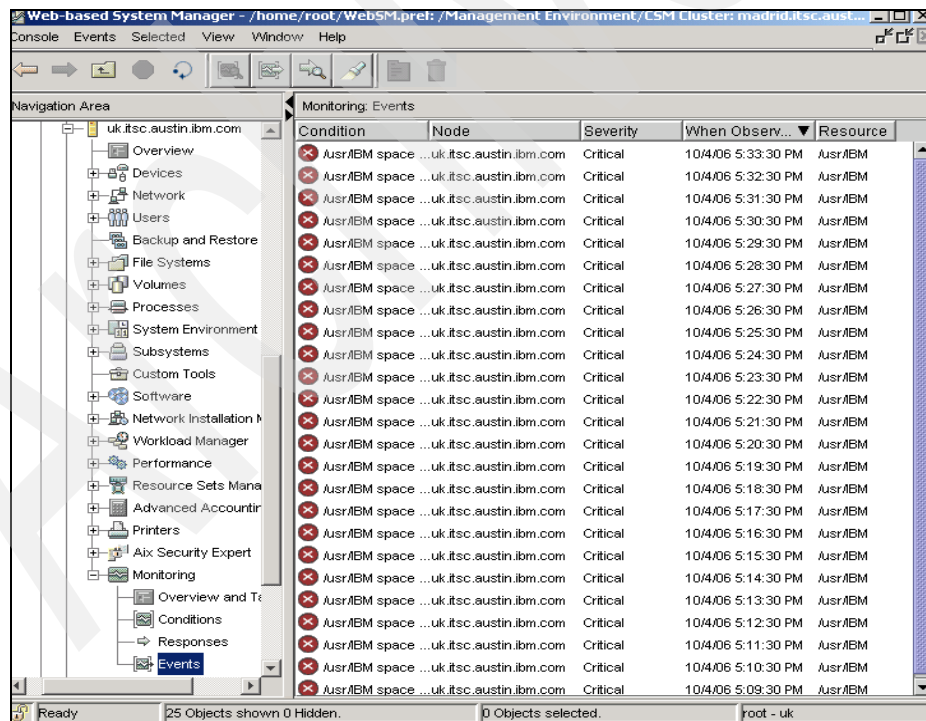
=====

Wednesday 10/04/06 17:09:30

Condition Name: /usr/IBM space used
Severity: Critical
Event Type: Event
Expression: PercentTotUsed > 90

Resource Name: /usr/IBM
Resource Class: IBM.FileSystem
Data Type: CT_INT32
Data Value: 96
Node Name: uk.itsc.austin.ibm.com
Node NameList: {uk.itsc.austin.ibm.com}
Resource Type: 0
=====

Also note that, as displayed in Figure 3-38, an entry in the RCST Event log was written until the problem was resolved.



The screenshot shows the 'Monitoring: Events' window in the Web-based System Manager. The left pane shows a navigation tree with 'Events' selected. The main pane displays a table of events, all of which are critical and related to the '/usr/IBM space used' condition on the 'uk.itsc.austin.ibm.com' node. The events occurred between 10/4/06 5:09:30 PM and 10/4/06 5:33:30 PM.

Condition	Node	Severity	When Observed	Resource
/usr/IBM space ...uk.itsc.austin.ibm.com	uk.itsc.austin.ibm.com	Critical	10/4/06 5:33:30 PM	/usr/IBM
/usr/IBM space ...uk.itsc.austin.ibm.com	uk.itsc.austin.ibm.com	Critical	10/4/06 5:32:30 PM	/usr/IBM
/usr/IBM space ...uk.itsc.austin.ibm.com	uk.itsc.austin.ibm.com	Critical	10/4/06 5:31:30 PM	/usr/IBM
/usr/IBM space ...uk.itsc.austin.ibm.com	uk.itsc.austin.ibm.com	Critical	10/4/06 5:30:30 PM	/usr/IBM
/usr/IBM space ...uk.itsc.austin.ibm.com	uk.itsc.austin.ibm.com	Critical	10/4/06 5:29:30 PM	/usr/IBM
/usr/IBM space ...uk.itsc.austin.ibm.com	uk.itsc.austin.ibm.com	Critical	10/4/06 5:28:30 PM	/usr/IBM
/usr/IBM space ...uk.itsc.austin.ibm.com	uk.itsc.austin.ibm.com	Critical	10/4/06 5:27:30 PM	/usr/IBM
/usr/IBM space ...uk.itsc.austin.ibm.com	uk.itsc.austin.ibm.com	Critical	10/4/06 5:26:30 PM	/usr/IBM
/usr/IBM space ...uk.itsc.austin.ibm.com	uk.itsc.austin.ibm.com	Critical	10/4/06 5:25:30 PM	/usr/IBM
/usr/IBM space ...uk.itsc.austin.ibm.com	uk.itsc.austin.ibm.com	Critical	10/4/06 5:24:30 PM	/usr/IBM
/usr/IBM space ...uk.itsc.austin.ibm.com	uk.itsc.austin.ibm.com	Critical	10/4/06 5:23:30 PM	/usr/IBM
/usr/IBM space ...uk.itsc.austin.ibm.com	uk.itsc.austin.ibm.com	Critical	10/4/06 5:22:30 PM	/usr/IBM
/usr/IBM space ...uk.itsc.austin.ibm.com	uk.itsc.austin.ibm.com	Critical	10/4/06 5:21:30 PM	/usr/IBM
/usr/IBM space ...uk.itsc.austin.ibm.com	uk.itsc.austin.ibm.com	Critical	10/4/06 5:20:30 PM	/usr/IBM
/usr/IBM space ...uk.itsc.austin.ibm.com	uk.itsc.austin.ibm.com	Critical	10/4/06 5:19:30 PM	/usr/IBM
/usr/IBM space ...uk.itsc.austin.ibm.com	uk.itsc.austin.ibm.com	Critical	10/4/06 5:18:30 PM	/usr/IBM
/usr/IBM space ...uk.itsc.austin.ibm.com	uk.itsc.austin.ibm.com	Critical	10/4/06 5:17:30 PM	/usr/IBM
/usr/IBM space ...uk.itsc.austin.ibm.com	uk.itsc.austin.ibm.com	Critical	10/4/06 5:16:30 PM	/usr/IBM
/usr/IBM space ...uk.itsc.austin.ibm.com	uk.itsc.austin.ibm.com	Critical	10/4/06 5:15:30 PM	/usr/IBM
/usr/IBM space ...uk.itsc.austin.ibm.com	uk.itsc.austin.ibm.com	Critical	10/4/06 5:14:30 PM	/usr/IBM
/usr/IBM space ...uk.itsc.austin.ibm.com	uk.itsc.austin.ibm.com	Critical	10/4/06 5:13:30 PM	/usr/IBM
/usr/IBM space ...uk.itsc.austin.ibm.com	uk.itsc.austin.ibm.com	Critical	10/4/06 5:12:30 PM	/usr/IBM
/usr/IBM space ...uk.itsc.austin.ibm.com	uk.itsc.austin.ibm.com	Critical	10/4/06 5:11:30 PM	/usr/IBM
/usr/IBM space ...uk.itsc.austin.ibm.com	uk.itsc.austin.ibm.com	Critical	10/4/06 5:10:30 PM	/usr/IBM
/usr/IBM space ...uk.itsc.austin.ibm.com	uk.itsc.austin.ibm.com	Critical	10/4/06 5:09:30 PM	/usr/IBM

Figure 3-38 Sample RCST d-mail for /usr/IBM space used

Setting up more complex monitoring for WebSphere Application Server

You may want to set up a more complex scenario (for example, to check whether a WebSphere Application Server is still alive by using a scheduled “ping” to the server itself).

The following example uses the ApacheBench (AB) simple tool for quickly generating HTTP GET and POST requests. It is part of the Apache HTTP Server and the IBM HTTP Server powered by Apache. It is automatically installed with IBM HTTP server and the executable can be found in <IHS_HOME>\bin\ab.

As a very simple example, we use the PingJSP command, delivered with Trade 61, to perform the keep alive ping. Note that most of the basic steps involved are as described in “Setting up monitoring for the WebSphere filesystem” on page 112. However, for this example we need to introduce a sensor because we do not monitor a standard AIX resource like a filesystem.

1. Create the sensor command.
2. Define the Sensor.
3. Define the RMC condition.
4. Define the RMC response.
5. Link the condition with the response.
6. Start the monitor.
7. Start the execution.

In this section, we use these steps to demonstrate how to define an RMC monitor to control a WebSphere Application by using a scheduled command to access the HTTP site.

1. Create the sensor command.

For our example, we created a sample Korn shell script to execute the AB and return the exit code, which should be zero (0) if successful. The file named PingTrade.ksh has been made available to all systems in the /exports/systemfiles/startup directory. Example 3-33 displays this script.

Example 3-33 Sample PingTrade.ksh script

```
#!/bin/ksh
HOSTTOPING=pl.itsc.austin.ibm.com
#
/usr/IBM/HTTPServer/bin/ab -S -d -c 5 -n 5
"http://${HOSTTOPING}:9080/trade/PingJsp.jsp" 1>/dev/null 2>/dev/null
RC=$?
#Should be zero in case connection was successful
echo "Int32=${RC}"
```

The following sensor attributes can be set by `sensor_command`: `Float32`, `Float64`, `Int32`, `Int64`, `Quantum`, `String`, `Uint32`, `Uint64`. We decided to use the attribute `Int32`. The returned variable `Int32` will be used later to create the conditional evaluation during monitor execution.

2. Define the sensor.

We issued the **`mksensor`** command to create a sensor named `PingTrade` with the Korn shell script as the sensor resource:

```
mksensor PingTrade /exports/systemfiles/startup/PingTrade.ksh
```

We checked the sensor and status by using the **`lssensor`** command. Example 3-34 displays the output.

Example 3-34 Output of `lssensor` for `mksensor`

```
#lssensor
PingTrade
ProbeSensor
ErrorLogSensor
#lssensor PingTrade
Name = PingTrade
ActivePeerDomain =
Command = /exports/systemfiles/startup/PingTrade.ksh
ConfigChanged = 0
ControlFlags = 0
Description =
ErrorExitValue = 1
ExitValue = 0
Float32 = 0
Float64 = 0
Int32 = 0
Int64 = 0
NodeNameList = {uk.itsc.austin.ibm.com}
RefreshInterval = 60
SD = [,0,0,0,0,0,0]
SavedData =
String =
Uint32 = 0
Uint64 = 0
UserName = root
```

3. Define the RMC condition.

We issued the following command to create the “PTNode Trade alive” condition:

```
mkcondition -r IBM.Sensor -e "Int32 != 0" -d "Trade keep alive Ping"
"PTNode Trade alive"
```

The response condition is set to fire if the value for Int32 is not equal to zero (0). We checked the condition and status by using the **lscondition** command. Example 3-35 displays the output.

Example 3-35 lscondition output for mkcondition on sensor

```
#lscondition "PTNode Trade alive"
```

Displaying condition information:

```
condition 1:
```

```
    Name           = "PTNode Trade alive"
    MonitorStatus   = "Not monitored"
    ResourceClass    = "IBM.Sensor"
    EventExpression  = "Int32 != 0"
    EventDescription = "Trade keep alive Ping"
    RearmExpression  = ""
    RearmDescription = ""
    SelectionString  = ""
    Severity         = "i"
    NodeNames        = {}
    MgtScope         = "1"
```

```
#lscondition -a |grep "PTNode"
```

```
"PTNode Trade alive"           "Not monitored"
```

4. Define the RMC response.

We used the **mkresponse** command to create the response “PTNode Trade alive” that has an action named “E-mail root” associated with it:

```
mkresponse -n "E-mail root" -s "/usr/sbin/rsct/bin/notifievent root"
-e b "PTNode Trade alive"
```

5. Link the condition with the response.

We issued the **mkcondresp** command to link the condition and the response:

```
mkcondresp "PTNode Trade alive" "PTNode Trade alive"
```

6. Start the monitor.

We started monitoring by issuing the **startcondresp** command:

```
startcondresp "PTNode Trade alive" "PTNode Trade alive"
```

We verified that the monitor was operational by using the **lsaudrec** command. Example 3-36 shows that monitoring of the condition has been started.

Example 3-36 Check execution using command lsaudrec for sensor

```
# lsaudrec
10/05/06 16:55:40      ERRM Info      Monitoring of condition PTNode
Trade alive is started successfully.
```

7. Start the execution.

After the condition was met, an e-mail was initiated to the defined user (as shown in Example 3-32 on page 116). An entry in the RCST Event log (as shown in Figure 3-38 on page 117) was written until the problem was resolved.

In addition, you can check the Current Attribute Settings of the sensor by using the **lssensor** command, as shown in Figure 3-37 on page 116.

Example 3-37 Changed status of sensor after triggered event

```
#lssensor PingTrade
Name = PingTrade
ActivePeerDomain =
Command = /exports/systemfiles/startup/PingTrade.ksh
ConfigChanged = 0
ControlFlags = 0
Description =
ErrorExitValue = 1
ExitValue = 0
Float32 = 0
Float64 = 0
Int32 = 79
Int64 = 0
NodeNameList = {uk.itsc.austin.ibm.com}
RefreshInterval = 60
```

The value for the Sensor Environment Variable Int32 was now set to 79, based on the return code of the Ping shell script. The Response Condition was set to fire if the value for Int32 was not equal to zero (0). Subsequently, the Response Event was triggered and recorded as shown in Figure 3-39 on page 122.

A more complex implementation would be to add more steps to the response which perform more realistic operations on the node, rather than having a simple e-mail action triggered.

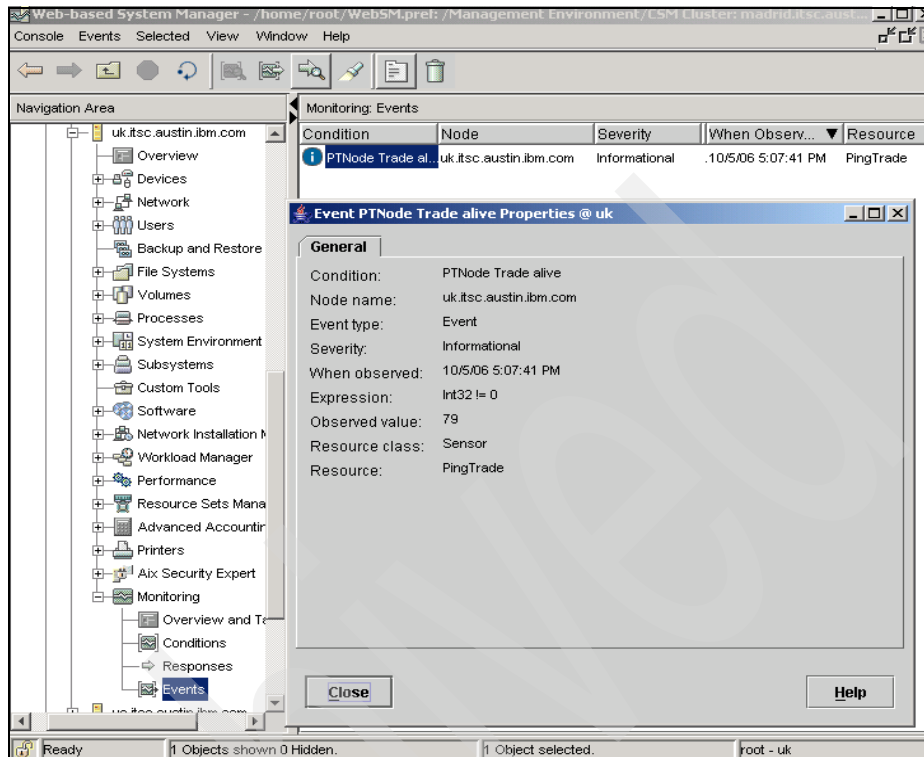


Figure 3-39 WebCSM Event View for PingTrade

Summary of CSM/RSCT monitoring possibilities

As shown, CSM RCT monitoring can be used to define conditions to monitor in the cell and responses to execute when the condition is true. This is especially significant when considering the capability to change LPAR resources using event-triggered scripts.

Note the following considerations when monitoring in a CSM cell:

- Log files

A monitor can be created to archive any system, WebSphere, or user-defined application log files from each application server node to the management server when they reach a certain size. This effectively creates a ring log system.

- File system usage

If an application makes heavy use of a file system, or if you want to protect against core dumps consuming disk space, you can create a monitor to clean up the file system when it reaches a certain utilization threshold.

► Node availability

You can create a monitor that uses the command **dping**. The **dping** command pings the specified servers. This command can be used to retrieve node status before CSM is fully installed, or when you suspect a problem with RMC and its heartbeating.

► Process availability

You can create monitors to monitor the status of critical processes in the cell. Conditions such as a process terminating or using extraneous memory can be remedied with a preconfigured response.

► Centralized logging with the audit log

Many business applications require logging for audit purposes. The CSM audit log can be used by a user-defined business application as a centralized logging facility. Because it is designed to be efficient and is available on all CSM-managed application server nodes in the cell, it is a powerful alternative to custom application log development.

► Monitoring as an application event framework

The CSM monitoring infrastructure can be used as an event framework for deployed applications in the cell. Conditions can be defined that, when true, will execute a change request:

- For resource allocation to the LPAR, like processor allocation
- For application requests to change the number of thread pools used within WebSphere

For example, consider these possibilities:

- Using the default monitored RSCT condition "Page space out rate" and "Page space in rate" to set a threshold to prevent performance degradation due to a high amount of paging. The threshold could be, for example, more than 500 page faults per second.

After the threshold is reached, it would trigger a script that would initiate a dynamic LPAR change request to the HMC to add more memory to the LPAR.

- Using the default monitored RSCT condition "Processors user time" to set a threshold to prevent performance degradation due high load on the CPU. The threshold could be, for example, that the average time all processors are executing in user mode is at least 70% of the time.

After the threshold is reached it would trigger a script that would initiate a dynamic LPAR change request to the HMC to add more processor capacity to the LPAR.

- Using the default monitored RSCT condition "Processors user time" to set a threshold to prevent unused CPU capacity being allocated. The threshold could be, for example, the average time that all processors are executing in user mode is at least 25% of the time.

After the threshold is reached, it would trigger a script that would initiate a dynamic LPAR change request to the HMC to remove processor capacity from the LPAR.

These are only high level examples to demonstrate the power of RSCT monitoring and dynamic LPAR operation.

Performance Monitoring with CSM

Using the WebSM CSM interface, you can even run Performance Monitoring on nodes. After you open a node view as described in "Accessing a node through CSM WebSM access" on page 104, you can select and start one of the available monitors as shown in Figure 3-40.

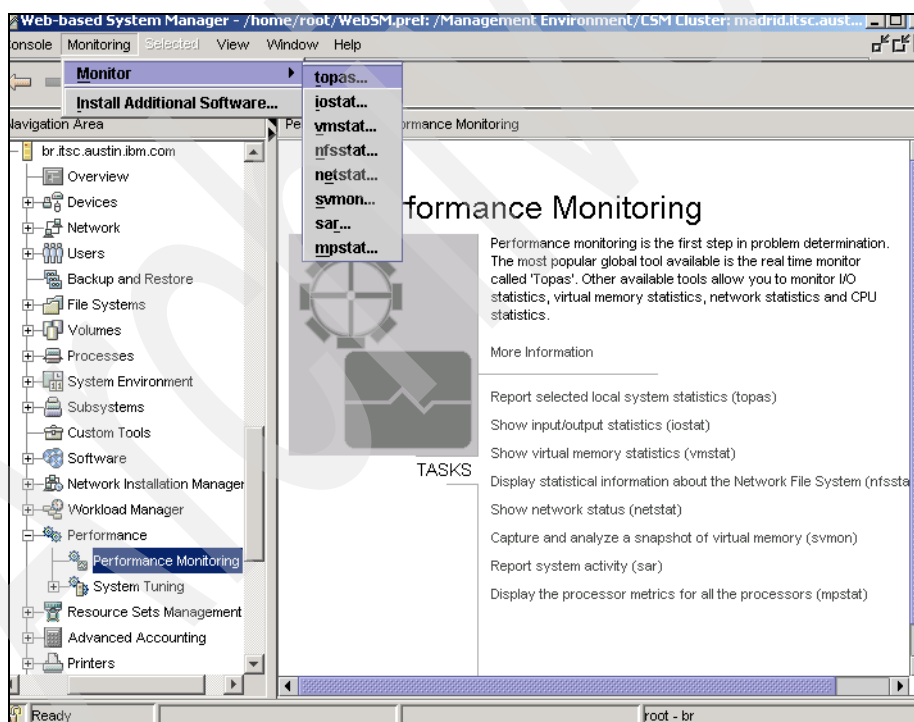


Figure 3-40 The topas command executed from CSM on partition br

Figure 3-41 on page 125 shows a display taken during actual execution of the topas monitors.

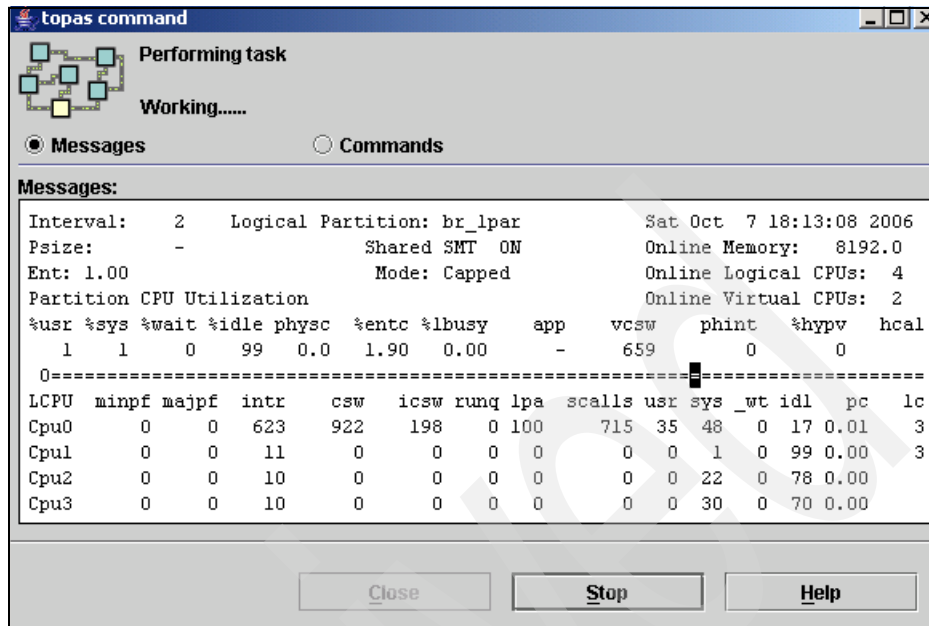


Figure 3-41 Output of topas command executed from CSM on partition br

3.2.4 Using provisioning at the hardware level

This sections discusses hardware provisioning types.

Dynamic LPAR

Dynamic LPAR is the ability to add or remove resources to a partition while that partition is running. Dynamic LPAR can be controlled from the HMC GUI or through scripts. The scripts are administered with the **drmgr** command and can be written in any programming language. Alphaworks provide a dynamic LPAR Tool Set for pSeries; refer to the following address for more information:

<http://www.alphaworks.ibm.com/tech/dlpar>

Micro-partitioning

Micro-partitioning is the ability to allocate fractions of a CPU to an LPAR. Each LPAR must have a minimum of 0.1 CPUs. CPU resource can be allocated in intervals of 0.01 CPUs.

Partitions using micro-partitioning technology are referred to as *shared processor partitions* (SPLPARs). An LPAR must be created as either a dedicated or shared processor partition, and you cannot mix shared and dedicated processors in one partition. However, you can create a shared processor partition with, for

example, an entitlement of 2.25 CPUs. All micro-partitions use CPUs from a single, shared pool.

Capacity on Demand (CoD)

CoD provisions CPUs and memory. It enables you to activate (and, depending on the CoD option purchased, deactivate) unused CPUs and memory in a machine. These can then be allocated as required, and are charged for on a usage basis. The activation process can be either manual or automatic (known as reserve CoD), and permanent or temporary.

Where CoD is used to meet short-term peaks in processing demand, the minimum charge is one “processor day,” which provides one CPU for 24 hours. Multiple CPUs can be provisioned in this way.

Unallocated CPUs and memory, whether active or inactive, are also used to replace failing parts. This process is transparent to the operating systems running on the failing hardware.

3.2.5 Using the provisioning features

This chapter only highlights some provisioning features. For more complete information about this topic, refer to *Partitioning Implementations for IBM eServer p5 Servers*, SG24-7039, and to *PowerVM Virtualization on IBM System p: Introduction and Configuration Fourth Edition*, SG24-7940.

AIX configuration

This chapter demonstrates methods of AIX system administration and configuration management that can be useful when you set up a standardized operating system image for deployment.

4.1 Asynchronous I/O capabilities for sockets and files

As documented in *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392, using AIX Asynchronous I/O capabilities is recommended. The device support is contained in the AIX base operating system fileset bos.iocp.rte. Verify whether the device support is already installed by running the following command:

```
lsllpp -l bos.iocp.rte
```

If this fileset is installed, you receive output similar to that shown in Example 4-1 on page 128. (Note that in AIX 6.1, Asynchronous I/O is installed by default.)

Example 4-1 Checking the iocp fileset

#lsllpp -l bos.iocp.rte			
Fileset	Level	State	Description

Path: /usr/lib/objrepos			
bos.iocp.rte	5.3.0.50	COMMITTED	I/O Completion Ports
API			
Path: /etc/objrepos			
bos.iocp.rte	5.3.0.50	COMMITTED	I/O Completion Ports
API			
#			

If this fileset is *not* installed, you will receive notification that is similar to the following message:

```
lsllpp: Fileset bos.iocp.rte not installed.
```

The bos.iocp.rte fileset is provided on the AIX installation media. Install this fileset as shown in Example 4-2 on page 129 and reboot the system before proceeding. The reboot is needed to create the iocp device within the operating system kernel.

In the example, we installed the fileset using the AIX generic installer command **geninstall** using the NIM lpp_source directory which we directly mounted.

```
#geninstall -d /export/lpp_source/lpp_source53ML05/installp/ppc/ -I
acqX bos.iocp.rte
+-----+
Pre-installation Verification...
+-----+
Verifying selections...done
Verifying requisites...done
Results...

SUCSESSES
-----
Filesets listed in this section passed pre-installation verification
and will be installed.

Selected Filesets
-----
bos.iocp.rte 5.3.0.50 I/O Completion Ports API

<< End of Success Section >>

FILESET STATISTICS
-----
1 Selected to be installed, of which:
1 Passed pre-installation verification ----
1 Total to be installed

+-----+
Installing Software...
+-----+

installp: APPLYING software for:
bos.iocp.rte 5.3.0.50

. . . . . << Copyright notice for bos.iocp >> . . . . .
Licensed Materials - Property of IBM

5765G0300
(C) Copyright International Business Machines Corp. 1999, 2006.

All rights reserved.
US Government Users Restricted Rights - Use, duplication or disclosure
restricted by GSA ADP Schedule Contract with IBM Corp.
```

. << End of copyright notice for bos.iocp >>.
Finished processing all filesets. (Total time: 11 secs).

Summaries:			
Installation Summary -----			
Name	Level	Part	Event
Result			

bos.iocp.rte	5.3.0.50	USR	APPLY
SUCCESS			
bos.iocp.rte	5.3.0.50	ROOT	APPLY
SUCCESS			
#			

After the system is running again, make sure that the iocp device has already been created before enabling the device. Example 4-3 on page 130 illustrates checking the device status using the command **lsdev**.

Example 4-3 Checking the iocp device

```
#lsdev |grep iocp
iocp0      Defined          I/O Completion Ports
#
```

As shown in Example 4-3 on page 130, the iocp0 device is in the Defined state. The state can be changed using the smit fastpath **iocp -> Change/Show Characteristics of I/O Completion Ports**, or by using the command **chdev** as shown in Example 4-4 on page 130.

Example 4-4 Change state settings for iocp subsystem

```
#chdev -l iocp0 -P -a autoconfig='available'
iocp0 changed
```

The changed status will become active after a reboot. But the status in the running system can be changed by using the smit fastpath **iocp -> Configure Defined I/O Completion Ports**, or by using the command **mkdev -l iocp0**.

Example 4-5 on page 131 illustrates changing the status of the iocp device, checking that the status is now Available, and checking that the device is in the same status after a reboot using command **lsattr -El iocp0**.

Example 4-5 Change state for iocp subsystem in the running system

```
#mkdev -l iocp0
iocp0 Available
#lsdev |grep iocp
iocp0 Available I/O Completion Ports
#lsattr -El iocp0
autoconfig available STATE to be configured at system restart True
```

To complete the setup of Asynchronous I/O capabilities, you also need to have the AIO device enabled. The device support is contained in the `bos.rte.aio` fileset, which is automatically installed with the operating system installation. Make sure that the aio device is in the Available state by running the `lsdev` command as shown in Example 4-6 on page 131.

Example 4-6 Checking the aio device

```
#lsdev |grep aio
aio0 Defined Asynchronous I/O (Legacy)
posix_aio0 Defined Posix Asynchronous I/O
#
```

If the device is in the Defined state, change the state for the next reboot of the server by using the `smit fastpath aio -> Change / Show Characteristics of Asynchronous I/O`, or by using the `chdev` command; see Example 4-7.

Example 4-7 Change state of aio subsystem

```
#chdev -l aio0 -P -a minservers="10" -a maxservers="100" -a
maxreqs="100000" -a autoconfig="available" -a fastpath="enable"
aio0 changed
#
```

Subsequent to changing the state of the running system, we used the command `mkdev -l aio0` to change the current state of the device from Defined to Available.

At this point, we had completed the setup of the Asynchronous I/O capabilities in the running system as well for any subsequent reboot of the system.

4.2 AIX Release Content List package information

This AIX Release Content List package `bos.content_list` contains a database that is used by the `/usr/sbin/which_fileset` command. This command is very useful

if you want to know which fileset a specific command belongs to. The output will be the full path of the file, the fileset it belongs to, and the current released level of that file, as shown in Example 4-8 on page 132.

Example 4-8 Output of which_filesset command

```
#which_filesset /usr/bin/emstat
/usr/bin/alstat -> /usr/bin/emstat      bos.perf.tools 5.3.0.0
/usr/bin/emstat      bos.perf.tools 5.3.0.0
#
```

If the command you are looking for is not installed on the system which_filesset returns, you receive the following message: No match found for:

4.3 AIX base operating system samples

The AIX bos.adt.samples fileset contains sample scripts and information if you need to automate DLPAR Resource allocation by using the **drmgr** command, as shown in Example 4-9 on page 132. After the fileset is installed on the system you can access the files, including file /usr/samples/dr/scripts/README, which contains details about the samples. For more information about this topic, refer to the following Tech Note:

<http://www.redbooks.ibm.com/abstracts/tips0121.html>

Example 4-9 lspp sample for bos.adt.samples fileset

```
[0:root@pl:]/usr/samples/dr/scripts # lspp -L bos.adt.samples
Fileset Level State Type Description
-----
bos.adt.samples      5.3.0.40   C    F    Base Operating System
Samples
#lspp -f bos.adt.samples |grep dr
/usr/samples/dr/scripts/sysadmin_mv_rsets.c

/usr/samples/dr/scripts/IBM_XYZ_fail_dr_example.sh
                        /usr/samples/cdrom/cdromt.c
                        /usr/samples/dr/scripts/IBM_XYZ_fail_dr_2.sh
                        /usr/samples/cdrom
                        /usr/samples/dr
                        /usr/samples/dr/scripts/IBM_template.c

/usr/samples/dr/scripts/sysadmin_fail_cpu_add.sh
                        /usr/samples/dr/scripts/IBM_template.pl
                        /usr/samples/dr/scripts/IBM_template.sh
```



```
/usr/samples/cdrom/cdromb.c
/usr/samples/cdrom/cdromdd.h
/usr/samples/dr/scripts/README

/usr/samples/dr/scripts/IBM_XYZ_move_app_example.sh
/usr/samples/dr/scripts
```

4.4 AIX-specific startup and runtime settings

To check for AIX-specific conditions during boot, refer to the bootlog files located in directory `/var/adm/ras/`. For this reason, we changed the size of the bootlog files from 4096 bytes to 64000 bytes. We also changed the verbosity to 3 by using the `alog` command as shown:

```
alog -C -t boot -f /var/adm/ras/bootlog -s 64000 -w 3
alog -C -t boot -f /var/adm/ras/conslog -s 64000 -w 3
```

To allow users to fork additional processes (and because this AIX setting affects DB2 performance), we increased the maximum number of processes per user. We used the `chdev` command to set the `maxuproc` value to 200.

```
chdev -l sys0 -a maxuproc='200'
```

4.5 AIX-specific Java environment settings

The following tables list the set of recommended settings for the AIX Java environment.¹

¹ Taken from:

http://publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp?topic=/com.ibm.aix.prftungd/doc/prftungd/java_tuning_aix.htm

Table 4-1 lists the environment settings.

Table 4-1 Recommended environment settings for Java

Environment variable ^a	Recommended value
AIXTHREAD_SCOPE ^b	S
AIXTHREAD_MUTEX_DEBUG	OFF
AIXTHERAD_COND_DEBUG	OFF
AIXTHREAD_RWLOCK_DEBUG	OFF
SPINLOOPTIME ^c	500

a. For Java 5, the **java** command automatically sets these settings (except SPINLOOPTIME). These settings are only necessary when using JNI™ to invoke Java (meaning when a non-Java application executes Java code or a Java program).

b. In AIX 5.2, Java runs with system-wide contention scope threads.

Each Java thread is mapped one-to-one to an AIX kernel thread

Calls to `java.lang.Thread.setPriority()` have no effect.

In AIX 5.3 and later, programs can set the priority of system contention scope threads Java 5.0 exploits this feature, but Java1.4.2 does not.

Calls to `java.lang.Thread.setPriority()` will change the priority.

c. SPINLOOPTIME controls the number of times the system will retry a busy lock before yielding to another process. The default value is 40. This should be increased to 500 or higher because a busy lock retry is inexpensive compared to the alternative. Use the **tprof** command to determine if the `check_lock` routine has high CPU usage. If it does, you can increase the value even more.

Table 4-2 lists the process settings.

Table 4-2 Recommended process settings for Java

Process limits	Setting
data area, in number of K bytes.	<code>ulimit -d unlimited</code>
stack size, in number of K bytes	<code>ulimit -s unlimited</code>
physical memory, in number of K bytes	<code>ulimit -m unlimited</code>
file descriptors a process may have	<code>ulimit -n unlimited</code>

Example 4-10 shows the contents of a startup file (`rc.was`) we used to set parameters.

Example 4-10 *rc.was* environment settings

```
#Check and set the required WebSphere process environment within file
rc.was
```

```

#
for LIMITV in d \
              s \
              m \
              n
do
LIMIT=~ulimit -${LIMITV} `
if [ "${LIMIT}" != "unlimited" ]
then
    ulimit "-${LIMITV}" unlimited
fi
done
#
export AIXTHREAD_SCOPE=S
export AIXTHREAD_MUTEX_DEBUG=OFF
export AIXTHREAD_COND_DEBUG=OFF
export AIXTHREAD_RWLOCK_DEBUG=OFF
export SPINLOOPTIME=500

```

4.6 AIX TCP/IP network settings

This section provides basic information about operating system tuning parameters. For more details about the recommended values listed in Table 4-3 on page 135, refer to the chapter “Tuning AIX systems” in *IBM WebSphere InfoCenter*, which is available at the following location:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.webSphere.base.doc/info/aes/ae/tprf_tuneaix.html

Table 4-3 Default tuning options

Name	Initial value	Recommended value
tcp_timewait	1	1
tcp_keepidle	14400	600
tcp_keepinit	150	40
tcp_keepintvl	150	10

There are many more TCP/IP settings that can be evaluated and tuned. In the next section we focus on the TCP/IP buffer size parameter.

TCP/IP buffer size

In TCP/IP, packets that are sent must be acknowledged. If a packet is not acknowledged, then the sender must be able to resend the lost packet.

Operating systems implement this task by creating a buffer holding all packets sent. A packet is removed from the buffer after an acknowledgement packet (ACK) is received.

A simple formula defines the theoretical limit of a connection:

$$\text{max transfer rate} = \text{TCP/IP buffer space} / \text{packet round trip time}$$

As an example, assume a buffer with 16 Kbyte, a network with unlimited bandwidth but with 100 ms round trip time, and 1 Kbyte packets. The sender fills the buffer immediately with 16 packets. Some 50 ms later, the receiver gets the data and sends the acknowledgement packets. These ACKs are received by the sender 100 ms after the first transmission. The buffer is cleared and the filled again. So 16 Kbyte packets can be transmitted within 100 ms. In this case, the maximum transmit rate is 160 Kbyte/sec.

The tunable value in this calculation is the TCP/IP buffer space. The round trip time is difficult to change. This is a theoretical limit. In the real world there are other factors, like TCP/IP slow start, bandwidth limit or lost packets, that can influence the transfer rate.

Note the following points regarding the buffer space:

- ▶ The default buffer size is 8 KB. The maximum size is 8 MB (8096 KB).
- ▶ Increasing the buffer size improves throughput, reduces the occurrence of flowcontrol, and reduces CPU cost.
- ▶ This is useful when transferring large data sets between WebSphere Application Server and a database.
- ▶ If the buffer size is too large, paging can increase.

We recommend setting the window size to 512 Kbyte. To do this, you must enable RFC 1323 support and enable selective acknowledgement (SACK).

Setting network tuning parameter values in AIX

Use the **no** command to configure network tuning parameters. The **no** command sets or displays current or next boot values for network tuning parameters. This command can also make permanent changes or defer changes until the next reboot.

Whether the command sets or displays a parameter is determined by the accompanying flag. The **-o** flag performs both actions; it can either display the value of a parameter, or set a new value for a parameter.

We used the **no -a** command to list all settings and the following commands to set the values:

- ▶ no -o tcp_keepidle=600
- ▶ no -o tcp_keepintvl=10
- ▶ no -o tcp_keepinit=40
- ▶ no -o sack=1
- ▶ no -o rfc1323=1
- ▶ no -o tcp_sendspace=524176
- ▶ no -o tcp_recvspace=524176
- ▶ no -o sb_max=1048352
- ▶ no -o tcp_mssdflt=1448

When you use the **no** command to change parameters, dynamic parameters are changed in memory and the change is in effect only until the next system boot. At that point, all parameters are set to their reboot settings.

To make dynamic parameter changes permanent, use the **-r** or **-p** options of the **no** command to set the options in the nextboot file. Reboot parameter options require a system reboot to take effect. For more information about this topic, refer to *AIX 5L Practical Performance Tools and Tuning Guide*, SG24-6478, and to the IBM System p and AIX Information Center at the following address:

<http://publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp>

4.7 WebSphere Server start and stop

To be able to start automatically when the host partition is started, we created a common startup file (rc.was) as shown in Example 4-11. This file also contains the recommended initial AIX Java tuning settings as described in 4.5, “AIX-specific Java environment settings” on page 133. Including the tuning settings in the startup file ensures that these specific settings are only used within the WebSphere process environment.

We placed this file in a common directory, /exports/systemfiles/startup, for general usage on all our WebSphere installations. Placing them in a common location enabled us to use the **dsh** command to start and stop the WebSphere server from a centralized control point on the CSM Management Server.

```
#!/bin/ksh
HOSTNAME=$(hostname)
#
ulimit -d unlimited
ulimit -s unlimited
ulimit -m unlimited
ulimit -n unlimited
export AIXTHREAD_SCOPE=S
export AIXTHREAD_MUTEX_DEBUG=OFF
export AIXTHREAD_COND_DEBUG=OFF
export AIXTHREAD_RWLOCK_DEBUG=OFF
export SPINLOOPTIME=500
#
APPSRVS="server1"
#
case "${HOSTNAME}" in
    "uk.itsc.austin.ibm.com") APPSRVN="TradeAppSrv"

    /usr/IBM/WebSphere/AppServer/bin/startManager.sh -replacelog
    ;;
    "br.itsc.austin.ibm.com") APPSRVN="AppSrv"

    /usr/IBM/WebSphere/AppServer/bin/startManager.sh -replacelog

    /usr/IBM/WebSphere/AppServer/profiles/${APPSRVN}/bin/startNode.sh
    -replacelog

    /usr/IBM/WebSphere/AppServer/profiles/${APPSRVN}/bin/startServer.sh
    ${APPSRVS} -replacelog
    ;;
    "pt.itsc.austin.ibm.com") APPSRVN="TradeAppSrv"

    /usr/IBM/WebSphere/AppServer/bin/startManager.sh -replacelog

    /usr/IBM/WebSphere/AppServer/profiles/${APPSRVN}/bin/startNode.sh
    -replacelog

    /usr/IBM/WebSphere/AppServer/profiles/${APPSRVN}/bin/startServer.sh
    ${APPSRVS} -replacelog
    ;;
    "pl.itsc.austin.ibm.com") APPSRVN="TradeAppSrv"
```

```
/usr/IBM/WebSphere/AppServer/profiles/${APPSRVN}/bin/startNode.sh
-replacelog

/usr/IBM/WebSphere/AppServer/profiles/${APPSRVN}/bin/startServer.sh
${APPSRVS} -replacelog
esac
```

By replacing the command **startServer** or **startNode** with **stopServer** or **stopNode** in the rc.was file, as shown in Example 4-11, we could create a new script file called stop.was.

In Example 4-12, we show sample session output using the **dsh** command to execute the stop.was script from the CSM Management Server madrid.itsc.austin.ibm.com on partition pt.itsc.austin.ibm.com. In addition, we used the option **-s** for streaming mode.

Example 4-12 Stopping the WebSphere server on partition pt

```
[0:@MADRID:]:#dsh -s -n pt "/exports/systemfiles/startup/stop.was"
pt.itsc.austin.ibm.com: ADMU0116I: Tool information is being logged in
file
pt.itsc.austin.ibm.com:
/usr/IBM/WebSphere/AppServer/profiles/Dmgr/logs/dmgr/stopServer.log
pt.itsc.austin.ibm.com: ADMU0128I: Starting tool with the Dmgr profile
pt.itsc.austin.ibm.com: ADMU3100I: Reading configuration for server:
dmgr
pt.itsc.austin.ibm.com: ADMU3201I: Server stop request issued. Waiting
for stop status.
pt.itsc.austin.ibm.com: ADMU4000I: Server dmgr stop completed.
pt.itsc.austin.ibm.com: ADMU0116I: Tool information is being logged in
file
pt.itsc.austin.ibm.com:
/usr/IBM/WebSphere/AppServer/profiles/TradeAppSrv/logs/nodeagent/stopSe
rver.log
pt.itsc.austin.ibm.com: ADMU0128I: Starting tool with the TradeAppSrv
profile
pt.itsc.austin.ibm.com: ADMU3100I: Reading configuration for server:
nodeagent
pt.itsc.austin.ibm.com: ADMU3201I: Server stop request issued. Waiting
for stop status.
pt.itsc.austin.ibm.com: ADMU4000I: Server nodeagent stop completed.
pt.itsc.austin.ibm.com: ADMU0116I: Tool information is being logged in
file
```

```
pt.itsc.austin.ibm.com:
/usr/IBM/WebSphere/AppServer/profiles/TradeAppSrv/logs/server1/stopServer.log
pt.itsc.austin.ibm.com: ADMU0128I: Starting tool with the TradeAppSrv profile
pt.itsc.austin.ibm.com: ADMU3100I: Reading configuration for server: server1
pt.itsc.austin.ibm.com: ADMU3201I: Server stop request issued. Waiting for stop status.
pt.itsc.austin.ibm.com: ADMU4000I: Server server1 stop completed.
```

Although the examples shown were used on only one node, the **dsh** command is capable of running concurrently on several nodes. So we could have shut down all nodes as well as started all servers in one command sequence, as shown in Example 4-13.

Example 4-13 Concurrent stop of WebSphere server on multiple partitions

```
#dsh -n pt,uk,pl "/exports/systemfiles/startup/stop.was"
uk.itsc.austin.ibm.com: ADMU4000I: Server dmgr stop completed.

pl.itsc.austin.ibm.com: ADMU4000I: Server server1 stop completed.
pt.itsc.austin.ibm.com: ADMU4000I: Server server1 stop completed.
#dsh -n pt,uk,pl "/exports/systemfiles/startup/rc.was"
```

WebSphere Application Server on AIX: under the hood

This chapter explains the structure and workings of Java Virtual Machine (JVM) and WebSphere Application Server, and describes how they interact with the underlying AIX platform. Understanding these topics will help you plan your deployment and tune your environment. Many IBM products run inside the application server environment, so the principles outlined here can be applied more generally to these, as well.

First the chapter explains why WebSphere Application Server is used and how it came about in its current 6.1 Version, followed by an examination of how it is laid out on disk (that is, the directory structures) after it is installed. Next, the high level WebSphere Application Server architecture is described and put into context within the wider operating system. Finally, an “under the hood” view examines how it all works, and looks at the three key “layer” that are essential to the current WebSphere Application Server 6.1 runtime environment:

- ▶ The IBM J9 Java Virtual Machine (JVM) Runtime
- ▶ The Eclipse 3.1.2 / OSGi Runtime and Component Model
- ▶ The underlying WebSphere Application Server runtime components extended by the WebSphere Application Server services

Note that discussion of components higher up the stack (such as the Service Integration Bus and its default messaging provider engine, the Portlet Container, the Session Initiation Protocol Container, or any of the Web services support components) is beyond the scope of this publication, because they have no direct interaction with the underlying AIX and WebSphere platform boundary focused on in this book.

5.1 Java and J2EE

We assume that the primary audience for this book consists of system architects and administrators responsible for designing, deploying, and managing WebSphere Application Server on System p and AIX. In the following sections we provide background on Java and J2EE for that audience.

Having standards like Java allowed client application vendors to write code without too much concern as to the platform it would be run on. The J2EE standard allows server-side code to be produced that can target a myriad of high-end enterprise platforms with testing only required (in theory) to certify for that platform. This reduces costs for the enterprise software vendor and allows customers to choose the platform that suits their needs without being driven by application software requirements.

So, how does this fact relate to WebSphere Application Server? WebSphere Application Server is the premier IBM implementation of the J2EE standard specification. J2EE is the standard and WebSphere Application Server is an implementation of that standard that provides all the services and interfaces that meet that specification.

AIX and WebSphere Application Server system administrators and application developers should understand that there are a number of things going on in the Java runtime environment inside the WebSphere Application Server process container, as well as a number of things going on outside the WebSphere Application Server process container that are interdependent. For instance, failover of functionality can occur both inside the JVM to another JVM, or outside the JVM to start another JVM, or something the JVM is communicating with (that is, a DBMS instance, MQ queue manager, and so on).

5.1.1 Java

Java started out as a language (Oak) developed for embedded system usage, with an objective of providing a portable environment that could be used on a range of projected set top boxes. It was essentially derived from C++, providing powerful object-oriented features such as classes to combine code and data, and single inheritance, while simplifying greatly to remove the more complex constructs such as pointers and multiple inheritance that led to reliability issues.

C and C++ programs had been the mainstay of systems and applications programmers, but the reliability of a C or C++ program cannot be verified for correctness because the pointers contain memory addresses and not data or objects that can be checked until runtime. So, to overcome the verification issue and provide a base for portability, the “virtual machine” concept was utilized.

Virtual machines provide a consistent machine architecture interface to the software it runs, while having a lower level that is ported transparently to different hardware. Java also introduced the system level concept of threads into the language itself, to allow multiple “concurrent threads of execution” to be specified within the language rather than forcing system-specific extensions to be used that would limit portability.

Java soon proved its use in the arena of the Internet as the power behind applets that provided the initial dynamicism for previously static Web sites. Applets run in the client inside the Web browser Java sandbox, which limits their usefulness because they have to be downloaded and can lead to large amounts of data traveling between the client and the server when anything powerful is needed.

At the time, C and scripted CGI programs were being used to provide server side dynamicism for Web sites by allowing the Web server to run external programs and receive the response using environmental variables and the system environment; however, these approaches had scalability issues which limited their usefulness.

Into this historical context, server-side Java was introduced to allow a server-side Java virtual machine to map Web requests to threads running code to a particular standard called the “servlet”. This was the beginning of Java2 Extended Edition, or J2EE (which is being renamed to JavaEE in the next release).

5.1.2 J2EE

J2EE is a standard. It is a specification for the application programmer interfaces and classes that an application server vendor must implement to adhere to the standard for enterprise-level server code. The emphasis is on “enterprise” in that the standard is designed for scalability, with two “containers” that manage threads and services for the Web and scalable business components, and numerous enterprise-level services.

J2EE consists of the following main components:

- ▶ A Web container to run multiple threads aimed at supporting Web users communicating with the application server using HTTP or HTTPS
 - The Web container hosts and runs servlets; that is, Java code that takes a HTTP request, performs any necessary data retrieval and manipulation, and then returns a response. Servlet code is Java code that contains HTML output statements, or calls something else (Java Server Page, or JSP™) to produce the HTML output.

- The Web container also “hosts” JSPs. This is effectively the inverse of the previous point because JSPs are aimed at a different type of developer and can be thought of as HTML code that includes embedded Java.

However, JSPs are compiled to a special case of a servlet at deployment or runtime, often using a component called Jasper. Jasper is the reference engine from Tomcat that Sun™ uses to verify application compliance with the JSP standard. Frameworks such as Struts and Java Server Faces (JSF) are built upon the JSP standard in combination with a servlet for control.

- Modern application servers, such as the WebSphere Application Server, may also use the Web container to host Web services supporting Apache Axis or JAX-WS with SOAP/XML over HTTP, or even portlets to produce page snippets designed to be hosted inside a page in a separate portal server. Under the covers, these are special cases of servlets.
- The J2EE reference engine for the Web container is Apache Tomcat, so if a J2EE Web application does not run in the WebSphere Application Server, then the Tomcat reference engine can be used to verify that the application is compliant with the standards.

- ▶ An Enterprise Java Bean (EJB) container that runs scalable business components

A *bean* in Java is essentially a name for a component. Note, however, that a Java Bean and an Enterprise Java Bean are *not* the same thing, as explained here:

- A Java Bean is a component that uses specific naming for interfaces for getters and setters to hold logic and data in a form that allows runtime “reflection” to allow its use.
- An Enterprise Java Bean uses specific interfaces to allow it to be hosted within the EJB container. There are three main types of Enterprise Java Bean (EJB):
 - The *session bean* is designed to host business logic rather than data. It offers a verb-based interface and set of methods, that is, “doXXXXX”. Session beans should be stateless, but a special type of session bean, the *stateful session bean*, can be used with hand-crafted database calls or other code to host state or data.
 - The *entity bean* is designed to represent a single instance of an entity, that is, a row in a database. Methods are available to determine which row the bean instance should represent, but an instance is essentially designed to be the host for data for a particular entity.

Entity beans tend to have verb interfaces to determine the entity, but they map to nouns in their names and their fields. For example, “finder” methods are used to map the single instance from the pool of

instances in a repository or database, and then fields are accessed by name.

Note the following points:

- In bean-managed persistence (BMP) entity beans, the developer is responsible for coding access to the raw data used to populate the entity bean.
- In container-managed persistence (CMP) entity beans, the developer defines the type of data required using the SQL-like EJB Query Language (EJB QL) and lets the infrastructure sort out the most appropriate access mechanism to retrieve it.
- In Message Driven Beans (MDBs), a scalable listener to Java Message Service (JMS) message queues is produced.
- The EJB Container was originally designed to use the CORBA IIOP mechanism to access the beans remotely, and similar techniques for development of CORBA objects was required. Today, performance is better with local interfaces, but these remote interfaces led to the architecture.

Essentially, the developer defines Java interfaces and some code and deployment descriptor configuration. Then, at deployment time, the container or deployment tool compile these to produce the code that actually runs. Just to emphasize: the code that actually runs as the EJB instances is *derived* from what was developed, but is generated at deployment time by the infrastructure.

- IBM was a contributor to the EJB container implementation, because many of the scalability ideas are based on or similar to those in CICS®.
- A useful standard application will use a remote interface for the session beans to allow access to the code running in the container, usually via a coarse-grained façade, and will then have fine-grained local interfaces for the other EJBs.

J2EE also includes other standards that provide the underpinning for the rest of the application server:

- ▶ Java Message Service (JMS) provides message queuing interfaces similar to those in WebSphere MQ, which IBM uses for the underlying implementation or transport (depending on configuration).
- ▶ Java Database Connectivity (JDBC™) provides a standard interface for database access for SQL and stored procedures that are then mapped to the underlying technology by a specific driver for that DBMS implementation (that is, DB2, Derby, Oracle®, SQL Server® and so on).

- ▶ Java Naming and Directory Interface (JNDI) provides a means for “looking up” objects by a given name to map to its provided object implementation. This is often combined with Lightweight Directory Access Protocol (LDAP) servers.
- ▶ Java Connector Architecture (JCA), also known as J2C, is a mechanism for mapping to earlier or non-Java enterprise information sources. It is often used as the means by which JDBC requests are passed by the EJB container to a particular DBMS for maximum scalability, or to access CICS, IMS™, or applications such as SAP®. It requires a connector to map the interface to the Java world.

5.2 Application Server 6.1 on AIX

The release of WebSphere Application Server 6 introduces significant changes and enhancements from earlier editions of WebSphere Application Server. Despite the move to supporting the J2EE 1.4 standard, the most important change for the platform administrator is the ability to replace the external WebSphere MQ processes to support the Java Message Service (JMS) with an implementation that is *internal* to the WebSphere Application Server JVM process. The following sections discuss this change, as well as other topics related to the implementation of WebSphere Application Server on AIX.

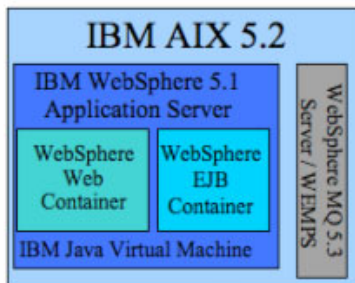
5.2.1 WebSphere Application Server and the JMS Messaging Engine

In WebSphere Application Server 5.1 and earlier, JMS was supported through an external implementation of WebSphere MQ Server using either WebSphere Embedded Messaging Publish and Subscribe (WEMPS) or the full WebSphere MQ Server implementation. Almost the same binaries were used for either setup, but only the full implementation could interact with external remote WebSphere MQ environments.

This setup led to a standard WebSphere Application Server implementation that required JMS to have a minimum of 10 processes running (9 minimum for WEMPS or WebSphere MQ), and environment variable setup requirements (that is, `AIXTHREAD_SCOPE=S` and `AIXTHREAD_MNRATIO=1:1`) to allow thread synchronization to function properly between these processes on AIX. This difference is illustrated in Figure 5-1 on page 148.

WAS and the JMS Messaging Engine

• WebSphere Application Server 5.1



- Uses WebSphere MQ 5.3 Server or the WEMPS cut-down version
- This requires a lot (10 or more) separate processes
- WAS JVM is 1 process, MQ Support is 9 or more so a lot of context switching goes on between processes at the OS level for messaging, which affects performance.
- This can cause problems on some OS's, i.e. AIX, where special settings are needed for threading.

• WebSphere Application Server 6.X

- WEMPS is replaced by WebSphere Platform Messaging / JMS Messaging Engine
- Fully Supports integration with WebSphere MQ
- Runs INSIDE the JVM so no context switching occurs
- Threading all handled seamlessly within the JVM
- Despite being all in Java rather than platform native code, performance should be better under load

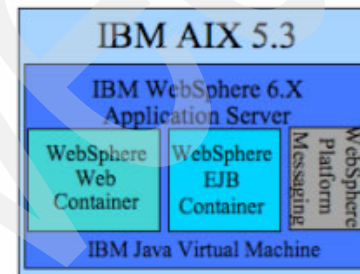


Figure 5-1 WebSphere Application Server and the JMS Messaging Engine: from Version 5.1 to Version 6.x

With the release of WebSphere Application Server 6, the Service Integration Bus and its embedded messaging implementation runs entirely inside the Java process. This change eliminates any external synchronization issues. This WebSphere MQ implementation in the Java space satisfies the requirements of the JMS specification, and can interoperate with remote WebSphere MQ environments.

In addition to the messaging implementation changes and numerous other tuning and structural improvements, WebSphere Application Server 6 introduced an internal high availability manager called HAManager to manage high availability for a cluster and the failover of Java singleton components from one cluster member to another. This is similar in concept to High Availability Cluster Multi Processing (HACMP), but sits inside the Java process and works with Java artifacts, such as the Service Integration Bus components. Message queuing components are a good example of a singleton because there should be only

one message queuing server actively receiving messages onto a queue at one time if message ordering is to be preserved.

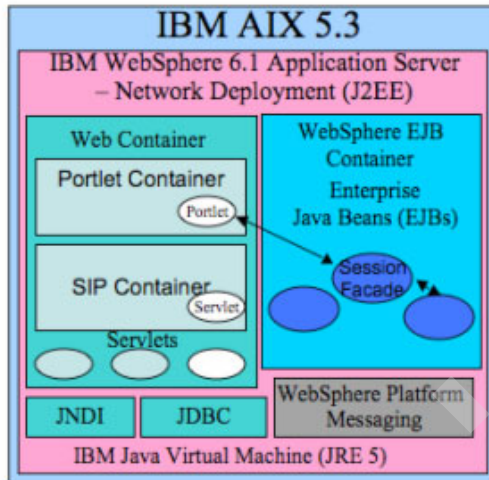
On WebSphere Application Server 6, the earlier Java 1.4 family of Java Virtual Machines was used to support the J2EE 1.4 standard. With WebSphere Application Server 6.1, the Java5 (1.5 JVM) “Tiger” Java Virtual Machine was introduced, but still supporting a J2EE 1.4 standard environment. This improves monitoring of performance inside the JVM through the introduction of the JVMTI standard, greatly improves memory management, makes coding easier and safer with new language features such as “generics”.

With each new release of the IBM Java Virtual Machine and WebSphere Application Server, other IBM-specific new features are added. With the Java5 release of the IBM J9 JVM, additional memory management features for improved garbage collection were added, as well as class sharing to minimize memory heads for multiple WebSphere Application Server processes running inside a single operating system image.

For WebSphere Application Server, the runtime environment was changed so that it is based on an Eclipse implementation of the OSGI framework. This allows a component extension approach to hosting J2EE services and components. A SIP container (for VoIP and multimedia support) and Portlet container were also added to the Web container.

Figure 5-2 on page 150 displays the new features added to WebSphere Application Server 6.1.

WebSphere Application Server 6.1



- Fully J2EE 1.4 compliant, but running on a Java 5 Virtual Machine
- Adds a portlet container supporting the WSRP (Web Services for Remote Portals) and JSR168 standards to allow integration with a distributed portal environment
- Session Initiation Protocol (SIP) container for integration between web apps and VoIP, Multimedia, etc with JSR116 standard compliance
- More support for active-active environments with Network Attached Storage hosting the transaction logs

Figure 5-2 New features introduced in WebSphere Application Server 6.1

5.2.2 Process structure

The Java Virtual Machine (JVM) is coded to run on a real machine, and is therefore not portable between different machine architectures. From the perspective of the code running on it, the Java Virtual Machine is a real machine with its own stack-based architecture, its own instruction set, and its own memory management and I/O mechanisms. The JVM has to map these virtual machine architecture components to the real machine on which it has been implemented. In our case, with WebSphere Application Server 6.1, the Java “instruction set” known as Java byte code instructions is derived from the Java 5 standard and the implementation is the IBM J9 JVM 1.5 for AIX and POWER.

The key thing to understand about Java is the emphasis on “virtual” in the phrase virtual machine. A running instance of the WebSphere Application Server is simply a Java process, although a powerful and complex one. A Java process is

simply a single process like any other, and unless Java-specific extensions have been provided to the operating system that enable it to understand Java classes, it is a “black box” from the perspective of the operating system. So, the application code and containers running inside that Java process must be scheduled and controlled by the virtual machine rather than the operating system.

Unless you are using Java extensions to `trc` and `tprof`, the AIX operating system tools cannot manage or observe the Java code running as part of WebSphere Application Server or inside WebSphere Application Server. If you run a `ps` you will see a “Java” process with a complex command line for running WebSphere Application Server, but will not see anything about J2EE container usage or applications running within WebSphere Application Server.

5.2.3 WebSphere Application Server on AIX - disk layout

The WebSphere Application Server runtime installs on AIX under `/usr/IBM/WebSphere/AppServer` by default. This is where the base code resides.

To run enterprise applications, one or more profiles are created that, by default, are contained in this directory. Applications may include customizations for specific roles that point back to this higher level base code.

The profiles consist of a set of directories that contain scripts that execute the base level WebSphere Application Server runtime components in the higher level directories, and specific XML file customizations to better support the applications that are hosted on that profile. On a WebSphere Application Server Network Deployment environment, for example, a `DMgr` profile is used to support the deployment manager, and other profiles support WebSphere Application Server nodes and specific standalone WebSphere Application Server configurations.

Before moving on to how WebSphere Application Server works, it is useful to know how it is laid out on the file system. An examination of the home directory of the user who installs and maintains WebSphere Application Server will illustrate why it is best to be consistent regarding who performs this role.

The home directory of the installing user ID contains a hidden `.WASRegistry` file, which identifies the installation root for all WebSphere Application Server installations and related products.

The default installation directory for WebSphere Application Server on AIX of `/usr/IBM/WebSphere/AppServer` uses the directories listed in Table 5-1 on page 152. For each separately configured profile, an instance of the top level environment exists under the profile name beneath the profiles directory.

The default installation file system layout is listed in Table 5-1. There are a number of components in Table 5-1 that may need further explanation, most notably the references to the Eclipse and OSGI runtime.

Developers and many administrators are familiar with the concept of the Eclipse IDE and its plug-ins that allow for expansion of the base runtime environment. More details are provided later in this chapter, but as of Version 6.1, WebSphere Application Server itself is essentially composed of server-side Eclipse plug-ins.

Table 5-1 WebSphere Application Server on AIX default installation file system layout

Directory	Description
bin (parent and profile)	This directory contains WebSphere Application Server-specific binaries (that is, shared objects and so on) and management scripts. In the profile version of this directory, the parent scripts are called.
cip	This directory contains the default customized installation package (CIP) config for use with the Eclipse-based installation factory tool.
configuration (parent and profile)	This directory contains the config.ini file used to control the Eclipse/OSGI base configuration and startup. This is a very important file because it controls what gets loaded as part of the OSGI initialization, so it can be used to start components on the JVM before WebSphere Application Server itself does.
deploytool	This directory contains an Eclipse-based tool for application deployment, as well as a complete Eclipse environment.
derby	This directory contains the lightweight Java-based Apache Derby database in embedded and network server versions.
etc (parent and profile)	This directory contains the Virtual Member Manager (VMM/WIM) environment for building a customer user and group management environment for use with federated user repositories; samples for the WS-Security Web services setup; and tmx4j open source JMX™ code that IBM uses for offering JMX management facilities.

Directory	Description
features	This directory contains the configuration for some of the Eclipse “features” used to group dependent Eclipse plug-ins. The subdirectories each contain a feature.xml file that lists plug-ins and versions that have prerequisites and dependencies that should be configured, managed and upgraded together. Features include the base product, the network deployment product, the UDDI server, the Web services gateway, and others. Note that the feature dependencies in most of the feature.xml files refer to version 6.0.0 products except for some of the IBM programming model extensions (PME) and new 6.1.0-specific features (labelled 7.0.0), showing that much of the core code is the same between the two 6.X versions, despite the architectural change.
firststeps (parent and profile)	This directory contains the firststeps application HTML and script files used for easy initial configuration. This makes use of the core WebSphere Application Server runtime library for its work.
installableApps (parent and profile)	This directory contains the EAR, WAR, etc files that form the applications that are available for deployment to the application servers configured as part of the cell managed by this installation.
installedApps (parent and profile)	This directory contains the deployed code from the installableApps that the profile nodes run.
installedConnectors (parent and profile)	This directory contains the resource adapter archive files used by the connectors installed for the application server that uses the J2EE Connector Architecture (JCA or J2C) to interface to external or earlier systems.
installedFilters (at profile level)	This directory contains the filters that the profile nodes run.
java	This directory contains the Java Virtual Machine used by this installation of WebSphere Application Server (in this case, the IBM J9 JVM).
lafiles	This directory contains the license agreement files in multiple languages for the WebSphere Application Server environment.

Directory	Description
lib	<p>This directory contains Java library files (JAR files) that support some of the base WebSphere Application Server product set. They include libraries to support some DBMSs, utility libraries, libraries to support the service integration bus and mail server interaction, some of the associated WebSphere Application Server tools (the log analyzer) and base libraries to support WebSphere Application Server startup. Java libraries for WebSphere MQ are in a subdirectory.</p> <p>One file of particular interest is that used to identify the Eclipse platform in use (.eclipseproduct), which reveals that an Eclipse 3.1.2 product is in use:</p> <pre>name=Eclipse Platform id=org.eclipse.platform version=3.1.2</pre> <p>This file goes along with the startup.jar file that contains the org.eclipse.core.launcher code for running the base Eclipse environment that can host some of the Eclipse-based tools and the bootstrap.jar file that performs similar functions. This is the same code as can be found on the Eclipse Web site for developing an AIX Eclipse application. The startup.jar and bootstrap.jar files start the Eclipse runtime by obtaining configuration information from properties files, config files and the classpath, and then bootstrapping Eclipse/OSGI with the correct information.</p>
logs	<p>This directory contains the log files for the parent environment (usually, only the creation log files for the profiles created using the manageprofile.sh script). In the profiles themselves, the directory contains the start and stop logs, and the error logs (that is, stdout and so on).</p>
optionalLibraries	<p>This contains directories for optional Java jar files that can be deployed to the environment to provide extra functionality, such as Jython, Struts, and the JSF-Portlet bridge. This contains Java code rather than shared objects or achieve library files.</p>

Directory	Description
plugins	<p>Like all Eclipse applications, this contains the Eclipse runtime environment plug-ins and much of the core WebSphere Application Server runtime environment itself that is now implemented as Eclipse plug-ins and OSGI bundles. Pay particular attention to:</p> <ul style="list-style-type: none"> ▶ "org.eclipse.osgi_3.1.2.jar - Core of the OSGI platform ▶ "com.ibm.cds_1.0.0.jar - Provides the class loader integration with the shared classes environment ▶ "com.ibm.ws.runtime_6.1.0.jar - Provides the base WebSphere Application Server runtime Java environment ▶ "org.eclipse.jdt.core_3.1.2.jar and com.ibm.jdt.core_6.1.0.jar that provide the Java runtime integration between WebSphere Application Server, Eclipse and the JVM.
profiles	<p>This directory contains the configuration and scripts specific to a configured runtime set of nodes and instances created by a particular profileTemplate. Common default profiles are the Dmgr01 profile created by the "dmgr" profileTemplate and the AppSrv01 profile created by the "managed" profileTemplate.</p> <p>Under these profile directories, pay particular attention to the logs directory containing log files for the instances, the config/cells directory containing the Web server plugin-cfg.xml file and the application server server.xml file for each instance, and the bin directory containing management scripts.</p>
profileTemplates	<p>This directory contains XML configuration files and ant scripts used by the manageprofiles.sh script for setting up the appropriate WebSphere Application Server environments for a particular role. Which directories of these files are available to allow which profile roles to be set up depends on the chosen WebSphere Application Server installation type purchased (that is, standalone, network deployment and so on).</p>
properties (parent and profile)	<p>This directory contains the property files and XML configuration files for key tools and WebSphere Application Server components for the base environment.</p>
runtimes	<p>This directory contains Java jar files for the client administration tools and the Web services thin client.</p>
samples (parent and profile)	<p>This directory contains runtime applications and Java source for sample applications provided with WebSphere Application Server. These are good resources for Java developers to learn how J2EE applications should be written.</p>

Directory	Description
sar2war_tool	This directory contains xml and an xsl script to convert a WebSphere Application Server Session Initiation Protocol (SIP) servlet archive (sar) for use by the WebSphere Application Server SIP container into a standard J2EE Web archive (war) file. Note: This sar file standard is not the same as that in use by JBoss for service archives.
Scheduler	This directory contains the DDL for the database implementations for various DBMSs that support the WebSphere Application Server Scheduler service that can invoke WebSphere Application Server-hosted Java code to a schedule.
systemApps	This directory contains the Java EAR files for system-level applications such as the event service or the scheduler calendar application.
temp (parent and profile)	This directory contains a dummy entry for the main installation—but for profile temp directories, it contains relevant property files and checkpoint serialized object files for the instance.
tmsStorage (at profile level)	This directory contains a persisted copy of the runtime deployment tasks for the deployment manager.
tranlog (at profile level)	This directory contains, at the bottom of the directory tree for the nodes and instances, a tranlog directory and a partnerlog directory that are together used for XA distributed transaction management. These should be stored on a shared storage device like a NAS (with NFSv4) for the highest availability so that other instances in the cluster can access them. This set of files contains details of in-doubt transactions.
UDDIReg	This directory contains the implementation of the IBM UDDI registry for Web services to support lookup of Web services within a private enterprise.
uninstall	This directory contains the Java code and XML configuration files to support uninstalling WebSphere Application Server.
universalDriver	This directory contains an architecture-neutral DB2 driver to support JDBC access to DB2 on multiple platforms.
util	This directory contains jacl and shell scripts for installing and configuring WebSphere Application Server and its features.

Directory	Description
web	<p>This directory contains HTML documentation for WebSphere Application Server, including some of the Rational® Rose® models used to create WebSphere Application Server itself, the application client configuration tool, the JMX management interfaces, and the programming model for entity beans.</p> <p>If you want to know how WebSphere Application Server works, and make WebSphere Application Server perform better on your platform at the lowest level, examine the documentation contained here. The documentation for the docs directory and all classes directory in particular contains significant information.</p>
wstemp (at profile level)	<p>This directory contains HTML documentation for WebSphere Application Server, including some of the Rational Rose models used to create WebSphere Application Server itself, the application client configuration tool, the JMX management interfaces, and the programming model for entity beans.</p> <p>If you want to know how WebSphere Application Server works, and make WebSphere Application Server perform better on your platform at the lowest level, examine the documentation contained here. The documentation for the docs directory and all classes directory in particular contains significant information.</p>

5.2.4 Application server configuration

The WebSphere Application Server configuration is stored in XML files under the application server PROFILE/config directory. The same core files are used at the application server instance level in each of the WebSphere Application Server editions. There are some differences in the configuration at the higher level between the high availability WebSphere Application Server editions and the standalone edition.

A WebSphere Application Server unit of management is called a *cell*, which in a high availability environment such as that from the WebSphere Application Server - Network Deployment edition, is made up of a deployment manager, node agents on every node (an operating system instance), and a number of application server instances that may be in zero or more clusters. Even in a standalone environment, the concept of a cell can be seen to underlie the configuration.

The node agents on the node are responsible for keeping the configuration of all WebSphere Application Server instances on the node in sync with the

configuration held on the deployment manager, both of which are WebSphere Application Server instances in their own right that have specialized configurations and purposes.

Core to all of the installations of WebSphere Application Server are the server.xml file and the serverindex.xml file for the given instance. However, other configuration files are also important. Table 5-2 lists and describes each of these XML configuration files. All of these XML files are updated when either the administration system console is used or when scripting via wsadmin.

Important: The XML configuration files listed in Table 5-2 should only be updated via administration system console operations or when scripting via wsadmin.

Table 5-2 WebSphere Application Server XML configuration files

File	Purpose
admin-authz.xml	This file contains the server instance role assignments and mappings from users and groups of users to these internal roles.
cell.xml	This file contains a small amount of information as to whether the cell is STANDALONE or DISTRIBUTED.
coregroup.xml	This file contains the configuration of core groups for high availability fast failover of singletons by the WebSphere Application Server HAManager for the Network Deployment edition and above.
cluster.xml	This file contains the high availability cluster topology configuration for the cluster node to enable it to identify its place in the cluster with the other cluster members.
hamanagerservice.xml	This file contains the timeout, thread pool configuration, and buffer management configuration for each core group managed by the HAManager in the WebSphere Application Server ND environments and above.
installed-channels.xml	This file contains some details of the installed channels for the environment. This is little used and often contains no real configuration.
libraries.xml	This file contains the configuration information for optional libraries to be deployed into a particular application environment, such as those for the Java Server Faces (JSF) standard that adds additional functionality to the Web container.

File	Purpose
multibroker.xml	This file contains the reliable multicast messaging configuration for the data replication for high availability.
namestore.xml	This file contains the data persistence binding configuration.
naming-authz.xml	This file contains the security configuration of roles and user and group mappings to those roles for naming operations.
nodes.xml	This file contains the node topology information for the cell so the cell can identify its place in the cell.
pmi-config.xml	This file contains the PMI performance metrics module configuration to identify what is being monitored and how WebSphere Application Server is monitoring it (that is, what level of monitoring, what components within that PMI configuration item are being monitored, and what Java level components are involved in that monitoring or reporting on that monitoring).
perftuners.xml	This file contains the rule configuration for the WebSphere Application Server built-in application-level and system-level monitoring advisors. WebSphere Application Server can simply, with minimal overhead, be configured to monitor itself according to a set of rules to allow it to advise on any misconfiguration or improvements that can be made for a particular usage and set of applications. The mapping of the rules to the Java code that validates the environment against those rules and reports against them is configured in this file.
plugin-cfg.xml	This file contains the configuration file for the Web server plug-in that configures the Web servers for application mappings, weightings for use of a particular cluster member, and general cluster load balancing and high availability configuration to be used in routing decisions when forwarding a request from a Web server to an application server instance.
pmirm.xml	This file contains the PMI request metrics configuration for what subsystems metrics are available for and to be monitored for, and any filtering that is to be applied.
resources-cei.xml	This file contains the Common Event Infrastructure (CEI) provider resource configuration for Common Business Events (CBE) that applications can emit and subscribe to. This is seen to best effect in the WebSphere Business Integration Server Foundation and WebSphere Process Server families. This was introduced with WebSphere Application Server 5.1.

File	Purpose
resources-pme502.xml	This file contains the WebSphere Application Server 5.02 programming model extensions (PME) configuration such as the Dynamic Caching (dynacache). This was introduced with WebSphere Application Server 5.0.2.
resources-pme.xml	This file contains the WebSphere Application Server programming model extensions for the more general WebSphere Application Server 5.0 products and above, such as the Object Pool Provider, the Work Manager Provider, and Scheduler Provider.
resources.xml	This file contains the configuration information for the J2C/JCA resource adapter providers that provide the underpinnings for EJB entity bean access to a particular DBMS, messaging provider configuration, and EIS integration product WebSphere Application Server configuration such as for integration with CICS or IMS. This file also contains the configuration for how these resource adapters are administered from within the WebSphere Application Server/J2EE environment.
security.xml	This file contains the WebSphere Application Server, J2EE, and JCA security configuration for the server. Configuration for single signon, trust association interceptors, CSIV2 IIOP security, and so on all goes in here. If there is an issue with integrating WebSphere Application Server security with platform security, such as for Kerberos, examine the contents of this file. It also contains some user ID and password data in some configurations, access information to key stores, key information, and SSL configuration. If WebSphere Application Server is to be locked down securely, this file should be understood. It is one of the most important configuration files in any WebSphere Application Server environment.
server.xml	This file contains the majority of the configuration for a particular WebSphere Application Server instance. It contains mappings, connection and buffer configuration, and component enablement configuration for the communications channels for the core WebSphere Application Server containers (that is, the Web Container, the EJB Container, the Service Integration Bus, and so on), thread pool configuration, and the majority of configuration items that are not covered elsewhere. This is the most important configuration file for a WebSphere Application Server instance.

File	Purpose
serverindex.xml	This file contains the port listener configuration and mappings from WebSphere Application Server variable names configured elsewhere within the WebSphere Application Server configuration to actual values. This is important for a platform and network administrator to be aware of because the majority of these ports must be open at the platform level through any firewalls for WebSphere Application Server to function correctly. This is one of the most important configuration files in a WebSphere Application Server environment.
sib-service.xml	This file contains the Service Integration Bus (SIB) service configuration.
systemapps.xml	This file contains the configuration for the instance to enable system-level applications such as the file transfer service or management EJBs.
variables.xml	This file contains the substitution mappings from internal configuration variables (like environment variables) to their actual values for the specific WebSphere Application Server environment; for example, the WAS_CELL_NAME variable might be mapped to myhostnameWASCellName01.
virtualhosts.xml	This file contains the virtual host configuration and mime types for that host for each of the virtual hosts configured for the environment. This will at the very least contain the configuration and mime types for default_host.
ws-security.xml	This file contains the security key and certificate store configuration, the login type and configuration, encryption configuration, and other security configuration for the WS-Security standard for Web services for WebSphere Application Server.

Many of the configuration files in Table 5-2 on page 158 occur in multiple places in the profile directory hierarchy. This allows a particular application server instance to override the configuration of the cluster and node it is part of, and to override some of the configuration at the cell level if needed.

The structure of the directories is shown in Example 5-1.

Example 5-1 Configuration file directory structure

```
cells
  cell1
    applications
```

```

        application1.ear
        application2.ear
clusters
    cluster1
    cluster2
coregroups
    DefaultCoreGroup
nodegroups
    DefaultNodeGroup
nodes
    node1
        servers
            server1
            server2
    node2
        servers
            server1
            server2
cell2
    applications
        application1.ear
        application2.ear
    clusters
        cluster1
        cluster2
    coregroups
        DefaultCoreGroup
    nodegroups
        DefaultNodeGroup
    nodes
        node1
            servers
                server1
                server2
        node2
            servers
                server1
                server2

```

The directory structure shown in Example 5-1 on page 161 is only fully implemented and meaningful in WebSphere Application Server ND and above, although parts of it still apply to the standalone edition of WebSphere Application Server.

The applications subdirectory contains the code and J2EE configuration for the applications configured at the level at which the applications directory is found. Because J2EE applications are EAR files which themselves contain Java code and WAR, EJB-JAR and JAR files, these directories are named after the EAR file for the given application and thus, have a .ear file extension¹.

Other non-XML files are used for some configuration. For example, some security configuration for Lightweight Third Party Authentication (LTPA) tokens and keystores is found in ltpa.jceks, key.p12, and trust.p12, although this is not directly configurable

5.2.5 Key server configuration files

At a minimum, understand the WebSphere Application Server configuration settings used in three core files:

- ▶ server.xml (see 5.2.6, “server.xml” on page 163)
- ▶ serverindex.xml (see 5.2.7, “serverindex.xml” on page 175)
- ▶ security.xml (see 5.2.8, “security.xml” on page 178)

These configuration files are the most important for the WebSphere Application Server environment. Local copies are held on the given node, and the deployment manager maintains its own master copy that it replicates out to the nodes it manages in the WebSphere Application Server - Network Deployment edition and above.

AIX and WebSphere Application Server administrators should be familiar with the core information in these files because it affects the platform configuration as well as the general application server configuration, and in many ways can be considered the WebSphere Application Server equivalent of the .conf files and the Object Data Manager (ODM) files used for configuration of the AIX environment.

5.2.6 server.xml

The server.xml file contains the main configuration for a WebSphere Application Server instance. It contains configuration information for the communications subsystems underlying the core containers at the Java level (that is, which communications channels are configured, what their maximum number of connections are, what their timeout and buffer sizes are and so on). It also contains more general configuration, such as the JCA and JDBC configuration and user settings that are used to access external resources outside of WebSphere Application Server.

¹ For more details about this configuration, refer to the J2EE standard documentation at: <http://java.sun.com>

This file is crucial to the environment and presents a security risk if available for reading by anyone other than the WebSphere Application Server administration staff, because the information it contains can be used to access those external resources.

Although `server.xml` can be directly edited, we advise against this and recommend configuring the configuration beans via `wsadmin` or by use of the administration system console. The `server.xml` configuration file examples in this section (Example 5-2 through Example 5-10 on page 174) were taken from a WebSphere Application Server ND environment configured with clusters inside a cell, on a node called `spock44p` in the domain `RnREnterprise.dyndns.org`, which explains some of the names used within the environment.

The identifier for the JVM process within the cell and cluster environment is configured as attributes within the outermost tag; the `process:Server` tag. As shown in Example 5-2, this tag identifies the XML Metadata Interchange (XMI) files used to apply against the XML files used by this server instance, as well as the instance name and cluster name for that server.

Example 5-2 server.xml: process:Server tag

```
<process:Server xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
...
xmlns:threadpoolmanager="http://www.ibm.com/websphere/appserver/schemas/6.0/threadpoolmanager.xmi"
...
xmi:id="Server_1172257092257" name="spock44pMember01" clusterName="spock44pCluster01">
```

As shown in Example 5-3, within the `process:Server` tag there are numerous other configuration items, with the configuration of individual WebSphere Application Server services being a major part.

Example 5-3 server.xml: process:Server tag detail

```
...
<stateManagement xmi:id="StateManageable_1172257094317" initialState="START"/>
  <statisticsProvider xmi:id="StatisticsProvider_1172257094318"
specification="com.ibm.orb=enabled"/>
```



```

    <services xmi:type="pmiservice:PMIService" xmi:id="PMIService_1172257094318"
enable="true" initialSpecLevel="" statisticSet="extended"
synchronizedUpdate="false"/>

...

    <services xmi:type="traceservice:TraceService" xmi:id="TraceService_1172257094319"
enable="true" startupTraceSpecification="*=info" traceOutputType="SPECIFIED_FILE"
traceFormat="BASIC" memoryBufferSize="8">
        <traceLog xmi:id="TraceLog_1172257094319" fileName="${SERVER_LOG_ROOT}/trace.log"
rolloverSize="20" maxNumberOfBackupFiles="1"/>
    </services>

...

    <services xmi:type="loggingervice.ras:RASLoggingService"
xmi:id="RASLoggingService_1172257094320" enable="true" messageFilterLevel="NONE"
enableCorrelationId="true">
        <serviceLog xmi:id="ServiceLog_1172257094320" name="${LOG_ROOT}/activity.log"
size="2" enabled="true"/>
    </services>

...

```

In Example 5-3 on page 164, note the use of variables such as `${SERVER_LOG_ROOT}` that are mapped from values in the `variables.xml` file when applied.

One of the most important services to WebSphere Application Server is the `orb:ObjectRequestBroker` service, shown in Example 5-4. The J2EE standard, particularly with the EJB container, is based on the generic OMG CORBA standard so the ORB configuration is key to EJB behavior. The configuration of this section includes thread pool information, mappings of Java code implementations to interfaces, interceptor configurations for manipulation of the communications and operations chains with the ORB, and any additional plugins that are to be loaded by the environment. Pay particular attention to the `com.ibm.ws.orb.transport.UseMultiHome` setting that controls whether the EJBs can be remotely accessed over all network interfaces, or just the one named in the `com.ibm.CORBA.LocalHost` setting.

Example 5-4 server.xml: orb.ObjectRequestBroker detail

```

<services xmi:type="orb:ObjectRequestBroker"
xmi:id="ObjectRequestBroker_1172257094320" enable="true" requestTimeout="180"

```

```

requestRetriesCount="1" requestRetriesDelay="0" connectionCacheMaximum="240"
connectionCacheMinimum="100" commT
raceEnabled="false" locateRequestTimeout="180" forceTunnel="never"
noLocalCopies="false" useServerThreadPool="false">
    <properties xmi:id="Property_1172257094321"
name="com.ibm.CORBA.enableLocateRequest" value="true"/>
...

    <properties xmi:id="Property_1172257094326"
name="com.ibm.ws.orb.transport.useMultiHome" value="true"/>
    <properties xmi:id="Property_1172257094327"
name="com.ibm.websphere.management.registerServerIORWithLSD" value="true"/>
    <interceptors xmi:id="Interceptor_1172257094322"
name="com.ibm.ejs.ras.RasContextSupport"/>
    <interceptors xmi:id="Interceptor_1172257094323"
name="com.ibm.ws.runtime.workloadcontroller.OrbWorkloadRequestInterceptor"/>
    <interceptors xmi:id="Interceptor_1172257094324"
name="com.ibm.ws.Transaction.JTS.TxInterceptorInitializer"/>
    <interceptors xmi:id="Interceptor_1172257094325"
...

    <interceptors xmi:id="Interceptor_1172257094334"
name="com.ibm.debug.olt.ivbtrjrt.OLT_RI"/>
    <plugins xmi:id="ORBPlugin_1172257094323"
name="com.ibm.ws.orbimpl.transport.WSTransport"/>
...

    <plugins xmi:id="ORBPlugin_1172257094328"
name="com.ibm.ws.csi.CORBAORBMethodAccessControl"/>
    <threadPool xmi:id="ThreadPool_1172257094324" minimumSize="10" maximumSize="50"
inactivityTimeout="3500" isGrowable="false" name="ORB.thread.pool"/>
</services>

```

In Example 5-4 on page 165, also pay attention to the `threadPool` element configuration because this controls the number of threads used by the ORB for interfacing to the EJB container, which is limited to 50 threads.

Next within the `server.xml` file is the configuration of the transport channels, shown in Example 5-5 on page 167, which includes the buffer sizes, which thread pools the threads for servicing them come from, and the chain configuration for layering transports on top of transports (that is, SSL on top of

TCP). Note that the channels are split into inbound and outbound channels to better support asynchronous handling of requests and responses within WebSphere Application Server. How this works is explained in the section where we look at the WebSphere Application Server internals and channel configuration. Note that the endPointName attributes refer to settings in serverindex.xml.

Example 5-5 server.xml: transport channel detail

```
<services xmi:type="channelService:TransportChannelService"
xmi:id="TransportChannelService_1172257094324" enable="true">
  <transportChannels xmi:type="channelService.channels:TCPInboundChannel"
xmi:id="TCPInboundChannel_1172257094324" name="TCP_1" endPointName="WC_adminhost"
maxOpenConnections="20000" inactivityTimeout="60" threadPool="ThreadPool_11722
57094325"/>
  <transportChannels xmi:type="channelService.channels:TCPInboundChannel"
xmi:id="TCPInboundChannel_1172257094325" name="TCP_2" endPointName="WC_defaulthost"
maxOpenConnections="20000" inactivityTimeout="60" threadPool="ThreadPool_117
2257094325"/>
  ...

  <transportChannels xmi:type="channelService.channels:TCPInboundChannel"
xmi:id="TCPInboundChannel_1172257094328" name="TCP_5"
endPointName="DCS_UNICAST_ADDRESS" maxOpenConnections="20000" inactivityTimeout="60"
threadPool="ThreadPoo
l_1172257094326"/>
  ...

  <transportChannels xmi:type="channelService.channels:SSLInboundChannel"
xmi:id="SSLInboundChannel_1172257094326" name="SSL_1" discriminationWeight="1"/>
  <transportChannels xmi:type="channelService.channels:SSLInboundChannel"
xmi:id="SSLInboundChannel_1172257094327" name="SSL_2" discriminationWeight="1"/>
  ...

  <transportChannels xmi:type="channelService.channels:HTTPInboundChannel"
xmi:id="HTTPInboundChannel_1172257094327" name="HTTP_1" discriminationWeight="10"
maximumPersistentRequests="100" keepAlive="true" readTimeout="60" writeTimeou
t="60" persistentTimeout="30" enableLogging="false"/>
  <transportChannels xmi:type="channelService.channels:HTTPInboundChannel"
xmi:id="HTTPInboundChannel_1172257094328" name="HTTP_2" discriminationWeight="10"
maximumPersistentRequests="100" keepAlive="true" readTimeout="60" writeTimeou
t="60" persistentTimeout="30" enableLogging="false"/>
  ...
</services>
```

```

        <transportChannels xmi:type="channelService.channels:WebContainerInboundChannel"
xmi:id="WebContainerInboundChannel_1172257094328" name="WCC_1"
discriminationWeight="1" writeBufferSize="32768"/>
        <transportChannels xmi:type="channelService.channels:WebContainerInboundChannel"
xmi:id="WebContainerInboundChannel_1172257094329" name="WCC_2"
discriminationWeight="1" writeBufferSize="32768"/>
...

        <transportChannels xmi:type="channelService.channels:DCSInboundChannel"
xmi:id="DCSInboundChannel_1172257094329" name="DCS_1" discriminationWeight="1"/>
        <transportChannels xmi:type="channelService.channels:DCSInboundChannel"
xmi:id="DCSInboundChannel_1172257094330" name="DCS_2" discriminationWeight="1"/>
...

        <transportChannels xmi:type="channelService.channels:UDPInboundChannel"
xmi:id="UDPInboundChannel_1172257094340" name="UDP_1" discriminationWeight="0"
endPointName="SIP_DEFAULTHOST"/>
        <transportChannels xmi:type="channelService.channels:TCPIInboundChannel"
xmi:id="TCPIInboundChannel_1172257094340" name="SIB_TCP_JFAP"
endPointName="SIB_ENDPOINT_ADDRESS" threadPool="ThreadPool_1172257094341"/>
        <transportChannels xmi:type="channelService.channels:TCPIInboundChannel"
xmi:id="TCPIInboundChannel_1172257094341" name="SIB_TCP_JFAP_SSL"
endPointName="SIB_ENDPOINT_SECURE_ADDRESS" threadPool="ThreadPool_1172257094341"/>
        <transportChannels xmi:type="channelService.channels:TCPIInboundChannel"
xmi:id="TCPIInboundChannel_1172257094342" name="SIB_TCP_MQFAP"
endPointName="SIB_MQ_ENDPOINT_ADDRESS" threadPool="ThreadPool_1172257094341"/>
        <transportChannels xmi:type="channelService.channels:TCPIInboundChannel"
xmi:id="TCPIInboundChannel_1172257094343" name="SIB_TCP_MQFAP_SSL"
endPointName="SIB_MQ_ENDPOINT_SECURE_ADDRESS" threadPool="ThreadPool_1172257094341"/>
...

<transportChannels xmi:type="channelService.channels:TCPOutboundChannel"
xmi:id="TCPOutboundChannel_1172257094343" name="SIB_TCP_JFAP_SSL_OUT"
inactivityTimeout="60" threadPool="ThreadPool_1172257094342"/>
        <transportChannels xmi:type="channelService.channels:TCPOutboundChannel"
xmi:id="TCPOutboundChannel_1172257094344" name="SIB_TCP_JFAP_TUN_OUT"
inactivityTimeout="60" threadPool="ThreadPool_1172257094342"/>
...
<transportChannels xmi:type="channelService.channels:HTTPOutboundChannel"
xmi:id="HTTPOutboundChannel_1172257094344" name="SIB_HTTP_JFAP_TUN_OUT"/>

```

```

    <transportChannels xmi:type="channelService.channels:HTTPOutboundChannel"
xmi:id="HTTPOutboundChannel_1172257094345" name="SIB_HTTP_JFAP_TUN_SSL_OUT"/>
    <transportChannels xmi:type="channelService.channels:HTTPTunnelOutboundChannel"

...

    <chains xmi:id="Chain_1172257094346" name="WCInboundAdmin" enable="true"
transportChannels="TCPInboundChannel_1172257094324 HTTPInboundChannel_1172257094327
WebContainerInboundChannel_1172257094328"/>
    <chains xmi:id="Chain_1172257094360" name="WCInboundDefault" enable="true"
transportChannels="TCPInboundChannel_1172257094325 HTTPInboundChannel_1172257094328
WebContainerInboundChannel_1172257094329"/>

...

<chains xmi:id="Chain_1172257094363" name="DCS" enable="true"
transportChannels="TCPInboundChannel_1172257094328 DCSInboundChannel_1172257094329"/>
    <chains xmi:id="Chain_1172257094364" name="DCS-Secure" enable="true"
transportChannels="TCPInboundChannel_1172257094328 SSLInboundChannel_1172257094328
DCSInboundChannel_1172257094330"/>
    <chains xmi:id="Chain_1172257094365" name="SIPCInboundDefault" enable="true"
transportChannels="TCPInboundChannel_1172257094329 SIPInboundChannel_1172257094329
SIPContainerInboundChannel_1172257094340"/>
    <chains xmi:id="Chain_1172257094366" name="SIPCInboundDefaultSecure"
enable="true" transportChannels="TCPInboundChannel_1172257094330
SSLInboundChannel_1172257094329 SIPInboundChannel_1172257094330
SIPContainerInboundChannel_1172257
094341"/>
    <chains xmi:id="Chain_1172257094367" name="SIPCInboundDefaultUDP" enable="true"
transportChannels="UDPInboundChannel_1172257094340 SIPInboundChannel_1172257094331
SIPContainerInboundChannel_1172257094342"/>
    <chains xmi:id="Chain_1172257094368" name="InboundBasicMessaging" enable="true"
transportChannels="TCPInboundChannel_1172257094340
JFAPInboundChannel_1172257094342"/>
    <chains xmi:id="Chain_1172257094369" name="InboundSecureMessaging" enable="true"
transportChannels="TCPInboundChannel_1172257094341 SSLInboundChannel_1172257094341
JFAPInboundChannel_1172257094343"/>

...

    <chains xmi:id="Chain_1172257094372" name="BootstrapBasicMessaging"
transportChannels="JFAPOutboundChannel_1172257094345
TCPOutboundChannel_1172257094342"/>

```

```

    <chains xmi:id="Chain_1172257094373" name="BootstrapSecureMessaging"
transportChannels="JFAPOutboundChannel_1172257094346 SSLOutboundChannel_1172257094345
TCPOutboundChannel_1172257094343"/>
    <chains xmi:id="Chain_1172257094374" name="BootstrapTunneledMessaging"
transportChannels="JFAPOutboundChannel_1172257094347
HTTPTunnelOutboundChannel_1172257094344 HTTPOutboundChannel_1172257094344
TCPOutboundChannel_1172257094344"/
>
    <chains xmi:id="Chain_1172257094375" name="BootstrapTunneledSecureMessaging"
transportChannels="JFAPOutboundChannel_1172257094348
HTTPTunnelOutboundChannel_1172257094345 HTTPOutboundChannel_1172257094345
SSLOutboundChannel_117225709
4346 TCPOutboundChannel_1172257094345"/>
...

```

One of the most important configuration elements at the service level within the application server is the `threadpoolmanager:ThreadPoolManager` configuration that controls the sizes and management of thread pools for the different containers and components within WebSphere Application Server, as shown in Example 5-6. Note that the largest size that the thread pools can reach is 50 threads and that these are not “growable”. Very careful consideration must be taken with modifying these values directly, because arrays are used in some of the thread pool management code for performance and going above the array size will cause failure.

Example 5-6 server.xml: threadpoolmanager detail

```

<services xmi:type="threadpoolmanager:ThreadPoolManager"
xmi:id="ThreadPoolManager_1172257094364" enable="true">
    <threadPools xmi:id="ThreadPool_1172257094364" minimumSize="10" maximumSize="50"
inactivityTimeout="3500" isGrowable="false" name="ORB.thread.pool"/>
    <threadPools xmi:id="ThreadPool_1172257094365" minimumSize="1" maximumSize="3"
inactivityTimeout="30000" isGrowable="false" name="server.startup" description="This
pool is used by WebSphere during server startup."/>
    <threadPools xmi:id="ThreadPool_1172257094366" minimumSize="5" maximumSize="20"
name="Default"/>
    <threadPools xmi:id="ThreadPool_1172257094325" minimumSize="10" maximumSize="50"
inactivityTimeout="3500" isGrowable="false" name="WebContainer"/>
    <threadPools xmi:id="ThreadPool_1172257094326" minimumSize="5" maximumSize="20"
name="TCPChannel.DCS"/>
    <threadPools xmi:id="ThreadPool_1172257094341" minimumSize="4" maximumSize="50"
name="SIBFAPInboundThreadPool" description="Service integration bus FAP inbound
channel thread pool"/>

```

```
<threadPools xmi:id="ThreadPool_1172257094342" minimumSize="4" maximumSize="50"
name="SIBFAPThreadPool" description="Service integration bus FAP outbound channel
thread pool"/>
</services>
```

The next part of the file includes the names of HTTP logging files and whether HTTP logging is to be enabled, as shown in Example 5-7. Again, note the mapping of variable names from variables.xml.

Example 5-7 server.xml: logging detail

```
<services xmi:type="loggingService.http:HTTPAccessLoggingService"
xmi:id="HTTPAccessLoggingService_1172257094365" enable="false"
enableErrorLogging="true" enableAccessLogging="true">
  <errorLog xmi:id="LogFile_1172257094365"
filePath="${SERVER_LOG_ROOT}/http_error.log" maximumSize="500"/>
  <accessLog xmi:id="LogFile_1172257094366"
filePath="${SERVER_LOG_ROOT}/http_access.log" maximumSize="500"/>
</services>
```

After the services section of the server.xml file, a number of different configuration items are used as shown in Example 5-8. A small section identifies the redirection of the system standard input and standard output files for the process. The rollover control settings for log file cycling and management and the stack trace settings are important for management of size, cycling, and retention of log files at the AIX level, so the AIX administrator should be aware of these settings and manage the external AIX environment accordingly.

Example 5-8 server.xml: StreamRedirect detail

```
<errorStreamRedirect xmi:id="StreamRedirect_1172257094366"
fileName="${SERVER_LOG_ROOT}/SystemErr.log" rolloverType="SIZE"
maxNumberOfBackupFiles="1" rolloverSize="1" baseHour="24" rolloverPeriod="24"
formatWrites="true" messageFormat
Kind="BASIC" suppressWrites="false" suppressStackTrace="false"/>
<outputStreamRedirect xmi:id="StreamRedirect_1172257094367"
fileName="${SERVER_LOG_ROOT}/SystemOut.log" rolloverType="SIZE"
maxNumberOfBackupFiles="1" rolloverSize="1" baseHour="24" rolloverPeriod="24"
formatWrites="true" messageForma
tKind="BASIC" suppressWrites="false" suppressStackTrace="false"/>
```

The next important section is the components section, which controls the individual components of WebSphere Application Server and how they behave

with separate entries for each core component (that is, the Web container, the core application server, the EJB container, and so on), as shown in Example 5-9. These entries include configuration to enable and control individual containers and what components are loaded within them, including thread pool management configuration and mapping of listeners to the containers, giving us component and service configuration within higher level component configuration.

Note that the application server component has the Web container and EJB container components configured within it. This information will be important later when the internals of the runtime plugin.xml file and how the application server starts up are considered.

The information here affects the memory used by the application server; that is, the number of sessions to be maintained, and how I/O is handled by the given component. User ID and password information for the session persistence database are also configured here, so this file must be carefully secured if this information is sensitive for a particular environment.

Example 5-9 server.xml: components section detail

```
<components xmi:type="namingserver:NameServer" xmi:id="NameServer_1172257094366">
  <stateManagement xmi:id="StateManageable_1172257094367" initialState="START"/>
</components>
<components xmi:type="applicationserver:ApplicationServer"
xmi:id="ApplicationServer_1172257094367" applicationClassLoaderPolicy="MULTIPLE">
  <stateManagement xmi:id="StateManageable_1172257094368" initialState="START"/>
  <services xmi:type="applicationserver:TransactionService"
xmi:id="TransactionService_1172257094367" enable="true"
totalTranLifetimeTimeout="120" clientInactivityTimeout="60"
propogatedOrBMTTranLifetimeTimeout="300" httpProxyPrefix="
" httpsProxyPrefix=""/>
  <services xmi:type="applicationserver:DynamicCache"
xmi:id="DynamicCache_1172257094368" enable="true">
    <cacheGroups xmi:id="ExternalCacheGroup_1172257094368" name="EsiInvalidator">
      <members xmi:id="ExternalCacheGroupMember_1172257094368" address="localhost"
adapterBeanName="com.ibm.websphere.servlet.cache.ESIInvalidatorServlet"/>
    </cacheGroups>
  </services>
  <components xmi:type="applicationserver.webcontainer:WebContainer"
xmi:id="WebContainer_1172257094368" enableServletCaching="false"
disablePooling="false">
    <stateManagement xmi:id="StateManageable_1172257094369" initialState="START"/>
    <services xmi:type="applicationserver.webcontainer:SessionManager"
xmi:id="SessionManager_1172257094369" enable="true" enableUrlRewriting="false"
enableCookies="true" enableSSLTracking="false" enableProtocolSwitchRewriting="false"
```



```

    sessionPersistenceMode="DATA_REPLICATION" enableSecurityIntegration="false"
allowSerializedSessionAccess="false" maxWaitTime="5" accessSessionOnTimeout="true">
    <defaultCookieSettings xmi:id="Cookie_1172257094369" domain=""
maximumAge="-1" secure="false"/>
    <sessionDatabasePersistence xmi:id="SessionDatabasePersistence_1172257094369"
datasourceJNDIName="jdbc/Sessions" userId="db2admin" password="{xor}0z1tPjsyNjE="
db2RowSize="ROW_SIZE_4KB" tableSpaceName=""/>
    <tuningParams xmi:id="TuningParams_1172257094384" usingMultiRowSchema="false"
maxInMemorySessionCount="1000" allowOverflow="true" scheduleInvalidation="false"
writeFrequency="TIME_BASED_WRITE" writeInterval="10" writeContents="0
NL_Y_UPDATED_ATTRIBUTES" invalidationTimeout="30">
    <invalidationSchedule xmi:id="InvalidationSchedule_1172257094384"
firstHour="14" secondHour="2"/>
    </tuningParams>
    <sessionDRSPersistence xmi:id="DRSSettings_1172257095445"
messageBrokerDomainName="spock44pCluster01"/>
    </services>
</components>
    <components xmi:type="applicationserver.ejbcontainer:EJBContainer"
xmi:id="EJBContainer_1172257094384" passivationDirectory="${USER_INSTALL_ROOT}/temp"
inactivePoolCleanupInterval="30000">
    <stateManagement xmi:id="StateManageable_1172257094385" initialState="START"/>
    <services
xmi:type="applicationserver.ejbcontainer.messageListener:MessageListenerService"
xmi:id="MessageListenerService_1172257094385">
    <threadPool xmi:id="ThreadPool_1172257094385" minimumSize="10"
maximumSize="50" inactivityTimeout="3500" isGrowable="false"
name="Message.Listener.Pool"/>
    </services>
    <cacheSettings xmi:id="EJBCache_1172257094385" cleanupInterval="3000"
cacheSize="2053"/>
    <timerSettings xmi:id="EJBTimer_1172257094385"
datasourceJNDIName="jdbc/DefaultEJBTimerDataSource" tablePrefix="EJBTIMER_"
pollInterval="300" numAlarmThreads="1"/>
    </components>
    <components xmi:type="portletcontainer:PortletContainer"
xmi:id="PortletContainer_1172257094386"/>
    <components xmi:type="applicationserver.sipcontainer:SIPContainer"
xmi:id="SIPContainer_1172257094386" name="" maxAppSessions="120000"
maxMessageRate="5000" maxDispatchQueueSize="3200" maxResponseTime="0"
statAveragePeriod="1000" st
atUpdateRange="10000"/>
    <webserverPluginSettings xmi:id="WebserverPluginSettings_1172257094386"
WaitForContinue="false" ConnectTimeout="0" MaxConnections="-1"
ExtendedHandshake="false" ServerIOTimeout="0"/>

```

</components>

For the Web container configuration, notice the session persistence configuration that controls the underlying web container clustering management of session state, and the sessionPersistenceMode of DATA_REPLICATION for availability of session data between cluster members. The cluster name is mentioned with the sessionDRSPersistence element and messageBrokerDomainName attribute to show that the Data Replication Service uses this domain name with the Reliable Multicast Messaging (RMM) protocol to share session data within the configured cluster.

The final part of the server.xml file controls the configuration items for the JVM and its command line as shown in Example 5-10, including extra arguments (such as the SPNEGO entry here for use of Kerberos and the SPNEGO TAI for single signon), and files to redirect native standard input and standard output to. The runAsUser and runAsGroup and priority information configured here gets mapped to the underlying platform (in this case, AIX).

Example 5-10 server.xml: JVM configuration detail

```
<processDefinitions xmi:type="processexec:JavaProcessDef"
xmi:id="JavaProcessDef_1172257094386" workingDirectory="${USER_INSTALL_ROOT}"
executableTargetKind="JAVA_CLASS" executableTarget="com.ibm.ws.runtime.WsServer">
  <execution xmi:id="ProcessExecution_1172257094387" processPriority="20"
runAsUser="" runAsGroup="" />
  <ioRedirect xmi:id="OutputRedirect_1172257094387"
stdoutFilename="${SERVER_LOG_ROOT}/native_stdout.log"
stderrFilename="${SERVER_LOG_ROOT}/native_stderr.log" />
  <monitoringPolicy xmi:id="MonitoringPolicy_1172257094387"
maximumStartupAttempts="3" pingInterval="60" pingTimeout="300" autoRestart="true"
nodeRestartState="STOPPED" />
  <jvmEntries xmi:id="JavaVirtualMachine_1172257094400" verboseModeClass="false"
verboseModeGarbageCollection="false" verboseModeJNI="false" runHProf="false"
debugMode="false" debugArgs="-Djava.compiler=NONE -Xdebug -Xnoagent -Xrunjdw
p:transport=dt_socket,server=y,suspend=n,address=7777"
genericJvmArguments="-Dcom.ibm.ws.security.spnego.isEnabled=true"
disableJIT="false" />
</processDefinitions>
```

5.2.7 serverindex.xml

After server.xml, the next most important file in a WebSphere Application Server environment is serverindex.xml. This file identifies the mapping of WebSphere Application Server variable names used elsewhere within the WebSphere Application Server configuration and environment to specific ports at the TCP/IP level for each WebSphere Application Server instance running on the node.

This information is important for firewall configuration, and can be useful when configuring the platform or Virtual I/O Server. To run multiple copies of WebSphere Application Server on an environment, or when creating multiple instances of WebSphere Application Server on a node for vertical clustering, these port values must be incremented for each successive instance or environment to avoid port clashes. Errors in this incrementing are one of the most common causes of failure for WebSphere Application Server to start up, along with lack of available memory.

Each instance on the node is identified by a serverEntries element with its role configured by the serverType attribute, and the configuration of the name is within the specialEndpoints element section and the port (or ports) in the endPoint element section; see Example 5-11.

Example 5-11 serverindex.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<serverindex:ServerIndex xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:serverindex="http://www.ibm.com/websphere/appserver/schemas/5.0/serverindex.xmi"
  xmi:id="ServerIndex_1" hostName="spock44p.RnREnterprise.dyndns.org"
endPointRefs="NamedEndPoint_1172252985953 NamedEndPoint_1172252985954
NamedEndPoint_1172252985955 NamedEndPoint_1172252985956 NamedEndPoint_1172252985957
NamedEndPoint_1172252985958 NamedEndPoint_1172252985959">
  <serverEntries xmi:id="ServerEntry_1172252985953" serverDisplayName="nodeagent"
serverName="nodeagent" serverType="NODE_AGENT">
    <specialEndpoints xmi:id="NamedEndPoint_1172252985974"
endPointName="BOOTSTRAP_ADDRESS">
      <endPoint xmi:id="EndPoint_1172252985974"
host="spock44p.RnREnterprise.dyndns.org" port="2809"/>
    </specialEndpoints>
    <specialEndpoints xmi:id="NamedEndPoint_1172252985953"
endPointName="ORB_LISTENER_ADDRESS">
      <endPoint xmi:id="EndPoint_1172252985975"
host="spock44p.RnREnterprise.dyndns.org" port="9900"/>
    </specialEndpoints>
  </serverEntries>
</serverindex:ServerIndex>
```

...

```

</specialEndpoints>
  <specialEndpoints xmi:id="NamedEndPoint_1172252985977"
endPointName="SOAP_CONNECTOR_ADDRESS">
  <endPoint xmi:id="EndPoint_1172252985984"
host="spock44p.RnREnterprise.dyndns.org" port="8878"/>
  </specialEndpoints>
</serverEntries>
  <serverEntries xmi:id="ServerEntry_1172257087924" serverName="spock44pMember01"
serverType="APPLICATION_SERVER">

<deployedApplications>UserTest_war.ear/deployments/UserTest_war</deployedApplications
>
  <specialEndpoints xmi:id="NamedEndPoint_1172257087924"
endPointName="BOOTSTRAP_ADDRESS">
  <endPoint xmi:id="EndPoint_1172257087924"
host="spock44p.RnREnterprise.dyndns.org" port="9810"/>
  </specialEndpoints>

...

<specialEndpoints xmi:id="NamedEndPoint_1172257090847"
endPointName="ORB_LISTENER_ADDRESS">
  <endPoint xmi:id="EndPoint_1172257090847"
host="spock44p.RnREnterprise.dyndns.org" port="0"/>
  </specialEndpoints>
</serverEntries>
  <serverEntries xmi:id="ServerEntry_1172338903155" serverName="spock44pMember02"
serverType="APPLICATION_SERVER">

...

</serverEntries>
</serverindex:ServerIndex>

```

Default values of port settings in serverindex.xml for Node Agents are listed in Table 5-3.

Table 5-3 Default port values: Node Agent

End point	Port value	Multicast IP Address
BOOTSTRAP_ADDRESS	2809	
ORB_LISTENER_ADDRESS	9900	

End point	Port value	Multicast IP Address
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS	9202	
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS	9201	
DCS_UNICAST_ADDRESS	9353	
DRS_CLIENT_ADDRESS	7888	
NODE_DISCOVERY_ADDRESS	7272	
NODE_IPV6_MULTICAST_DISCOVERY_ADDRESS	5001	ff01::1
NODE_MULTICAST_DISCOVERY_ADDRESS	5000	232.133.104.73
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	9901	
SOAP_CONNECTOR_ADDRESS	8878	

Default values of port settings in serverindex.xml for First Instances are shown in Table 5-4.

Table 5-4 Default port values: First Instance

End point	Port value
BOOTSTRAP_ADDRESS	9809
SOAP_CONNECTOR_ADDRESS	8879
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	9401
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS	9403
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS	9402
WC_adminhost	9060
WC_defaulthost	9080
DCS_UNICAST_ADDRESS	9352
WC_adminhost_secure	9043
WC_defaulthost_secure	9443
SIP_DEFAULTHOST	5060
SIP_DEFAULTHOST_SECURE	5061
SIB_ENDPOINT_ADDRESS	7276

End point	Port value
SIB_ENDPOINT_SECURE_ADDRESS	7286
SIB_MQ_ENDPOINT_ADDRESS	5558
SIB_MQ_ENDPOINT_SECURE_ADDRESS	5578
CELL_DISCOVERY_ADDRESS	7277
ORB_LISTENER_ADDRESS	9100

5.2.8 security.xml

The security.xml file configures the security in use by the environment for WebSphere Application Server-specific, J2EE standard, and Java standard values. Throughout this file are mappings between J2EE security standards and the IBM implementation of that standard. For example, IBM implements the java.security.provider standard with the IBMJSSE2 implementation, where JSSE2 is the IBM Java Secure Socket Extensions provider.

An AIX platform administrator will need to understand this file when configuring host security and SSL and firewall configuration, and when installing IBM cryptography adapters within the environment to allow the platform and WebSphere Application Server environments to function together.

The outermost element in the file is the security:Security element that is used to configure general settings such as whether security is enabled, how the user names are configured and what the cache timeout settings are; see Example 5-12.

Example 5-12 security.xml: outermost element

```
<?xml version="1.0" encoding="UTF-8"?>
<security:Security xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:orb.securityprotocol="http://www.ibm.com/websphere/appserver/schemas/5.0/orb.se
curityprotocol.xmi" xmlns:security="http://www.ibm.com/websphere/appserver/sche
mas/5.0/security.xmi" xmi:id="Security_1" useLocalSecurityServer="true"
useDomainQualifiedUserNames="true" enabled="false" cacheTimeout="600"
issuePermissionWarning="true" activeProtocol="BOTH" enforceJava2Security="true"
enforceFineGra
inedJCASecurity="false" appEnabled="false" dynamicallyUpdateSSLConfig="true"
activeAuthMechanism="LTPA_1" activeUserRegistry="LDAPUserRegistry_1"
defaultSSLSettings="SSLConfig_1">
```

Directly inside the security:Security tag is the authMechanisms element that configures what type of security is to be used by the instance (Lightweight Third Party Authentication), as shown in Example 5-13. This then controls what other elements are used within the configuration.

Of particular importance are the Trust Association Interceptor (TAI) settings that load an appropriate TAI to map the external security to the internal tokens using the trustAssociation and interceptors elements. For example, the SPNEGO TAI allows Kerberos authentication to be used by the Web container for single signon.

Example 5-13 security.xml: authMechanisms detail

```
<authMechanisms xmi:type="security:LTPA" xmi:id="LTPA_1" OID="oid:1.3.18.0.2.30.2"
authContextImplClass="com.ibm.ISecurityLocalObjectTokenBaseImpl.WSSecurityContextLTPA
Impl" authConfig="system.LTPA" simpleAuthConfig="system.LTPA" auth
ValidationConfig="system.LTPA" timeout="120" keySetGroup="KeySetGroup_1">

    <trustAssociation xmi:id="TrustAssociation_1" enabled="true">

...

    <interceptors xmi:id="TAInterceptor_3"
interceptorClassName="com.ibm.ws.security.spnego.TrustAssociationInterceptorImpl"/>

...
    </trustAssociation>
    <singleSignon xmi:id="SingleSignon_1" requiresSSL="false" domainName=""
enabled="true"/>
    </authMechanisms>

...

```

The userRegistries element, shown in Example 5-14 on page 180, is used to identify the location for user information. With WebSphere Application Server 6.1, a combination of user registries can be used concurrently via the Federated Repositories option. This section includes file and LDAP registry configuration, including support for Windows Active Directory®. This section is useful for identifying hostnames, IP addresses and ports for the target LDAP and Kerberos servers, so it can be useful for platform and network firewall configuration information.

Example 5-14 security.xml: userRegistries detail

```
<userRegistries xmi:type="security:LocalOSUserRegistry" xmi:id="LocalOSUserRegistry"
serverId="" serverPassword="{xor}" realm="" useRegistryServerId="true"
primaryAdminId=""/>
  <userRegistries xmi:type="security:CustomUserRegistry"
xmi:id="CustomUserRegistry_1" useRegistryServerId="true" primaryAdminId=""
customRegistryClassName="com.ibm.websphere.security.FileRegistrySample"/>
```

The authConfig and applicationLoginConfig section follows the userRegistries sections and configure modules for handling specific security environments, as shown in Example 5-15, such as those set up by the Java Authentication and Authorization Service (JAAS) or Tivoli Access Manager (TAM).

Example 5-15 security.xml: specific security environment detail

```
<authConfig xmi:id="AuthorizationConfig_1" useJACCProvider="false">
  <authorizationProviders xmi:id="AuthorizationProvider_1"
j2eePolicyImplClassName="com.tivoli.pd.as.jacc.TAMPolicy" name="Tivoli Access
Manager"
policyConfigurationFactoryImplClassName="com.tivoli.pd.as.jacc.TAMPolicyConfiguration
Fac
tory"
roleConfigurationFactoryImplClassName="com.tivoli.pd.as.jacc.TAMRoleConfigurationFact
ory" initializeJACCProviderClassName="com.tivoli.pd.as.jacc.cfg.TAMConfigInitialize"
requiresEJBArgumentsPolicyContextHandler="false" supportsDyn
amicModuleUpdates="true"/>
</authConfig>
  <applicationLoginConfig xmi:id="JAASConfiguration_1">
    <entries xmi:id="JAASConfigurationEntry_1" alias="ClientContainer">
      <loginModules xmi:id="JAASLoginModule_1"
moduleClassName="com.ibm.ws.security.common.auth.module.proxy.WSLoginModuleProxy"
authenticationStrategy="REQUIRED">
        <options xmi:id="Property_1" name="delegate"
value="com.ibm.ws.security.common.auth.module.WSClientLoginModuleImpl"/>
      </loginModules>
    </entries>
  </applicationLoginConfig>
```

Some of the most important configuration sections inside the security.xml file are found in the CSI and IBM element sections, as shown in Example 5-16 on page 181. These sections configure the CSIv2 security setup that underlies the J2EE EJB container standard, and in the latter case, the IBM

WebSphere-specific settings for it. Layers of security are configured in this area with IIOP settings and garbage collection settings to ensure appropriate management of the information used for security. Remember that IIOP is the protocol used for communication with the EJBs in the EJB container.

Example 5-16 security.xml: CSI and IBM element section detail

```
<CSI xmi:id="IIOPSecurityProtocol_1">
  <claims xmi:type="orb.securityprotocol:CommonSecureInterop" xmi:id="CSiv2 Inbound
Configuration" stateful="true">
    <layers xmi:type="orb.securityprotocol:IdentityAssertionLayer"
xmi:id="IdentityAssertionLayer_1">
      <supportedQOP xmi:type="orb.securityprotocol:IdentityAssertionQOP"
xmi:id="IdentityAssertionQOP_1" enable="false"/>
    </layers>

    ...

    <serverAuthentication xmi:id="IIOPTransport_1" sslConfig=""/>
  </layers>
</claims>
  <performs xmi:type="orb.securityprotocol:CommonSecureInterop" xmi:id="CSiv2
Outbound Configuration" stateful="true" sessionGCInterval="300000"
sessionGCIdleTime="900000">
    <layers xmi:type="orb.securityprotocol:IdentityAssertionLayer"
xmi:id="IdentityAssertionLayer_2">

    ...

    </layers>
  </layers>

  ...

  <serverAuthentication xmi:id="IIOPTransport_2" sslConfig=""/>
  </layers>
</performs>
</CSI>

  <IBM xmi:id="IIOPSecurityProtocol_2">
    <claims xmi:type="orb.securityprotocol:SecureAssociationService"
xmi:id="SecureAssociationService_1">
      <layers xmi:type="orb.securityprotocol:TransportLayer"
xmi:id="TransportLayer_3">
        <supportedQOP xmi:type="orb.securityprotocol:TransportQOP"
xmi:id="TransportQOP_5" enableProtection="true" confidentiality="true"
integrity="true"/>
      </layers>
    </claims>
  </IBM>
```

```

        <serverAuthentication xmi:id="IIOPTransport_3" sslConfig=""/>
    </layers>
</claims>
    <performs xmi:type="orb.securityprotocol:SecureAssociationService"
xmi:id="SecureAssociationService_2">
        <layers xmi:type="orb.securityprotocol:TransportLayer"
xmi:id="TransportLayer_4">
            <supportedQOP xmi:type="orb.securityprotocol:TransportQOP"
xmi:id="TransportQOP_6" enableProtection="true" confidentiality="false"
integrity="false"/>
            <serverAuthentication xmi:id="IIOPTransport_4" sslConfig=""/>
        </layers>
    </performs>
</IBM>

```

The repertoire elements section, shown in Example 5-17, configures SSL/TLS and keystore settings for the Cell and Node. Note that for Web services security, SSL is *not* part of the WS-Security standards because it is subject to replay attacks under certain circumstances. For this reason, TLS is likely to be configured here.

Example 5-17 security.xml: repertoire element section detail

```

<repertoire xmi:id="SSLConfig_1" alias="CellDefaultSSLSettings"
managementScope="ManagementScope_1">
    <setting xmi:id="SecureSocketLayer_1" clientAuthentication="false"
securityLevel="HIGH" enabledCiphers="" jsseProvider="IBMJSSE2" sslProtocol="SSL_TLS"
keyStore="KeyStore_1" trustStore="KeyStore_2" trustManager="TrustManager_1" keyM
anager="KeyManager_1"/>
</repertoire>
<repertoire xmi:id="SSLConfig_1172252999890" alias="NodeDefaultSSLSettings"
managementScope="ManagementScope_1172252999891">
    <setting xmi:id="SecureSocketLayer_1172252999891" clientAuthentication="false"
securityLevel="HIGH" enabledCiphers="" jsseProvider="IBMJSSE2" sslProtocol="SSL_TLS"
keyStore="KeyStore_1172252999891" trustStore="KeyStore_2" trustManag
er="TrustManager_1172252999891" keyManager="KeyManager_1172252999891"/>
</repertoire>

```

The systemLoginConfig element section, shown in Example 5-18 on page 183, contains a large number of entries elements that contain loginModules elements. These elements and their attribute values configure the different login modules and options for different types of login.

The module configurations for Lightweight Third Party Authentication (LTPA) and Simple WebSphere Authentication Mechanism (SWAM) high level enabling configuration is here. SWAM was important in earlier WebSphere Application Server releases, but is deprecated in Version 6.1 and will be removed in a future release.

Example 5-18 security.xml: systemLoginConfig section detail

```
<systemLoginConfig xmi:id="JAASConfiguration_2">
  <entries xmi:id="JAASConfigurationEntry_4" alias="SWAM">
    <loginModules xmi:id="JAASLoginModule_4"
moduleClassName="com.ibm.ws.security.common.auth.module.proxy.WSLoginModuleProxy"
authenticationStrategy="REQUIRED">
      <options xmi:id="Property_8" name="delegate"
value="com.ibm.ws.security.server.lm.swamLoginModule"/>
    </loginModules>
  </entries>
  <entries xmi:id="JAASConfigurationEntry_5" alias="LTPA">
    <loginModules xmi:id="JAASLoginModule_5"
moduleClassName="com.ibm.ws.security.common.auth.module.proxy.WSLoginModuleProxy"
authenticationStrategy="REQUIRED">
      <options xmi:id="Property_9" name="delegate"
value="com.ibm.ws.security.server.lm.ltpaLoginModule"/>
    </loginModules>
  </entries>
  ...
  <entries xmi:id="JAASConfigurationEntry_28" alias="WSS_OUTBOUND">
    <loginModules xmi:id="JAASLoginModule_29"
moduleClassName="com.ibm.ws.security.server.lm.wsMapCSIV20outboundLoginModule"
authenticationStrategy="REQUIRED"/>
  </entries>
</systemLoginConfig>
```

The properties element section, shown in Example 5-19, identifies individual properties that can be configured and enabled as part of the WebSphere Application Server security setup.

Example 5-19 security.xml: properties element section detail

```
<properties xmi:id="Property_20" name="security.enablePluggableAuthentication"
value="true" required="false"/>
```

...

```
<properties xmi:id="Property_37" name="com.ibm.audit.auditPolicy" value="REQUIRED"
required="false"/>
  <properties xmi:id="Property_38" name="com.ibm.audit.auditQueueSize" value="5000"
required="false"/>
```

...

```
  <properties xmi:id="Property_43" name="com.ibm.security.useFIPS" value="false"
required="false"/>
  <properties xmi:id="Property_44" name="com.ibm.websphere.security.DeferTAIttoSSO"
value="com.ibm.ws.security.spnego.TrustAssociationInterceptorImpl" description="Trust
Association Interceptors in this list will be invoked after Single
Sign On user validation." required="false"/>
  <properties xmi:id="Property_1172272088006"
name="com.ibm.security.SAF.authorization" value="false"/>
```

The webAuthAttrs element section, shown in Example 5-20, configures the behavior of the authentication mechanisms at the application server level for Web container, particularly what happens when a particular type of authentication mechanism is not available for a given application.

Example 5-20 security.xml: webAuthAttrs element section detail

```
<webAuthAttrs xmi:id="DescriptiveProperty_1"
name="com.ibm.wsspi.security.web.webAuthReq" value="lazy" type="String"
displayNameKey="" nlsRangeKey="" hoverHelpKey="" range="lazy,persisting,always"
inclusive="false" firstClass="false"/>
>
  <webAuthAttrs xmi:id="DescriptiveProperty_2"
name="com.ibm.wsspi.security.web.failOverToBasicAuth" value="false" type="boolean"
displayNameKey="" nlsRangeKey="" hoverHelpKey="" range="" inclusive="false"
firstClass="false"/>
```

Within the WebSphere Application Server ND and WebSphere Application Server XD environments, security can be configured and managed at various levels (that is, at the cell level, the node level, and so on). This is configured in the managementScopes element sections, shown in Example 5-21 on page 185, that identify what levels are in use in the given environment.

Example 5-21 *security.xml: managementScopes element section detail*

```
<managementScopes xmi:id="ManagementScope_1" scopeName="(cell):spock44pCell01"
scopeType="cell"/>
  <managementScopes xmi:id="ManagementScope_1172252999891"
scopeName="(cell):spock44pCell01:(node):spock44pNode01" scopeType="node"/>
```

One of the *most* important configuration items for the security.xml file is the keyStores element section, shown in Example 5-22. It identifies the keystore locations, what security is used to access them (including passwords), what internal providers are used for access and management, and then which security management scope they apply to.

Example 5-22 *security.xml: keyStores element section detail*

```
<keyStores xmi:id="KeyStore_1" name="CellDefaultKeyStore" password="{xor}CDo9Hgw="
provider="IBMJCE" location="${CONFIG_ROOT}/cells/spock44pCell01/key.p12"
type="PKCS12" fileBased="true" hostList="" managementScope="ManagementScope_1"
/>
  <keyStores xmi:id="KeyStore_2" name="CellDefaultTrustStore"
password="{xor}CDo9Hgw=" provider="IBMJCE"
location="${CONFIG_ROOT}/cells/spock44pCell01/trust.p12" type="PKCS12"
fileBased="true" hostList="" managementScope="ManagementScope_1"
/>
  <keyStores xmi:id="KeyStore_3" name="CellLTPAKeys" password="{xor}CDo9Hgw="
provider="IBMJCE" location="${CONFIG_ROOT}/cells/spock44pCell01/ltpa.jceks"
type="JCEKS" fileBased="true" hostList="" managementScope="ManagementScope_1"
/>
  <keyStores xmi:id="KeyStore_1172252999891" name="NodeDefaultKeyStore"
password="{xor}CDo9Hgw=" provider="IBMJCE"
location="${CONFIG_ROOT}/cells/spock44pCell01/nodes/spock44pNode01/key.p12"
type="PKCS12" fileBased="true" hostList="" managementScope="ManagementScope_1172252999891"
/>
  ...
<keyStores xmi:id="KeyStore_1172252999910" name="NodeDefaultTrustStore"
password="{xor}CDo9Hgw=" provider="IBMJCE"
location="${CONFIG_ROOT}/cells/spock44pCell01/nodes/spock44pNode01/trust.p12"
type="PKCS12" fileBased="true" hostList=""
managementScope="ManagementScope_1172252999891"
/>
```

During the setup of an SSL connection, the application server uses trust managers for certificate validation during the handshake process, with a standard X509 and an IBM-specific PKI provider provided by default, although

custom trust managers can also be installed. These perform tasks like checking certificate revocation lists (CRLs).

The trustManagers element configuration section, shown in Example 5-23, should be considered a key part of the SSL configuration for the platform, and should be examined if the platform is experiencing SSL-related networking problems.

Example 5-23 security.xml: trustManagers element section detail

```
<trustManagers xmi:id="TrustManager_1" name="IbmX509" provider="IBMJSSE2"
algorithm="IbmX509" managementScope="ManagementScope_1"/>
  <trustManagers xmi:id="TrustManager_2" name="IbmPKIX" provider="IBMJSSE2"
algorithm="IbmPKIX" trustManagerClass="" managementScope="ManagementScope_1">
    <additionalTrustManagerAttrs xmi:id="DescriptiveProperty_1"
name="com.ibm.security.enableCRLDP" value="true" type="boolean" displayNameKey=""
nlsRangeKey="" hoverHelpKey="" range="" inclusive="false" firstClass="false"/>
    <additionalTrustManagerAttrs xmi:id="DescriptiveProperty_2"
name="com.ibm.jsse2.checkRevocation" value="true" type="boolean" displayNameKey=""
nlsRangeKey="" hoverHelpKey="" range="" inclusive="false" firstClass="false"/>
  </trustManagers>
  <trustManagers xmi:id="TrustManager_1172252999891" name="IbmX509"
provider="IBMJSSE2" algorithm="IbmX509"
managementScope="ManagementScope_1172252999891"/>
  <trustManagers xmi:id="TrustManager_1172252999911" name="IbmPKIX"
provider="IBMJSSE2" algorithm="IbmPKIX" trustManagerClass=""
managementScope="ManagementScope_1172252999891">
    <additionalTrustManagerAttrs xmi:id="DescriptiveProperty_1172252999911"
name="com.ibm.security.enableCRLDP" value="true" type="boolean" displayNameKey=""
nlsRangeKey="" hoverHelpKey="" range="" inclusive="false" firstClass="false"/>
    <additionalTrustManagerAttrs xmi:id="DescriptiveProperty_1172252999912"
name="com.ibm.jsse2.checkRevocation" value="true" type="boolean" displayNameKey=""
nlsRangeKey="" hoverHelpKey="" range="" inclusive="false" firstClass="false"/
>
  </trustManagers>
```

The keyManagers, keySetGroups, and keySets element sections, shown in Example 5-24 on page 187, are used to configure access to keystores and set up the management scope they apply to and what providers they use. Passwords for accessing those keystores are configured here, therefore, they should be protected.

```
<keyManagers xmi:id="KeyManager_1" name="IbmX509" provider="IBMJSSE2"
algorithm="IbmX509" keyManagerClass="" managementScope="ManagementScope_1"/>
  <keyManagers xmi:id="KeyManager_1172252999891" name="IbmX509" provider="IBMJSSE2"
algorithm="IbmX509" keyManagerClass=""
managementScope="ManagementScope_1172252999891"/>
  <keySetGroups xmi:id="KeySetGroup_1" name="CellLTPAKeySetGroup" autoGenerate="true"
wsSchedule="WSSchedule_1" keySet="KeySet_1 KeySet_2"
managementScope="ManagementScope_1"/>
    <keySets xmi:id="KeySet_1" name="CellLTPAKeyPair" aliasPrefix="LTPAKeyPair"
password="{xor}CDo9Hgw=" maxKeyReferences="2" deleteOldKeys="true"
keyGenerationClass="com.ibm.ws.security.ltpa.LTPAKeyPairGenerator" isKeyPair="true"
keyStore="KeyStore_3" managementScope="ManagementScope_1">
      <keyReference xmi:id="KeyReference_1172271449639" keyAlias="LTPAKeyPair_3"
version="3"/>
      <keyReference xmi:id="KeyReference_1196517802804" keyAlias="LTPAKeyPair_4"
version="4"/>
    </keySets>
    <keySets xmi:id="KeySet_2" name="CellLTPASecret" aliasPrefix="LTPASecret"
password="{xor}CDo9Hgw=" maxKeyReferences="2"
keyGenerationClass="com.ibm.ws.security.ltpa.LTPAKeyGenerator" keyStore="KeyStore_3"
managementScope="ManagementScope_1">
      <keyReference xmi:id="KeyReference_1172271460420" keyAlias="LTPASecret_3"
version="3"/>
      ...
    </keySets>
    <keySets xmi:id="KeySet_2" name="CellLTPASecret" aliasPrefix="LTPASecret"
password="{xor}CDo9Hgw=" maxKeyReferences="2"
keyGenerationClass="com.ibm.ws.security.ltpa.LTPAKeyGenerator" keyStore="KeyStore_3"
managementScope="ManagementScope_1">
      <keyReference xmi:id="KeyReference_1172271460420" keyAlias="LTPASecret_3"
version="3"/>
      <keyReference xmi:id="KeyReference_1196517820815" keyAlias="LTPASecret_4"
version="4"/>
    </keySets>
```

The wsSchedules, wsNotifications, and wsCertificateExpirationMonitor element sections, shown in Example 5-25, are used to configure scheduled checks against keys and certificates for expiration or invalidation (that is, against certificate revocation lists) and to perform the necessary actions and notifications.

Example 5-25 security.xml: wsSchedules, wsNotifications, wsCertificateExpirationMonitor section detail

```
<wsSchedules xmi:id="WSSchedule_1" name="LTPAKeySetSchedule" frequency="90"
dayOfWeek="1" hour="22" nextStartDate="1211148043754"/>
  <wsSchedules xmi:id="WSSchedule_2" name="ExpirationMonitorSchedule" frequency="30"
dayOfWeek="1" hour="21" minute="30" nextStartDate="1201469458829"/>
  <wsNotifications xmi:id="WSNotification_1" name="MessageLog" logToSystemOut="true"
emailList=""/>
  <wsCertificateExpirationMonitor xmi:id="WSCertificateExpirationMonitor_1"
name="Certificate Expiration Monitor" autoReplace="true" deleteOld="true"
daysBeforeNotification="60" isEnabled="true" wsNotification="WSNotification_1"
wsSchedule="WSSchedule_2"/>
```

The sslConfigGroups element section, shown in Example 5-26, is used to map a given SSL configuration for the environment to the appropriate management scope configured in the managementScopes section.

Example 5-26 security.xml: sslConfigGroups element section detail

```
<sslConfigGroups xmi:id="SSLConfigGroup_1" name="spock44pCell01" direction="inbound"
sslConfig="SSLConfig_1" managementScope="ManagementScope_1"/>
  <sslConfigGroups xmi:id="SSLConfigGroup_2" name="spock44pCell01"
direction="outbound" sslConfig="SSLConfig_1" managementScope="ManagementScope_1"/>
  <sslConfigGroups xmi:id="SSLConfigGroup_1172252999925" name="spock44pNode01"
direction="inbound" sslConfig="SSLConfig_1172252999890"
managementScope="ManagementScope_1172252999891"/>
  <sslConfigGroups xmi:id="SSLConfigGroup_1172252999926" name="spock44pNode01"
direction="outbound" sslConfig="SSLConfig_1172252999890"
managementScope="ManagementScope_1172252999891"/>
</security:Security>
```

5.3 IBM J9 Java Virtual Machine Architecture

This section provides an overview of Java Virtual Machines in general, and then focuses on the specifics of the IBM JVM implementation used by WebSphere on AIX.

5.3.1 Generic Java Virtual Machine overview

To obtain more detailed information about the specifications for implementing a Java Virtual Machine, refer to the following sources.

- ▶ The Java Virtual Machine Specification is available at the following address:
<http://java.sun.com/docs/books/jvms/>
- ▶ The Java Language Specification is available at the following address:
<http://java.sun.com/docs/books/jls/>

Figure 5-3 on page 190 illustrates a generic Java Virtual Machine implementation architecture.

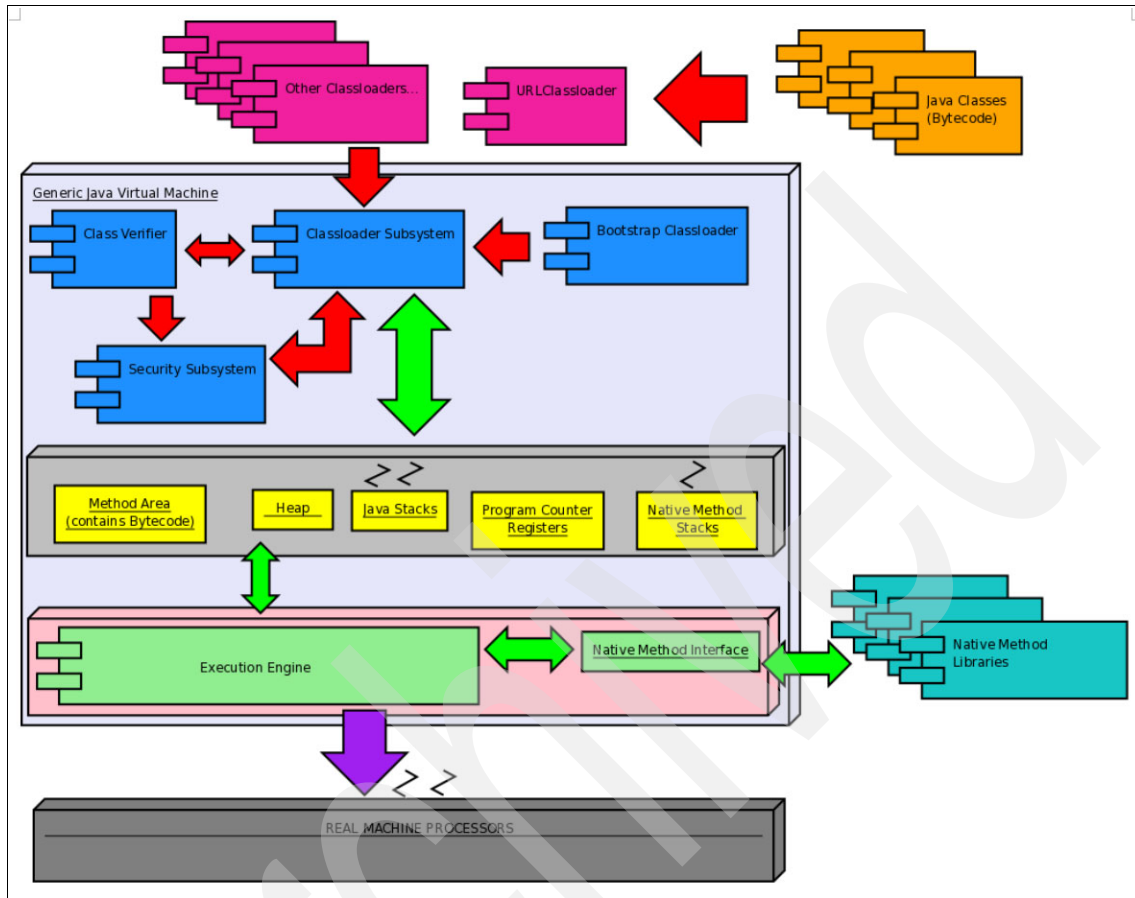


Figure 5-3 Generic JVM implementation architecture

Class file format

Java class files have a predefined format laid out in the Java standards. This format uses *byte code* to define the Java instructions, not native machine instructions. It also splits the class file into various parts identifying constants, field names and types, method names and types, and referenced classes, fields and methods. The methods contain the code that is later translated into native code for execution.

Classloaders

The Java class files are loaded by the Java virtual machine using *classloaders*. From the very beginning, Java allowed multiple classloaders, in a loose hierarchy, to be used to load code within a single Java virtual machine.

At the core is the Java bootstrap classloader which loads classes in a default way, usually from local disk, and is part of the JVM implementation itself and not modifiable by users. All but the bootstrap classloader are typically written in Java. They are designed to enable extensibility to the functionality of the classloading mechanism, such as in the URLClassLoader, which can load classes across a network using a URL to identify the class to be loaded.

The classloaders load Java byte code into the virtual machine and pass it through a class verifier and check it against the “domain” security mechanisms to ensure the code is safe for the given user to execute. The goal of the classloaders is to put the byte code itself into the class area, and create the necessary constant pool, static variables, and other artifacts necessary for the code to work with.

As part of the IBM implementation, classes are split in two and stored in different places. The first part is the immutable part of classes (primarily, method names and byte code). It is placed in a ROMClass area (where ROM stands for Read Only Memory to signify that this never changes after the initial write).

The second half of the class is placed in a RAMClass area, and tends to be based around an instance of a class after it is created, such as static variables and several caches. There may be multiple instances a class of a given name, partitioned by classloader, so classes are segregated by classloader.

Stack-based

The Java virtual machine is a stack-based machine and works by pushing and popping information onto a stack, and then performing operations that manipulate this information and leave a result on this stack. There are no Java virtual registers except for a program counter per thread.

Because the language is multithreaded, there are multiple Java stacks with multiple stack frames that must be accommodated. And because the Java language allows calls into the base operating system using the Java Native Interface (JNI), and the JVM has no control over how this platform-specific code works, it plays safe by having a separate stack for native code and takes care of weaving the execution together properly.

Execution

Up to now we have been working with Java code and Java code that “wraps” and calls into native code. When the Java code from the class area is executed by a thread, it is passed to an execution engine that actually executes the code on the platform processor.

It is important to recognize that there may be a difference between the number of Java threads and the number of platform threads, and a difference in sizes

between Java variables and platform variables. These differences are implementation-specific.

The Java language itself is usually thought of as 32-bit. This statement is not strictly accurate because the Java language does not have true pointers (it has references), and it does not have registers. The language implementation can be 32-bit, 64-bit, or even 16-bit, as long as the Java language specification is adhered to and expected results are produced.

The execution engine implementation for the Java Virtual Machine is not specified beyond the behavior of the Java byte code instructions and the results. The implementer can have an execution engine that performs interpretation to map each byte code instruction, one by one, to a stream of native instructions. Or the implementer can perform Just-In-Time (JIT) compilation to translate Java methods into native code before executing it. Or it can do a mix of the two ways, depending some statistical analysis at classloading time. Ultimately, the Java byte code is translated to platform-specific native code before execution on the real processors.

Instructions

Java has instructions that can be expected of any real machine, such as adding, subtracting, and so on. They map fairly well to native instructions. It has some instructions that can be optimized in hardware, such as some of those used for arithmetic functions with the `BigDecimal` class that have been optimized on the POWER6 processor. However, it also has some instructions that are complex to implement due to the difference between a virtual machine and a real machine.

For example, the Java `INVOKEVIRTUAL` byte code instruction creates the stack frame entries for a method call, loads the necessary byte code for the target method, and then calls into it. This is difficult for a processor to optimize because the Java language specifies additional requirements on execution that real machines do not typically have, including exception throwing, and the target may well be byte code rather than native code.

Memory management

Java does not allow the programmer to explicitly deallocate objects or the memory they use, and the Java virtual machine is tasked in the Java language specification with performing garbage collection to tidy up the Java heap for objects that are no longer reachable, typically meaning in scope of any executing threads. How the garbage collection is implemented is left to the implementer, but usually a traditional mark-sweep algorithm is chosen to identify what can be tidied up. There are options that allow parts of the work to be performed on a background thread. For more information about this topic, refer to “Garbage collection” on page 209.

5.3.2 IBM J9 JVM

On IBM platforms, and some others, an IBM Java Virtual Machine is implemented that offers extra features and enhanced performance beyond the standard reference JVM. The IBM Java Virtual Machine is tuned to make the most of the platform and uses a combination of Just-In-Time (JIT) compilation and interpretation, based on the expected execution overheads of both compared to the number of executions and underlying code performance. The IBM implementation shipped for use by WebSphere Application Server is called J9, and it ships in both 64-bit and 32-bit versions; see Figure 5-4.

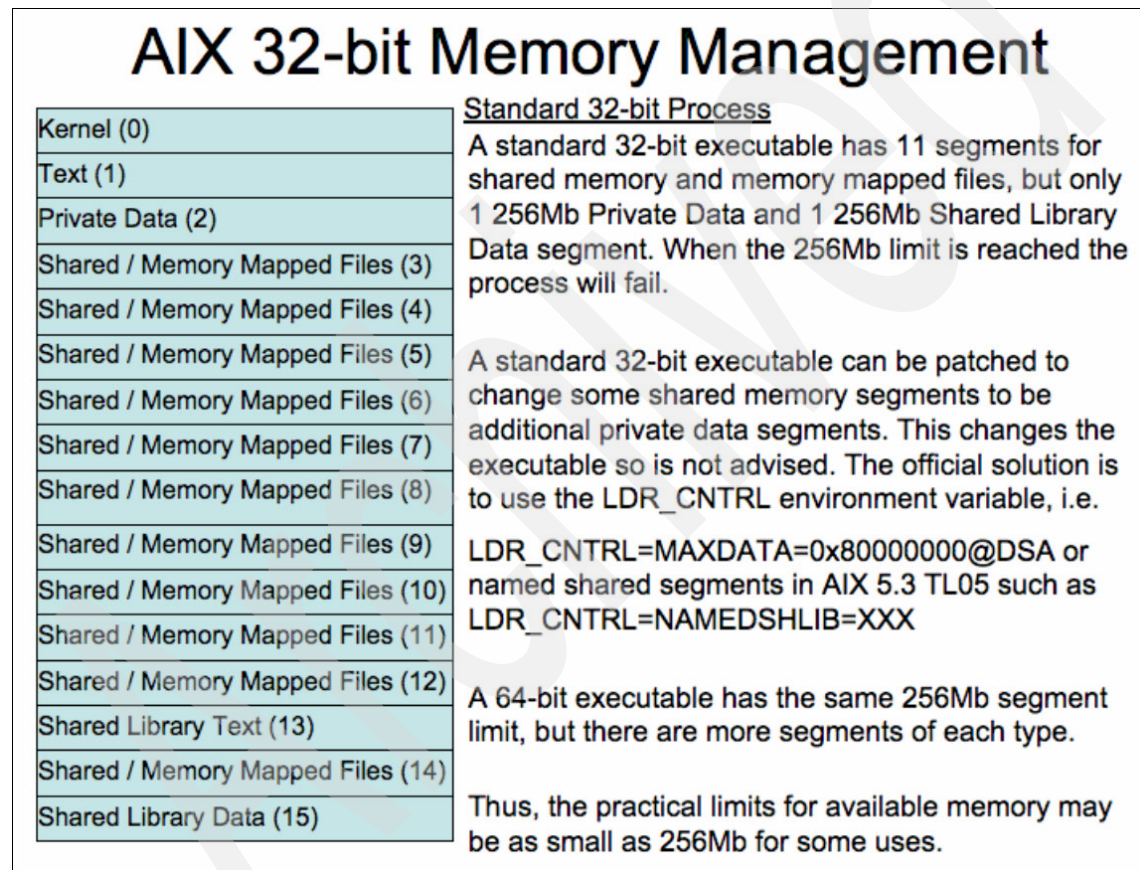


Figure 5-4 AIX 32-bit memory management

For AIX, all processes are split into 256 MB segments that have specific uses. For a 32-bit process that has only 16 segments (to access 4 GB of address space), a number of these (11) are normally used for shared memory, and one in particular (segment 2) contains the process heap and stack (heading towards

each other) and any private shared object data segments. A typical application server and JVM usually have some native code libraries accessed via the Java Native Interface (JNI) that would take up space in this segment.

From 1.4 JVM onward, IBM made use of some of the shared memory segments to support a large heap for Java applications without the use of the AIX LDR_CNTRL environment variable. However, even with this, the heap is limited to just over 3 GB of RAM. In this large model, the user stack remains in segment 2, but the user heap moves to start in segment 3 and make use of a number of the shared memory segments (see Figure 5-5). For small applications, this new layout may be sufficient.

If your application can leverage a large heap size, or if it requires high precision calculations that can leverage extra registers, then 64-bit JVM is preferable.

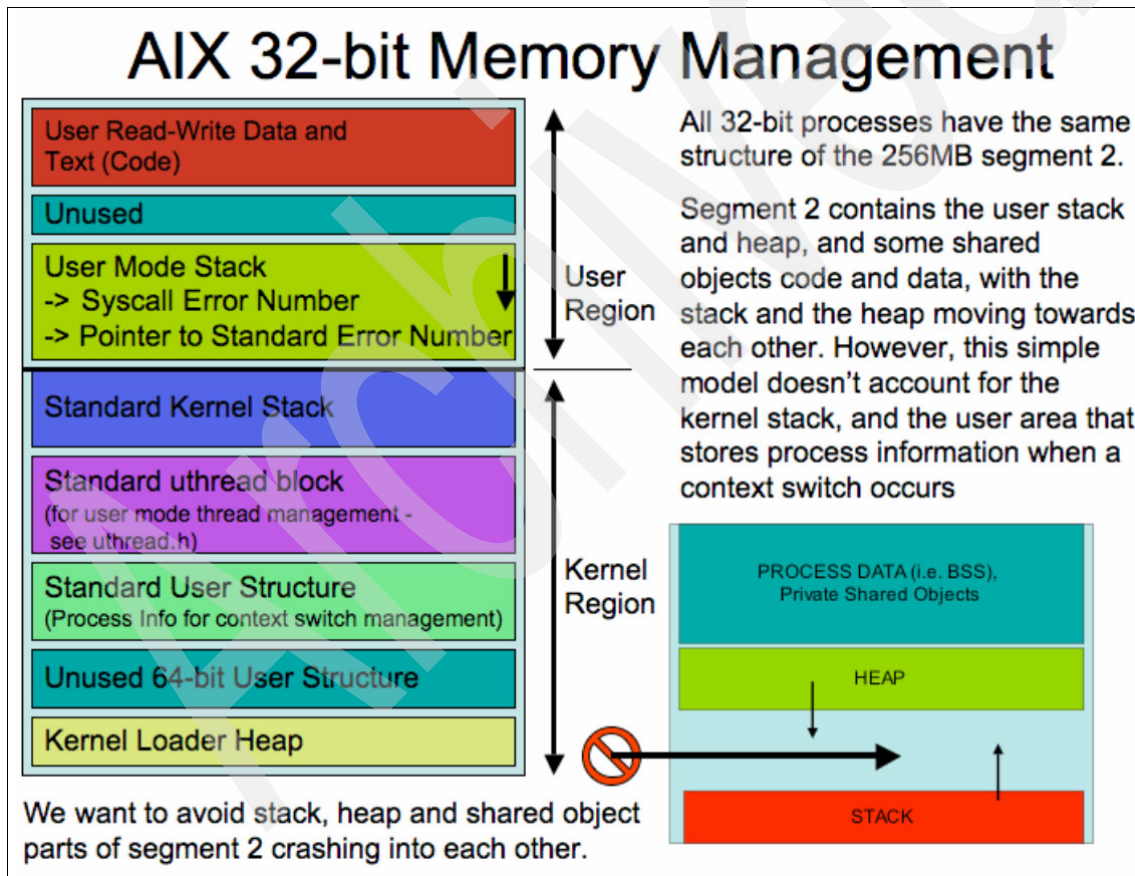


Figure 5-5 AIX 32-bit memory management, JVM 1.4

With the 64-bit AIX processes, such as supported by the IBM 64-bit J9 JVM, the first 4 GB of address space is reserved for compatibility with 32-bit code, but up to 448 PB of user heap address space is available to the process. So, although 256 MB segments are still in use, the large numbers of them available makes the previous issue irrelevant with current systems using less than a few TB of RAM; see Figure 5-6.

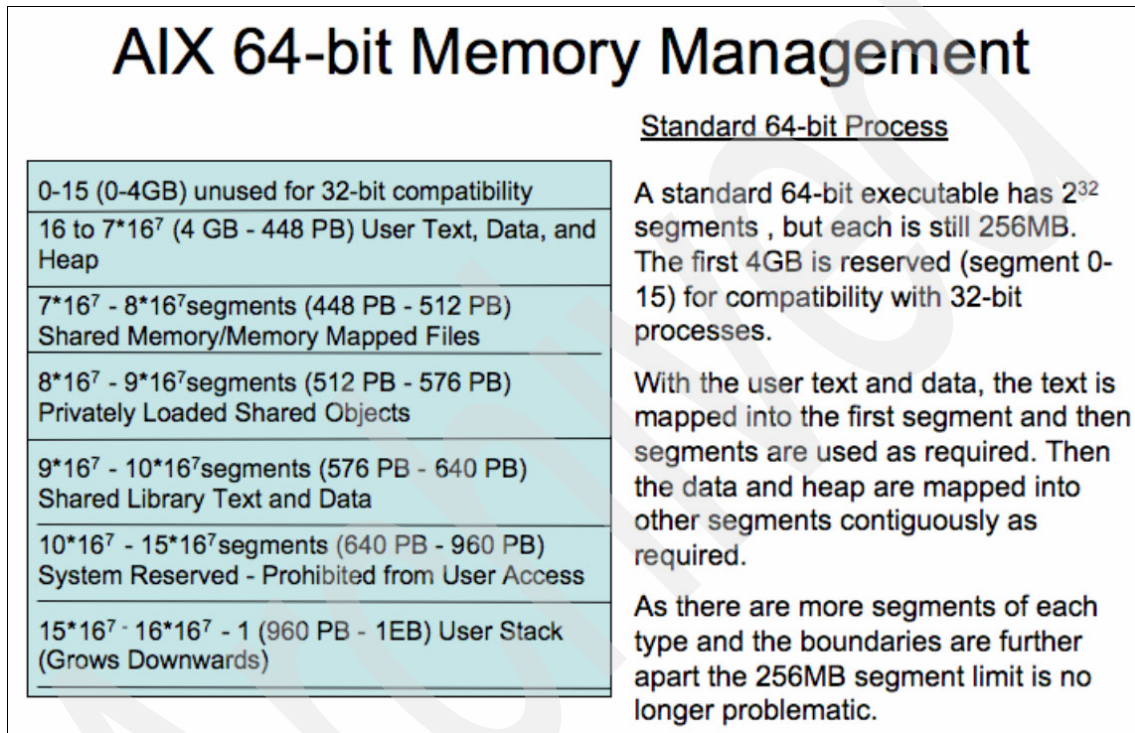


Figure 5-6 AIX 64-bit memory management

The IBM J9 Java 5 JVM was developed with features to best exploit the AIX platform and offer benefits to Java code outside of platform-specific features. Understanding how it works enables it to be used to best effect for enhanced performance with reduced resource utilization.

The Java Virtual Machine is just that, a virtual machine. That is, to the code that it runs it seems to be a real machine with its own instruction set, execution engine, thread management, memory management, and so on. But to the real host machine, it is just a single process.

Most Java Virtual Machine Runtime implementations consist of the Java byte code execution engine in the JVM itself, the Just-In-time (JIT) compiler, and the garbage collector; see Figure 5-7. All of these elements have been enhanced in the IBM J9 implementation that underpins the WebSphere Application Server Version 6.1 and above on AIX.

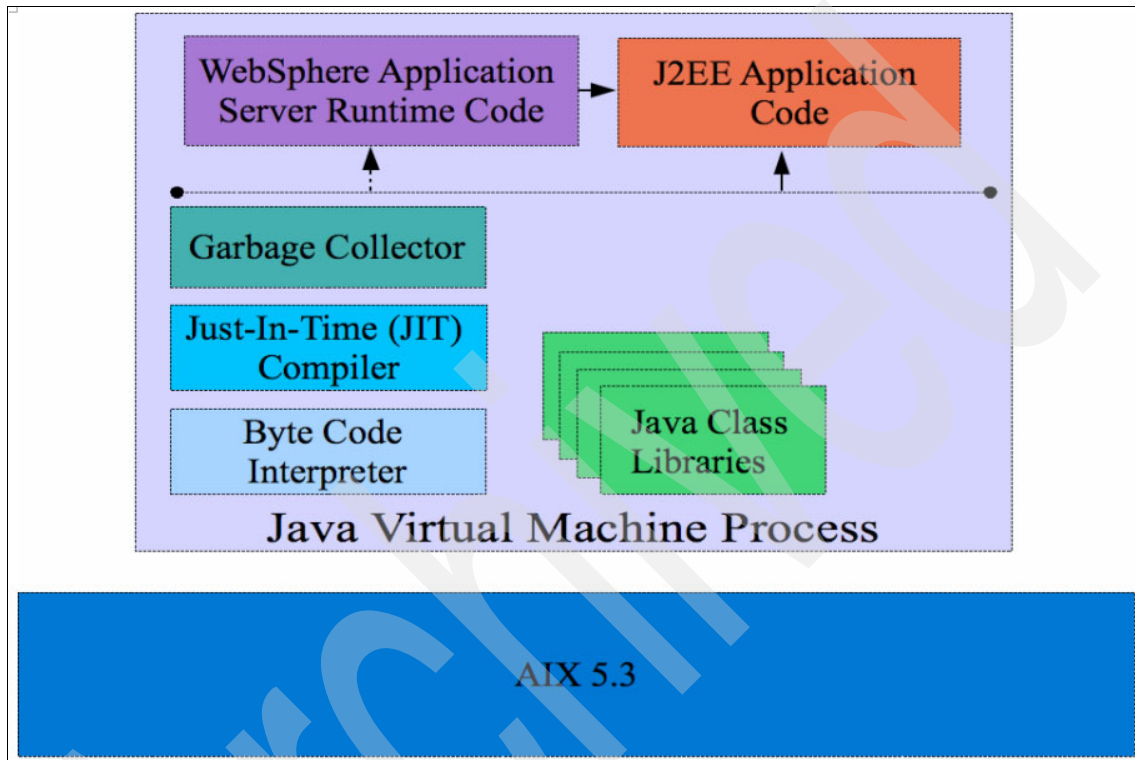


Figure 5-7 WebSphere Application Server inside the JVM on AIX

IBM J9 JVM features

The following list highlights the features of the IBM Java5 J9 JVM, as compared to a generic JVM:

- ▶ Shared classes
- ▶ Type safe and generational garbage collection
- ▶ Asynchronous, queued compilation in a background thread
- ▶ Five levels of code optimization with continuous sampling and profiling
- ▶ Support for large page sizes
- ▶ Numerous RAS and serviceability enhancements

Each of these features exploits the strengths of the underlying platform to accelerate any application running within the JVM (that is, the WebSphere

Application Server). The IBM J9 JVM code is continuously optimized, from a speed and memory utilization perspective, to exploit the platform. The following sections Lets look at the most important of these in depth.

Just-in-Time (JIT) Compiler: optimization and compilation

Another key feature of the IBM J9 JVM on which WebSphere Application Server on AIX runs is the way in which Java byte code is turned into native code. Traditionally, the JVM loaded the byte code for each Java class it encountered during its execution and either executed it directly via interpretation or used a synchronous JIT compiler to turn the class into native code for executing on the real machine directly. The key to this behavior is that traditional Java execution of native code is synchronous and static; that is, after the code is native, that is the way it stays.

The IBM current J9 Java Virtual Machine for AIX, and other platforms, borrows methods from optimizing compiler and the database server design disciplines. Indeed, the some of the technology in the JIT Compiler itself is independent of Java and might well be used with static C or Fortran front-ends. The Java Virtual Machine is running code dynamically to achieve its ends, so it needs to disable the AIX 5.3 TL03 Stack Execution Disable setting in order to execute code it generates and runs on its stack. It does this through the use of the DEP_EXEMPT flag in the XCOFF header of the Java launcher. This is perfectly safe for the JVM process and does not affect the rest of the system.

The Java instruction set is not the same as the native code instruction set, so at load time the Java byte code is data. It is not until that byte code is translated to native code that the underlying platform sees it as real code and can handle management of the processor caches; perform instruction pre-fetches; load the Java stack variables and heap references into the data cache; populate processor registers, and so on. RISC processors, such as the IBM POWER family and those underlying most modern CISC processor implementations, rely on this information to maximize performance through appropriate processor resource allocation and scheduling.

When the IBM J9 JVM runs, it looks ahead and compiles Java byte code to native code asynchronously on a *background* thread, but it does not stop there. As in a database server, the background thread uses statistics and heuristics in its processing. So, when the asynchronous background thread first compiles a method to native code, it uses heuristics to work out the costs and benefits of compiling and optimizing the code to different levels for better memory and processor performance. These heuristics are important because there is no point in wasting expensive processing resources in compiling to the highest levels of performance if the code is only run once and does not take much time to run anyway.

But how, you may wonder, does the JIT compiler running on the background thread know how often the code will execute? Like a database server, it collects statistics, via JVM interpreter sampling, JIT compiler sampling, or even by the insertion of profiling hooks into the code itself. This is best explained with a step-by-step example.

Suppose the background thread finds a class `myClass` with a method `myMethod` that will take 0.2 milliseconds to run unoptimized and 0.1 milliseconds to run optimized. If the optimization process takes 0.1 milliseconds and the method is only run once, then there is no benefit in optimization because the outcome is the same either way, with an overall execution time for compilation and execution of 0.2 milliseconds. However, if the code is executed hundreds of times, then there is a benefit in taking that 0.1 millisecond cost up front.

The IBM J9 JIT compiler and JVM goes further than this example; it offers five levels of execution (for example, interpretation, compilation and multi-level optimization) and the background thread collects statistics on how often everything is used. If a class method is used to the point where it appears that further optimization is necessary, the byte code is recompiled to a further level of optimization on the background thread and then transactionally replaced in the native code store for later use. What this means is that a traditional WebSphere Application Server application—and even WebSphere Application Server itself—gets faster over time with more use because the optimization occurs continually to produce faster and more efficient native code. This is known as dynamic recompilation with runtime execution profiling.

The process is as follows:

1. Each method starts out being interpreted; that is, level 1. This runs on the JVM directly.
2. After a number of invocations, the heuristics are used to compile the byte code at cold or warm level.
3. The background sampling thread is used to identify “hot” methods.
4. When at a hot level, a profiling step may be used to decide how and whether to transition to a scorching level. Certain hot methods are identified as “hot enough” to be recompiled with profiling hooks inserted.
5. The profiled hot methods that require further optimization are transitioned to scorching level.

The JIT Compiler and Interpreter both sample the usage of methods to make the decision as to when the optimization is necessary. This includes deciding when profiling hooks should be inserted into the hot code to determine the requirements of transitioning to a scorching optimization level.

Many types of optimization are available in the JIT Compiler with the IBM J9 JVM:

- ▶ Inlining
- ▶ Cold block outlining
- ▶ Value propagation
- ▶ Block hoisting
- ▶ Loop unroller
- ▶ Asynchronous check removal
- ▶ Copy propagation
- ▶ Loop versioning
- ▶ Common sub-expression elimination
- ▶ Partial redundancy elimination
- ▶ Optimal store placement
- ▶ Simplifier
- ▶ Escape analysis
- ▶ Dead tree removal
- ▶ Switch analysis
- ▶ Redundant monitor elimination

Readers who are familiar with C and C++ compilers should find themselves on familiar territory here because these are all features found in compilers such as Visual Age C++. Add to these Full Speed Debug and Hot Code Replace and you can see features that show that Java is no lacking when considering compiler performance optimization. Because all of this compilation and optimization takes place asynchronously on a background thread, there is little negative impact on the code execution.

Figure 5-8 on page 200 illustrates IBM J9 JIT Compiler optimization.

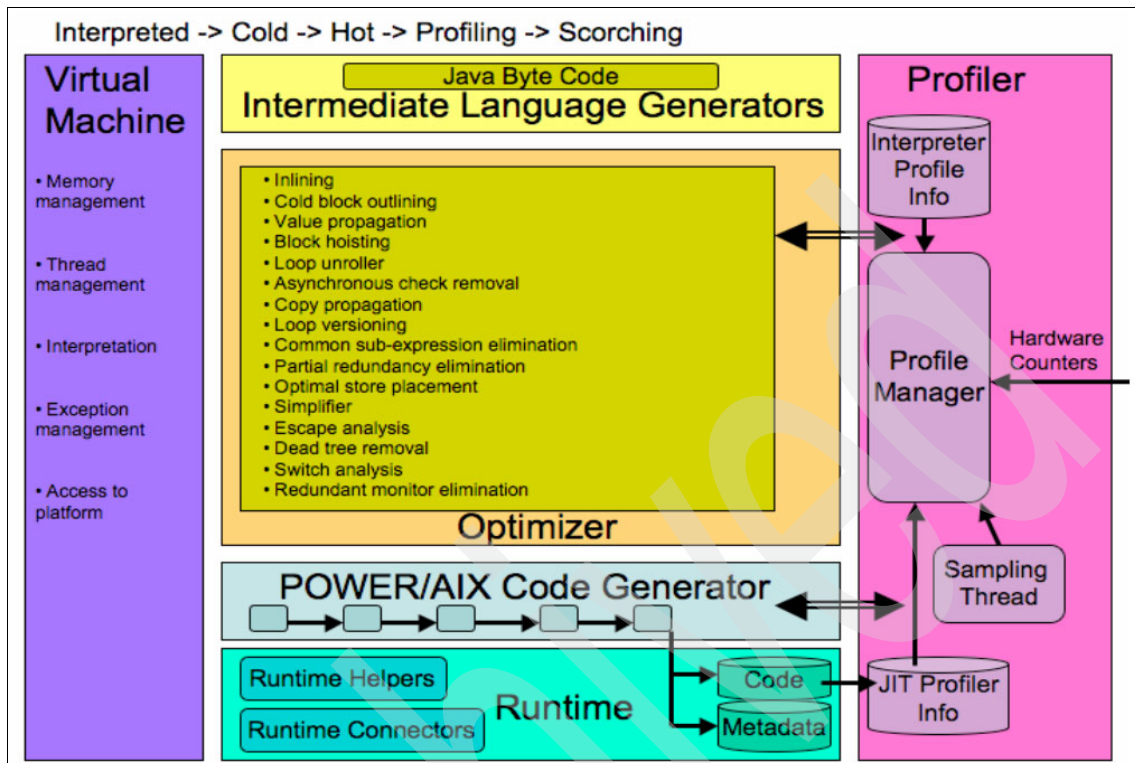


Figure 5-8 IBM J9 JIT Compiler optimization

Note that most methods in WebSphere Application Server are not hot, so they do not get fully optimized. However, methods that are hot and are part of the key working set will transition to the higher compilation levels fairly quickly.

One particular optimizer feature of interest is the use of micropartitioning and how the physical and virtual processors available are handled by the JVM. If there is only one physical processor running one physical thread, then there is no need of protection from multiple accessors to shared memory, resources, and so on, so the optimizer removes all locking to get further performance increases. Therefore, contrary to common perception, using one processor may be relatively faster than using two processors with more than one processor's worth of CPU allocation across them, because all of the synchronization overheads have been removed.

All of these features of the JIT Compiler optimization are used, not merely on the code deployed to WebSphere Application Server, but also to the WebSphere Application Server code itself, with the WebSphere Application Server working set moving towards its most optimal performance over a period of time as the

code is executed in real production usage patterns. If the usage patterns change, the code optimizations in use may also change because this is a dynamic behavior, not a static one.

Shared Classes

The Shared Classes feature improves startup of multiple WebSphere instances and reduces the memory footprint of subsequent instances. Essentially, the Shared Classes feature of the J9 JVM is a cache of commonly used classes across a set of Java Virtual Machine instances. Each class is split into its immutable and changeable parts and stored separately in shared memory and local memory; the ROMClass and the RAMClass respectively.

- ▶ The ROMClass is essentially the byte code and class data that does not change over time, so it can be safely shared across JVM instances using (usually) shared memory segment 0x07000000 (the first shared memory segment).
- ▶ The RAMClass is stored inside the local process memory as usual. Each class file has a signature to identify it. The environment detects if the class file changes on disk during the lifetime of the cache and causes a reload, if necessary.

Note: For more detailed information about using Shared Classes, refer to 6.5, “Using shared classes” on page 325.

The current Java 5 J9 JVM stores the shared classes in shared memory to allow it to be accessed between processes, with a semaphore protecting the shared memory region from multiple JVMs writing to it. The details regarding the names of the shared memory regions and the semaphores can be found in the `/tmp/javasharedresources` file.

This implementation has a downside in that the shared memory is then not available for other processes, so IBM limited WebSphere Application Server usage of shared classes to 50 MB. This implementation is also not WPAR-friendly because shared memory is not accessible between WPARs. The recently released Java 6 J9 implementation uses memory mapped files so, Java6 processes do not have the 50 MB limitation. This will enable a future WebSphere Application Server implementation based on the Java6 JVM to have a far larger class cache (possibly holding all of WebSphere Application Server), which can be shared across WPARs.

The IBM implementation of the standard Java URLClassLoader has been changed to first look in the cache for the ROMClass rather than keep going back to disk with each new JVM instance and reloading each class into local JVM process memory. Thus, startup time of subsequent JVM instances is greatly

improved due to reduced disk activity and the virtual memory footprint of each subsequent JVM instance is reduced because it is loading the ROMClass information from virtual memory directly rather than loading it from disk into its own address space.

There are downsides to this approach, but IBM has addressed them. Any Java code that subclasses the standard `URLClassLoader`, which is common, will benefit from the shared class cache. Code that does not use the shared class cache can be made to use it through the help of IBM provided helper APIs, such as in the `com.ibm.cds_1.0.0.jar` file used within the OSGI/Eclipse framework underlying WebSphere Application Server. Instrumented code also changes, so the cache needs special handling. Protection is needed to ensure that JVMs started by different users can only see what they are meant to see, so multiple caches are supported on a system.

Java parameters are provided within the Java runtime to allow class sharing to be examined, with the **-Xshareclasses** option switching on class sharing and allowing monitoring through its suboptions, and **-Xscmx** to control the cache size for cache creation. Note that **-Xscmx** only has any effect if the **-Xshareclasses** option is used.

The suboptions for the **-Xshareclasses** option allow listing of all the caches in use and their usage count and printing of statistics on the number of classes that are cached.

To create a cache named `myCache` of 10 MB that holds the classes used by Java application class `myClass`, use this command:

```
java -Xshareclasses:name=myCache -Xscmx10M myClass
```

To destroy the cache named `myCache`, use this command:

```
java -Xshareclasses:name=myCache,destroy
```

To print the basic statistics for the cache named `myCache`, use this command:

```
java -Xshareclasses:name=myCache,printStats
```

To print detailed statistics for the cache named `myCache`, use this command:

```
java -Xshareclasses:name=myCache,printAllStats
```

To list all of the caches in use in the system, use this command:

```
java -Xshareclasses:listAllCaches
```

For all options, apart from creating the cache, it is normal to get a message stating: Could not create the Java virtual machine.

What happens with shared classes for WebSphere Application Server on AIX?

Examining the output of the **ps -eaf** command shows that WebSphere Application Server enables class sharing:

```
-Xshareclasses:name=webspherev61_%g,groupAccess,nonFatal -Xscmx50M
```

This syntax creates a 50 MB shared class cache that is accessible by all JVMs started by users in the same primary UNIX group as the user starting this instance. Thus, if root starts WebSphere Application Server (which we do *not* advise), the %g parameter maps to the system group and a cache accessible by all WebSphere Application Server instances started by members of this group is created that is called webspherev61_system, from the preceding command.

If a normal user in the “staff” primary group starts a WebSphere Application Server instance, then a separate shared class cache is created called webspherev61_staff and no ROMClass data is shared between these caches.

Normally, a wasuser account and wasgroup group are used for the application server instances as best practice, so using **java -Xshareclasses:listAllCaches** provides the output shown in Example 5-27.

Example 5-27 listAllCaches output

```
# /usr/IBM/WebSphere/AppServer/java/jre/bin/java
-Xshareclasses:listAllCaches
```

Shared Cache	OS shmid	in use	Last detach time
webspherev61_system	1048576	2	Thu Jun 14 23:31:27 2007
webspherev61_staff	3	1	Thu Jun 14 23:20:17 2007
colin	3	1	Thu Jun 14 23:20:17 2007
webspherev61_wasgroup	1048576	2	Thu Jun 14 23:31:27 2007

Could not create the Java virtual machine.

Example 5-27 shows that there are multiple WebSphere Application Server JVMs in use that have been started by users under different primary groups, and that there are also multiple JVMs using two of the caches. Also, a JVM has created a cache called colin.

Example 5-28 on page 204 displays the contents of one of the caches, which shows how WebSphere Application Server is using the cache.

Example 5-28 Sample cache statistics

```
# /usr/IBM/WebSphere/AppServer/java/jre/bin/java
-Xshareclasses:name=webspherev61_wasgroup,printStats
```

Current statistics for cache "webspherev61_wasgroup":

```
base address      = 0x0700000010000090
end address       = 0x07000000131FFFF0
allocation pointer = 0x0700000012FB1510
```

```
cache size        = 52428656
free bytes        = 1320004
ROMClass bytes    = 50009216
Metadata bytes    = 1099436
Metadata % used   = 2%
```

```
# ROMClasses      = 11397
# Classpaths      = 9
# URLs            = 23
# Tokens          = 0
# Stale classes   = 0
% Stale classes   = 0%
```

Cache is 97% full

Could not create the Java virtual machine.

The output in Example 5-28 shows that the cache is stored in a virtual memory region starting at segment 0x07000000 for the 64-bit JVM process, the first shared memory segment, and with WebSphere Application Server running with no additional applications the 50 MB cache is 97% full with 11397 classes. Example 5-28 illustrates usage of the **printStats** command line option to show that a large part of the cache contains the core Java standard library.

Example 5-29 Sample cache statistics, using printAllStats

```
# /usr/IBM/WebSphere/AppServer/java/jre/bin/java
-Xshareclasses:name=webspherev61_wasgroup,printAllStats
```

Current statistics for cache "webspherev61_wasgroup":

```
1: 0x07000000131FF7A8 CLASSPATH
   /usr/IBM/WebSphere/AppServer/java/jre/lib/ext/ibmorb.jar
```



```

/usr/IBM/WebSphere/AppServer/java/jre/lib/ext/ibmext.jar
/usr/IBM/WebSphere/AppServer/java/jre/lib/vm.jar
/usr/IBM/WebSphere/AppServer/java/jre/lib/core.jar
/usr/IBM/WebSphere/AppServer/java/jre/lib/charsets.jar
/usr/IBM/WebSphere/AppServer/java/jre/lib/graphics.jar
/usr/IBM/WebSphere/AppServer/java/jre/lib/security.jar
/usr/IBM/WebSphere/AppServer/java/jre/lib/ibmpkcs.jar
/usr/IBM/WebSphere/AppServer/java/jre/lib/ibmorb.jar
/usr/IBM/WebSphere/AppServer/java/jre/lib/ibmcfw.jar
/usr/IBM/WebSphere/AppServer/java/jre/lib/ibmorbapi.jar
/usr/IBM/WebSphere/AppServer/java/jre/lib/ibmjcefw.jar
/usr/IBM/WebSphere/AppServer/java/jre/lib/ibmjgssprovider.jar
/usr/IBM/WebSphere/AppServer/java/jre/lib/ibmjseprovider2.jar
/usr/IBM/WebSphere/AppServer/java/jre/lib/ibmjaaslm.jar
/usr/IBM/WebSphere/AppServer/java/jre/lib/ibmcertpathprovider.jar
/usr/IBM/WebSphere/AppServer/java/jre/lib/server.jar
/usr/IBM/WebSphere/AppServer/java/jre/lib/xml.jar
1: 0x07000000131FF748 ROMCLASS: java/lang/Object at 0x0700000010000090.
    Index 2 in classpath 0x07000000131FF7A8
1: 0x07000000131FF708 ROMCLASS: java/lang/J9VMInternals at 0x0700000010000718.
    Index 2 in classpath 0x07000000131FF7A8
1: 0x07000000131FF6C8 ROMCLASS: java/lang/Class at 0x07000000100020E0.
    Index 2 in classpath 0x07000000131FF7A8
1: 0x07000000131FF688 ROMCLASS: java/io/Serializable at 0x0700000010008630.
    Index 3 in classpath 0x07000000131FF7A8
1: 0x07000000131FF648 ROMCLASS: java/lang/reflect/GenericDeclaration at
0x07000000100086F0.
    Index 3 in classpath 0x07000000131FF7A8
1: 0x07000000131FF608 ROMCLASS: java/lang/reflect/Type at 0x0700000010008840.
    Index 3 in classpath 0x07000000131FF7A8
1: 0x07000000131FF5C8 ROMCLASS: java/lang/reflect/AnnotatedElement at
0x07000000100088F8.
    Index 3 in classpath 0x07000000131FF7A8
...

```

Note, in Example 5-28 on page 204, that the cache is 97% full. You might be wondering if a bigger cache would be beneficial for running large application server work loads. This may be correct, but the law of diminishing returns and tradeoffs comes into play in this analysis.

In practical terms, a JVM should be limited to about 3 to 4 GB of heap space to avoid garbage collection performance issues (for which multiple instances is the answer) and the cache eats into this. Too much usage of shared memory also impacts other processes, so it should be used sparingly. However, many

administrators would still want to control the cache size themselves and make their own decisions as to the setting for **-Xscmx**.

You will not find this configuration setting in scripts, properties files, and other configuration elements. The setting is added to the command line dynamically by the `WsServerLauncher` class, which is found in the core `com.ibm.ws.runtime_6.1.0.jar` file.

The configuration for it and other command line options is found in the `aix.systemlaunch.properties` file inside the `com.ibm.ws.runtime_6.1.0.jar` JAR file from the application server plugins directory. Example 5-30 shows the contents of that file.

Example 5-30 aix.systemlaunch.properties file

```
# Default system properties

# Default environment settings
com.ibm.websphere.management.launcher.defaultEnvironment=EXTSHM=ON

# Default JVM options
com.ibm.websphere.management.launcher.defaultJvmOptions=-Declipse.security
-Dosgi.install.area=${WebSphere Application Server_INSTALL_ROOT}
-Dosgi.configuration.area=${USER_INSTALL_ROOT}/configuration -Djava.awt.headless=true
-Dosgi.framework.extensions=com.ibm.cds
-Xshareclasses:name=webspherev61_%g,groupAccess,nonFatal -Xscmx50M
```

The implementation of shared classes is documented to some extent, but can readily be seen through a simple view of the file system. Inside the `/tmp` directory, a subdirectory is created by the shared classes option when it is first run, called `javasharedresources`. Inside this subdirectory is evidence of the caches created and their reliance on UNIX system V features to implement the cache management; that is, shared memory to hold the cache and semaphores to lock the cache access to manage multiple accesses.

The JVM shared classes implementation knows to look in this directory to find the key management information for managing access to the shared classes from multiple JVMs. The contents of this directory are shown in Example 5-31 on page 207, and the process illustrated in Figure 5-9 on page 207.

Note that it is the `ROMClasses` that are stored in the cache (Java byte code), rather than native code itself. The native code is stored and handled by the JIT compiler owned by each JVM.

Example 5-31 javasharedresources directory listing

```
# ls /tmp/javasharedresources
C230D1A64_memory_colin_G01
C230D1A64_memory_webspherev61_wasgroup_G01  C230D1A64_semaphore_webspherev61_system
C230D1A64_memory_webspherev61_staff_G01
C230D1A64_semaphore_colin                    C230D1A64_semaphore_webspherev61_wasgroup
C230D1A64_memory_webspherev61_system_G01    C230D1A64_semaphore_webspherev61_staff
```

Figure 5-9 illustrates the IBM J9 JVM shared classes.

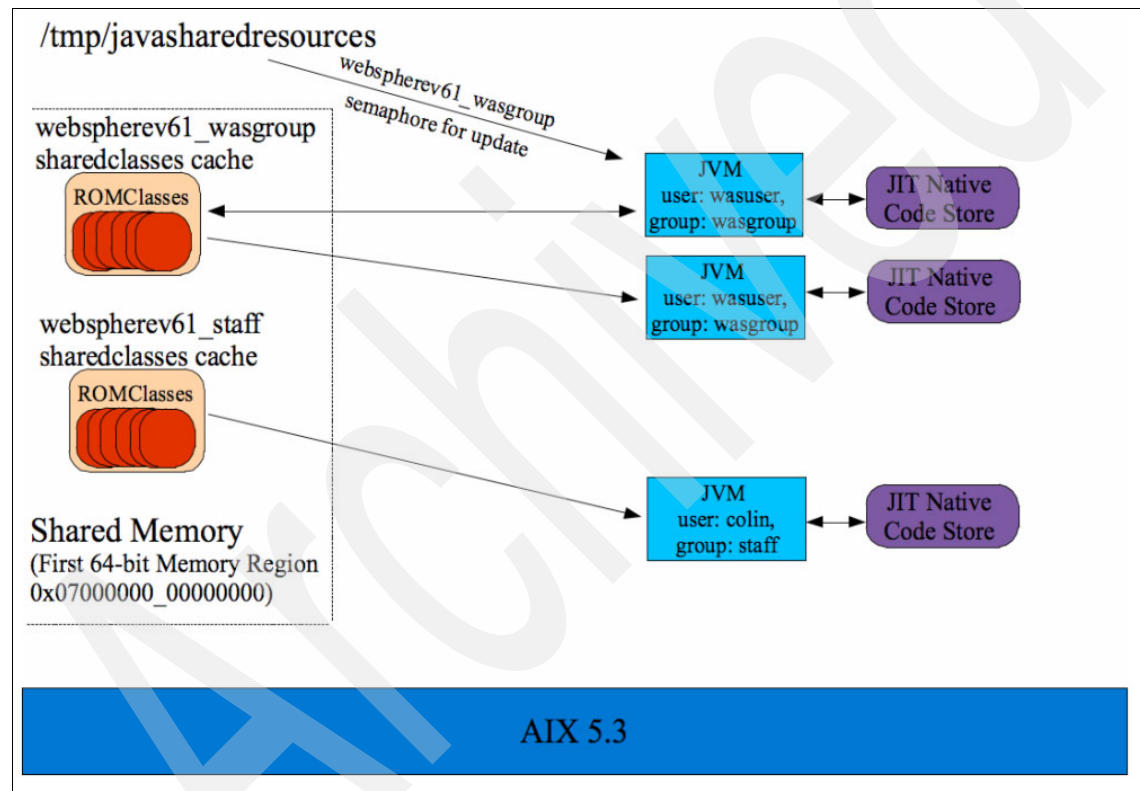


Figure 5-9 IBM J9 JVM shared classes

Now that you have seen how it works, you might be wondering why this is all necessary. The main benefits of using shared classes include greatly improved startup time of each JVM using the shared cache, and a decrease in overall memory resource requirements.

To best illustrate this, we used a memory-constrained single processor POWER3™ machine with 1 GB of RAM; this machine used to struggle running multiple instances of WebSphere 6.0.2.

Example 5-52 on page 256 shows the time taken to load multiple instances, using the root account to avoid any impact of other limits. This example displays the benefit of the cache directly.

Example 5-32 Benefit of using shared classes

```
# ./timeit.sh
Listing caches
JVMSHRC005I No shared class caches available
Could not create the Java virtual machine.
About to start instance 1 at: Sat 16 Jun 21:00:36 2007
ADMU0116I: Tool information is being logged in file

/usr/IBM/WebSphere/AppServer/profiles/sandpit2/logs/server1/startServer.log
ADMU0128I: Starting tool with the sandpit2 profile
ADMU3100I: Reading configuration for server: server1
ADMU3200I: Server launched. Waiting for initialization status.
ADMU3000I: Server server1 open for e-business; process id is 303260
Instance 1 completed loading at Sat 16 Jun 21:04:50 2007
Shared Cache      OS shmid   in use   Last detach time
webspherev61_system 5242880   1       Sat Jun 16 21:01:20 2007

Could not create the Java virtual machine.
About to start instance 2 at: Sat 16 Jun 21:04:51 2007
ADMU0116I: Tool information is being logged in file
      /usr/IBM/WebSphere/AppServer/profiles/sandpit/logs/server1/startServer.log
ADMU0128I: Starting tool with the sandpit profile
ADMU3100I: Reading configuration for server: server1
ADMU3200I: Server launched. Waiting for initialization status.
ADMU3000I: Server server1 open for e-business; process id is 213026
Instance 2 completed loading at Sat 16 Jun 21:08:05 2007
# /usr/IBM/WebSphere/AppServer/java/jre/bin/java -Xshareclasses:listAllCaches
Shared Cache      OS shmid   in use   Last detach time
webspherev61_system 5242880   2       Sat Jun 16 21:05:29 2007

Could not create the Java virtual machine.
```

The output in Example 5-32 shows that the first instance of WebSphere Application Server took 4 minutes and 14 seconds to start. The second instance took 3 minutes and 14 seconds to start. The second instance took 76% of the time of the first instance to start up. This improvement is important for restarting

after a failure when vertical scaling and clustering is used (that is, multiple instances in a single operating system image).

The benefit to memory usage is also significant, and can be seen when looking at the resident set size (RSS) and virtual memory size (SZ) in pages, by using the **ps** command as shown in Example 5-33. In the example, the first instance shows a virtual memory usage of 84124 x 4K pages, or 328 MB. The second instance shows 78380 x 4K pages, or 306 MB.

This 22 MB saving may not seem that significant, but this environment is not running anything but base WebSphere Application Server, so in production use the savings could be enough to allow support for a much larger number of user sessions in a JVM or a few more JVM instances running in the system. In practice, the savings due the shared class cache are usually much larger than is seen here.

Example 5-33 Shared classes: memory usage

```
# ps aux
USER      PID %CPU %MEM    SZ   RSS    TTY STAT   STIME   TIME  COMMAND
...
root    303260   8.1  10.0 84124 84128 pts/0 A    21:01:19   3:19 /usr/IBM/WebSphe
root    213026   7.3   9.0 78380 78384 pts/0 A    21:05:29   2:41 /usr/IBM/WebSphe
...
```

Page sizes

AIX has supported multiple page sizes, configured using the **vmo** command, for a few releases, with 16 MB supported from AIX 5.1, and 64 K and 16 GB pages supported in AIX 5.3 TL04 on a POWER5+™ machine, as well as the more traditional 4 K page size. A 64 K page is for general usage, with the 16 MB and 16 GB options for large memory and high performance applications. As previously, the 4 K page size is the default but now AIX will automatically make use of some 64 K pages when the system could benefit from it.

The J9 JVM can make use of the AIX support for multiple page sizes via the **-X1p<size>** option to use large pages for the heap. This can further improve the memory behavior previously discussed. For the most benefit with minimal effort, use **-X1p64K** to make use of 64 K pages; however, this will only work from AIX 5.3 TL04 onward on a POWER5+ or above, and the benefit varies greatly with the type of system usage. For more information about using large pages, refer to 6.6, “Using large page sizes” on page 329.

Garbage collection

Java was designed to be a “safe” language, with the complexities and risky features of C and C++ removed and strong typing. One of the risky features is

support for developer-controlled memory management, in which a developer who allocates memory is also responsible for freeing it. Java uses a garbage collection mechanism to relieve developers of the responsibility of tidying up memory management.

Java garbage collection is not “magical” and the mechanism can still be overwhelmed by poor coding practices such as creating millions of unnecessary objects (that is, using immutable strings in buffers). The developer has no control over when garbage collection is run, which can lead to “stalls” in a production environment when the garbage collector has to run to make space and compact the heap.

Recognizing that this is a common problem, but also that usage patterns are not all alike, IBM has included the ability to tune the garbage collector for different behavior. However, controlling the garbage collection policy can only assist, but not eliminate, the performance issues inherent in a poorly coded application.

For an in-depth discussion of garbage collection policy and tuning the IBM Java Virtual Machine, see 6.1, “The importance of JVM tuning” on page 304 through 6.4, “Heap sizing” on page 316 in Chapter 6, “Tuning the IBM Java Virtual Machine” on page 303.

DLPAR support

The JVM has classes that extend `java.lang.management` to detect changes in processor and memory resources in a DLPAR environment. Events are triggered from the JVM, which code can subscribe to; see 6.7, “Dynamic logical partitions” on page 333, for more details.

Memory allocation control and malloctype

With 32-bit applications, control of the way the C `malloc` call made use of space on one or more heaps was more important for high performance than it is today. Heaps could be split into buckets, and some applications were dependent on a particular behavior from AIX3.1.

Although things have greatly improved, it is still possible to control the use of the heap using the AIX `malloctype` and `malloptions` environment variables, but performance benefits are likely to be small for all but the most demanding applications. Note that it is the native heap that is controlled by these options rather than the Java heap, but the JVM will respect the settings for the native heap.

Thus, if there is a WebSphere Application Server application that makes significant use of native library code calls that will result in numerous memory allocations, then it may be worth experimenting with the `malloctype=Watson`

setting in particular that is available in AIX 5.3 to supplement the `malloctype=default` allocator commonly used for 64-bit code.

Threading and thread management

In the past, threading and thread management has been an issue on AIX, because the JVM has always used UNIX 98 User Mode threads for maximum performance. This could result in problems in communicating with other processes that were also using their own user mode threads, because there was no context for control of synchronization.

The default for AIX was always `AIXTHREAD_SCOPE=P` and `AIXTHREAD_MNRRATIO=8` for a mapping of 8 user mode threads to a single kernel mode thread. With the current release, the mode is simple and equivalent to kernel mode threads; that is, `AIXTHREAD_SCOPE=S` and `AIXTHREAD_MNRRATIO=1:1`. These settings are now implicit within the JVM and are ignored by the JVM if set. From AIX 6.1 onward, this is the default setting for the operating system anyway.

Signal management

It is possible to disable some of the non-essential non-exception UNIX signals by using the `-Xrs` (reduce signals) setting. This can have a minimal performance benefit but in practical terms, it should be left alone unless instructed otherwise by IBM support. Similarly, the `-Xnosigcatch` and `-Xsigcatch` settings can be used to control what signals are caught and ignored by the JVM, and therefore WebSphere Application Server. But again, these settings should be left at their defaults unless you are instructed otherwise.

5.3.3 IBM J9 JVM internal implementation

The implementation of the IBM J9 Java Virtual Machine is like any other virtual machine for a modern operating system. It consists of a set of executables, a set of native shared objects that provide most of the implementation at runtime, and a set of Java code to map the provided standard Java classes and interfaces to the underlying Java virtual machine code.

Thus, the JVM consists of C/C++ code (most commonly) and Java code that communicates with it. In the JVM Java code, the implementer is faced with a choice of what to implement in Java code and what to implement in C/C++ code. The C/C++ is precompiled, but it carries the risk of misused pointers that can cause a failure. This is the same issue faced by developers who use the Java Native Interface (JNI) to call into C/C++ shared objects from Java, or vice versa.

When implementing an API, a choice is made between having the API as a thin wrapper for a native method implemented in some other language (usually C or

C++), or using more elemental Java APIs to build the required functionality. For example, the HTTP functionality implemented by the `java.net.URLConnection` class can be implemented entirely in Java using the `java.net.Socket` class, or it can be implemented directly using the underlying host system functionality exposed via a Java Native Interface (JNI) library call.

To produce a JNI-based API, a class must be implemented in Java, an underlying shared object must be implemented using an external language, and this external code calls the operating system APIs, as illustrated in Figure 5-10. Ultimately, the core functionality of the Java socket interface is implemented in the JVM via calls to the host operating system, but there is a variance in where the line is drawn. The tuning of options at the operating system level and the effect they have vary with where this line is drawn.

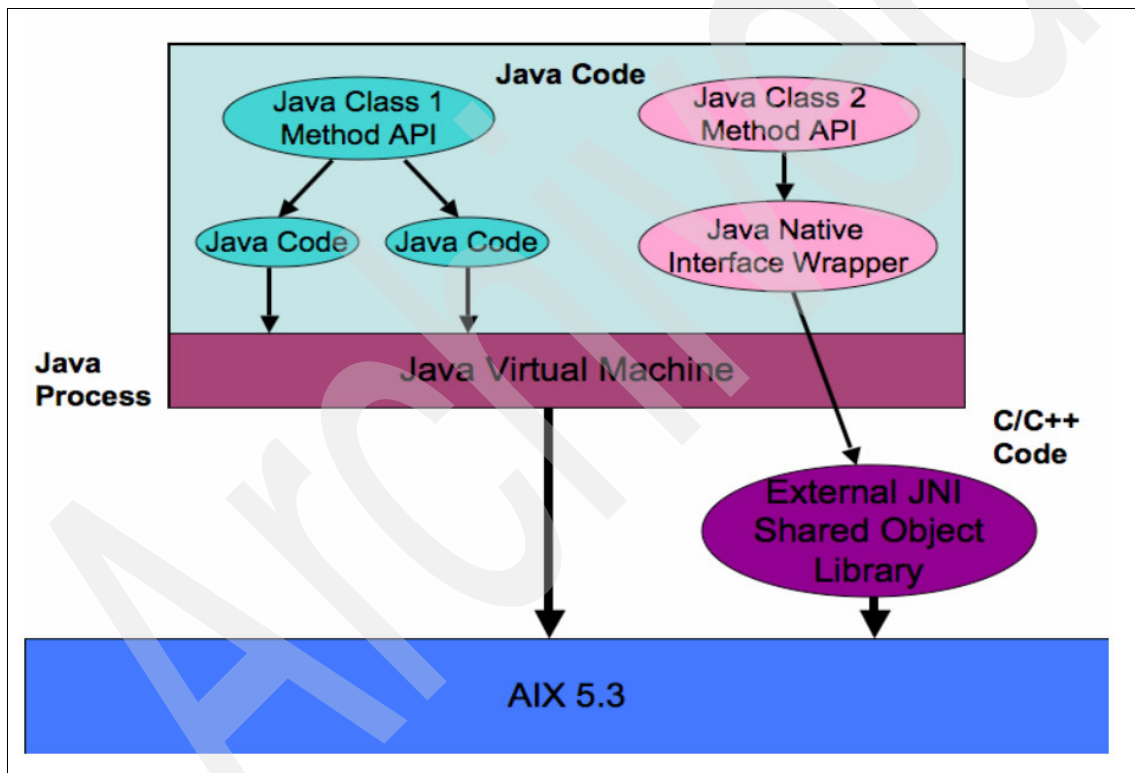


Figure 5-10 JNI and the IBM J9 JVM

In earlier versions of WebSphere Application Server, a great deal of low level external C/C++ code was accessed via JNI and shared objects, most of which was derived from the earlier IBM Component Broker product. Today, most WebSphere Application Server code is pure Java, but that which is not pure Java

is informative for understanding how WebSphere Application Server operates on AIX.

Note the distinction here between low level JVM/Java standard library code (which is to be expected for mapping Java sockets, file system, and such code down to the bare operating system calls), and that specific to WebSphere Application Server, which is a Java application.

Using a default installation of WebSphere Application Server, we have a directory of /usr/IBM/WebSphere/AppServer. Under the java directory, the IBM J9 JVM used by WebSphere Application Server can be found, with a jre/bin directory containing a significant amount of shared objects implementing the JVM functionality itself. Here we find libj9jit23.so for the JIT compilation; libj9gc23.so for the garbage collector; libj9shr23.so for the shared classes feature; and libj9thr23.so for thread management, and libraries containing the code underlying the Java library (including libnet.a for networking support; libnio.a for socket nio; and so on). This is common to all Java applications using this JVM; see Example 5-34.

Example 5-34 Shared objects and libraries listing

```
# ls -l *.a *.so
-rwxr-xr-x 1 root system 1968329 06 Oct 2006 libawt.a
-rwxr-xr-x 1 root system 656910 06 Oct 2006 libcmm.a
-rwxr-xr-x 1 root system 65145 06 Oct 2006 libdbx_j9.so
-rwxr-xr-x 1 root system 293731 06 Oct 2006 libdcpr.a
-rwxr-xr-x 1 root system 29977 06 Oct 2006 libdt_socket.a
-rwxr-xr-x 1 root system 893132 06 Oct 2006 libfontmanager.a
-rwxr-xr-x 1 root system 252822 06 Oct 2006 libhprof.a
-rwxr-xr-x 1 root system 121730 06 Oct 2006 libinstrument.a
-rwxr-xr-x 1 root system 119209 06 Oct 2006 libivertl23.so
-rwxr-xr-x 1 root system 132369 06 Oct 2006 libj9bcv23.so
-rwxr-xr-x 1 root system 527015 06 Oct 2006 libj9dbg23.so
-rwxr-xr-x 1 root system 213120 06 Oct 2006 libj9dmp23.so
-rwxr-xr-x 1 root system 224328 06 Oct 2006 libj9dyn23.so
-rwxr-xr-x 1 root system 1061815 06 Oct 2006 libj9gc23.so
-rwxr-xr-x 1 root system 144744 06 Oct 2006 libj9gcchk23.so
-rwxr-xr-x 1 root system 12309 06 Oct 2006 libj9hookable23.so
-rwxr-xr-x 1 root system 1329718 06 Oct 2006 libj9jextract.so
-rwxr-xr-x 1 root system 6898716 06 Oct 2006 libj9jit23.so
-rwxr-xr-x 1 root system 456205 06 Oct 2006 libj9jitd23.so
-rwxr-xr-x 1 root system 198450 06 Oct 2006 libj9jpi23.so
-rwxr-xr-x 1 root system 290970 06 Oct 2006 libj9jvmti23.so
-rwxr-xr-x 1 root system 376447 06 Oct 2006 libj9prt23.so
-rwxr-xr-x 1 root system 25264 06 Oct 2006 libj9rdbi23.so
-rwxr-xr-x 1 root system 196690 06 Oct 2006 libj9shr23.so
```

-rwxr-xr-x	1	root	system	95750	06	Oct	2006	libj9thr23.so
-rwxr-xr-x	1	root	system	113369	06	Oct	2006	libj9trc23.so
-rwxr-xr-x	1	root	system	154247	06	Oct	2006	libj9ute23.so
-rwxr-xr-x	1	root	system	1166348	06	Oct	2006	libj9vm23.so
-rwxr-xr-x	1	root	system	253485	06	Oct	2006	libj9vrb23.so
-rwxr-xr-x	1	root	system	88708	06	Oct	2006	libj9zlib23.so
-rwxr-xr-x	1	root	system	4643	06	Oct	2006	libjaas.a
-rwxr-xr-x	1	root	system	315497	06	Oct	2006	libjava.a
-rwxr-xr-x	1	root	system	31147	06	Oct	2006	libjava_crw_demo.a
-rwxr-xr-x	1	root	system	2750	06	Oct	2006	libjawt.a
-rwxr-xr-x	1	root	system	1250321	06	Oct	2006	libjclscar_23.so
-rwxr-xr-x	1	root	system	378018	06	Oct	2006	libjdpw.a
-rwxr-xr-x	1	root	system	346076	06	Oct	2006	libjpeg.a
-rwxr-xr-x	1	root	system	178958	06	Oct	2006	libjnick.so
-rwxr-xr-x	1	root	system	109393	06	Oct	2006	libjpkcs11.so
lrwxrwxrwx	1	root	system	10	06	Oct	2006	libjsig.a -> libjsig.so
-rwxr-xr-x	1	root	system	16925	06	Oct	2006	libjsig.so
-rwxr-xr-x	1	root	system	477446	06	Oct	2006	libjsound.a
-rwxr-xr-x	1	root	system	4407	06	Oct	2006	libmanagement.a
-rwxr-xr-x	1	root	system	179678	06	Oct	2006	libnet.a
-rwxr-xr-x	1	root	system	80491	06	Oct	2006	libnio.a
-rwxr-xr-x	1	root	system	2144	06	Oct	2006	librmi.a
-rwxr-xr-x	1	root	system	114869	06	Oct	2006	libunpack.a
-rwxr-xr-x	1	root	system	12620	06	Oct	2006	libwrappers.a
-rwxr-xr-x	1	root	system	132664	06	Oct	2006	libzip.a

Contrast the list in shown Example 5-34 on page 213 with the shared objects specific to WebSphere Application Server itself, as shown in Example 5-35. These shared objects can be found, by default, in the /usr/IBM/WebSphere/AppServer/bin directory.

Example 5-35 Shared objects specific to WebSphere Application Server

```
# cd /usr/IBM/WebSphere/AppServer/bin
# ls -l *.so
```

-rwxr-xr-x	1	root	system	6556	06	Oct	2006	libgetClasses.so
-rwxr-xr-x	1	root	system	48530	06	Oct	2006	libibmaiodbg.so
-rwxr-xr-x	1	root	system	28317	06	Oct	2006	libibmaio.so
-rwxr-xr-x	1	root	system	60175	06	Oct	2006	libNativeFile.so
-rwxr-xr-x	1	root	system	24472	06	Oct	2006	libpmiJvmPiProfiler.so
-rwxr-xr-x	1	root	system	30381	06	Oct	2006	libpmiJvmTiProfiler.so
-rwxr-xr-x	1	root	system	11860	06	Oct	2006	libSelector.so
-rwxr-xr-x	1	root	system	16250	06	Oct	2006	libSystemData.so
-rwxr-xr-x	1	root	system	21720	06	Oct	2006	libUnixRegistryImpl.so

We will look at these in more detail later, but these implementations are for asynchronous I/O, native security platform security implementation, native platform process management and integration, performance tool integration, and for IIOP/CORBA communications integration. All of these are areas where an application server implementer might choose for a native implementation to integrate better with the platform, or to achieve performance improvement because of the specific role of the application server within an enterprise.

To understand more about what is going on under the covers, and to understand how to interpret problems using the IBM Java tools, a JAVADUMP or JAVACORE file can be used. In many ways this is like a Java equivalent of an AIX core file. It contains thread, heap, garbage collection, lock, and stack information.

You can use the administration console or system console to generate a JAVADUMP file from WebSphere Application Server. However, from an AIX command line, you can generate a Java dump file of any running Java process by using **kill -QUIT n** to send a SIGQUIT signal, where **n** is the process ID.

The options for JAVADUMP are controlled by the **JAVA_DUMP_OPTS** environment variable. **IBM_JAVACOREDIR** controls the location it is written to. For the Java5 JVM used with WebSphere Application Server, the JAVADUMP signal will produce output that is similar to Example 5-36.

Example 5-36 Sample JAVADUMP

```
NULL
-----
OSECTION      TITLE subcomponent dump routine
NULL          =====
1TISIGINFO    Dump Event "user" (00004000) received
1TIDATETIME   Date:                2007/12/01 at 14:40:39
1TIFILENAME   Javacore filename:
/usr/IBM/WebSphere/AppServer/profiles/testwas/javacore.20071201.144039.123064.txt
NULL
-----
OSECTION      GPINFO subcomponent dump routine
NULL          =====
2XHOSLEVEL    OS Level           : AIX 5.3
2XHCPUS       Processors -
3XHCPUARCH    Architecture      : ppc64
3XNUMCPUS     How Many         : 1
NULL
```

```

1XHERROR2      Register dump section only produced for SIGSEGV, SIGILL or SIGFPE.
NULL
NULL
-----
OSECTION      ENVINFO subcomponent dump routine
NULL          =====
1CIJAVAVERSION J2RE 5.0 IBM J9 2.3 AIX ppc64-64 build j9vmap6423-20060504
1CIVMVERSION  VM build 20060501_06428_BHdSMr
1CIJITVERSION  JIT enabled - 20060428_1800_r8
1CIRUNNINGAS  Running as a standalone JVM
1CICMDLINE    /usr/IBM/WebSphere/AppServer/java/bin/java -Declipse.security
-Dwas.status.socket=34571 -Dosgi.install.area=/usr/IBM/WebSphere/AppServer
-Dosgi.configuration.area=/usr/IBM/WebSphere/AppServer/profiles/testwas/configuration
-Djava.awt.headless=true -Dosgi.framework.extensions=com.ibm.cds
-Xshareclasses:name=webspherev61_%g,groupAccess,nonFatal -Xscmx50M -
...
com.ibm.wsspi.bootstrap.WSPreLauncher -nosplash -application
com.ibm.ws.bootstrap.WSLauncher com.ibm.ws.runtime.WsServer
/usr/IBM/WebSphere/AppServer/profiles/testwas/config spock44pNode01Cell
spock44pNode02 server1
1CIJAVAHOMEDIR Java Home Dir:    /usr/IBM/WebSphere/AppServer/java/jre
1CIJAVADLLDIR  Java DLL Dir:    /usr/IBM/WebSphere/AppServer/java/jre/bin
1CISYSCP       Sys Classpath:
/usr/IBM/WebSphere/AppServer/java/jre/lib/ext/ibmorb.jar;/usr/IBM/WebSphere/AppServer
/java/jre/lib/ext/ibmext.jar;/usr/IBM/WebSphere/AppServer/java/jre/lib/vm.jar;/usr/IB
M/WebSphere/AppServer/java/jre/lib/core.jar;/usr/IBM/WebSphere/AppServer/java/jre/lib
/charsets.jar;/usr/IBM/WebSphere/AppServer/java/jre/lib/graphics.jar;/usr/IBM/WebSph
ere/AppServer/java/jre/lib/security.jar;/usr/IBM/WebSphere/AppServer/java/jre/lib/ibmp
kcs.jar;/usr/IBM/WebSphere/AppServer/java/jre/lib/ibmorb.jar;/usr/IBM/WebSphere/AppSe
rver/java/jre/lib/ibmcfw.jar;/usr/IBM/WebSphere/AppServer/java/jre/lib/ibmorbapi.jar;
/usr/IBM/WebSphere/AppServer/java/jre/lib/ibmjcefw.jar;/usr/IBM/WebSphere/AppServer/j
ava/jre/lib/ibmjgssprovider.jar;/usr/IBM/WebSphere/AppServer/java/jre/lib/ibmjssprov
ider2.jar;/usr/IBM/WebSphere/AppServer/java/jre/lib/ibmjaaslm.jar;/usr/IBM/WebSphere/
AppServer/java/jre/lib/ibmcertpathprovider.jar;/usr/IBM/WebSphere/AppServer/java/jre/
lib/server.jar;/usr/IBM/WebSphere/AppServer/java/jre/lib/xml.jar;
1CIUSERARGS   UserArgs:
2CIUSERARG      -Xjcl:jclscar_23
2CIUSERARG
-Dcom.ibm.oti.vm.bootstrap.library.path=/usr/IBM/WebSphere/AppServer/java/jre/bin
2CIUSERARG
-Dsun.boot.library.path=/usr/IBM/WebSphere/AppServer/java/jre/bin
2CIUSERARG
-Djava.library.path=/usr/IBM/WebSphere/AppServer/java/jre/bin:/usr/IBM/WebSphere/AppS
erver/java/jre/bin/j9vm:/usr/IBM/WebSphere/AppServer/java/jre/bin:/usr/IBM/WebSphere/
AppServer/bin:/usr/lib

```

```

2CIUSERARG          -Djava.home=/usr/IBM/WebSphere/AppServer/java/jre
...
Djava.class.path=/usr/IBM/WebSphere/AppServer/profiles/testwas/properties:/usr/IBM/We
bSphere/AppServer/properties:/usr/IBM/WebSphere/AppServer/lib/startup.jar:/usr/IBM/We
bSphere/AppServer/lib/bootstrap.jar:/usr/IBM/WebSphere/AppServer/lib/j2ee.jar:/usr/IB
M/WebSphere/AppServer/lib/implproxy.jar:/usr/IBM/WebSphere/AppServer/lib/urlprotocols.j
ar:/usr/IBM/WebSphere/AppServer/deploytool/itp/batchboot.jar:/usr/IBM/WebSphere/AppSe
rver/deploytool/itp/batch2.jar:/usr/IBM/WebSphere/AppServer/java/lib/tools.jar
2CIUSERARG          vfprintf
2CIUSERARG          _port_library 0x09001000A01DAD60
2CIUSERARG          -Xdump
NULL
1CIJVMMI            JVM Monitoring Interface (JVMMI)
NULL               -----
2CIJVMMIOFF         [not available]
NULL
NULL
-----
OSECTION            MEMINFO subcomponent dump routine
NULL               -----
1STHEAPFREE          Bytes of Heap Space Free: 247e330
1STHEAPALLOC         Bytes of Heap Space Allocated: 655cc00
NULL
1STSEGTYP            Internal Memory
NULL               segment          start          alloc          end
type      bytes
1STSEGMENT           00000001104E7870 0000000115AA3450 0000000115AB344C 0000000115AB3450
01000040 10000
...
1STSEGMENT           00000001104E7448 000000011056B810 000000011057B810 000000011057B810
01000040 10000
NULL
1STSEGTYP            Object Memory
NULL               segment          start          alloc          end
type      bytes
1STSEGMENT           00000001104E7AA8 0700000000000000 070000000655CC00 070000000655CC00
00000009 655cc00
NULL
1STSEGTYP            Class Memory
NULL               segment          start          alloc          end
type      bytes
1STSEGMENT           00000001160FE678 0000000115E2E890 0000000115E2EB70 0000000115E2EB70
00010040 2e4
...

```

```

1STSEGMENT      00000001104E82E8 0700000010000090 07000000130F8C70 07000000131FFFF0
00020104 31fff70
NULL
1STSEGTYPE      JIT Code Cache
NULL            segment          start          alloc          end
type      bytes
1STSEGMENT      000000011051D108 000000011057B9B0 000000011057B9B0 0000000110D7B9B0
00000068 800000
NULL
1STSEGTYPE      JIT Data Cache
NULL            segment          start          alloc          end
type      bytes
1STSEGMENT      000000011056B628 0000000110DB31B0 00000001110804F4 00000001115B31B0
00000048 800000
NULL
1STGCHTYPE      GC History
3STHSTTYPE      14:28:57:601484000 GMT j9mm.53 -   GlobalGC end: workstackoverflow=0
overflowcount=0 weakrefs=17573 soft=2098 phantom=4 finalizers=239 newspace=0/0
oldspace=50488384/106286080 loa=0/0
3STHSTTYPE      14:28:57:601144000 GMT j9mm.61 -   Class unloading end
3STHSTTYPE      14:28:57:593173000 GMT j9mm.66 -   Classloader unload
3STHSTTYPE      14:28:57:592714000 GMT j9mm.66 -   Classloader unload
3STHSTTYPE      14:28:57:592308000 GMT j9mm.60 -   Class unloading start
3STHSTTYPE      14:28:57:592282000 GMT j9mm.57 -   Sweep end
3STHSTTYPE      14:28:57:575557000 GMT j9mm.56 -   Sweep start
3STHSTTYPE      14:28:57:575530000 GMT j9mm.55 -   Mark end
3STHSTTYPE      14:28:57:68626000 GMT j9mm.54 -   Mark start
3STHSTTYPE      14:28:57:68410000 GMT j9mm.52 -   GlobalGC start: weakrefs=17872
soft=2120 phantom=36 finalizers=656 globalcount=34 scavengedcount=0
3STHSTTYPE      14:26:01:88564000 GMT j9mm.53 -   GlobalGC end: workstackoverflow=0
overflowcount=0 weakrefs=16865 soft=2021 phantom=4 finalizers=233 newspace=0/0
oldspace=57044640/106286080 loa=0/0
...
3STHSTTYPE      14:20:18:221639000 GMT j9mm.52 -   GlobalGC start: weakrefs=9900
soft=1605 phantom=0 finalizers=151 globalcount=4 scavengedcount=0
3STHSTTYPE      14:19:56:798427000 GMT j9mm.53 -   GlobalGC end: workstackoverflow=0
overflowcount=0 weakrefs=8166 soft=1585 phantom=0 finalizers=103 newspace=0/0
oldspace=44678944/52428800 loa=2621440/2621440
NULL
NULL
-----
OSECTION        LOCKS subcomponent dump routine
NULL            =====
NULL
1LKPOOLINFO     Monitor pool info:

```

```

2LKPOOLTOTAL      Current total number of monitors: 65
NULL
1LKMONPOOLDUMP Monitor Pool Dump (flat & inflated object-monitors):
2LKMONINUSE       sys_mon_t:0x0000000111CB7BE0 infl_mon_t: 0x0000000111CB7C30:
3LKMONOBJECT
org/eclipse/osgi/framework/eventmgr/EventManager$EventThread@0700000000334740/0700000
000334758: <unowned>
3LKNOTIFYQ        Waiting to be notified:
3LKWAITNOTIFY     "Framework Event Dispatcher" (0x0000000111DAC100)
2LKMONINUSE       sys_mon_t:0x0000000111CB7D90 infl_mon_t: 0x0000000111CB7DE0:
3LKMONOBJECT
org/eclipse/osgi/framework/eventmgr/EventManager$EventThread@0700000000335FD8/0700000
000335FF0: <unowned>
3LKNOTIFYQ        Waiting to be notified:
3LKWAITNOTIFY     "Start Level Event Dispatcher" (0x0000000111D41B00)
...
2LKMONINUSE       sys_mon_t:0x0000000113EE6F40 infl_mon_t: 0x0000000113EE6F90:
3LKMONOBJECT      java/lang/Object@0700000000C9F7E0/0700000000C9F7F8: owner
"server.startup : 0" (0x000000011457EC00), entry count 1
NULL
1LKREGMONDUMP JVM System Monitor Dump (registered monitors):
2LKREGMON         Thread global lock (0x0000000110012400): <unowned>
2LKREGMON         NLS hash table lock (0x0000000110012490): <unowned>
2LKREGMON         portLibrary_j9sig_sync_monitor lock (0x0000000110012520):
<unowned>
2LKREGMON         portLibrary_j9sig_asynch_reporter_shutdown_monitor lock
(0x00000001100125B0): <unowned>
2LKREGMON         portLibrary_j9sig_async_monitor lock (0x0000000110012640):
<unowned>
2LKREGMON         Hook Interface lock (0x00000001100126D0): <unowned>
2LKREGMON         &(vm->bytecodeTableMutex) lock (0x0000000110012760): <unowned>
2LKREGMON         MM_SublistPool lock (0x00000001100127F0): <unowned>
...
2LKREGMON         MM_GCExtensions::gcStats lock (0x0000000110012D00): <unowned>
2LKREGMON         &vm->verboseStateMutex lock (0x0000000110012D90): <unowned>
2LKREGMON         VM thread list lock (0x0000000110012E20): Flat locked by
"[osthread]" (0x0000000110014D48), entry count 1
2LKREGMON         VM exclusive access lock (0x0000000110012EB0): <unowned>
2LKREGMON         VM Runtime flags Mutex lock (0x0000000110012F40): <unowned>
...
2LKREGMON         VM mem segment list lock (0x0000000110013840): <unowned>
2LKREGMON         FinalizeListManager lock (0x00000001100138D0): <unowned>
2LKREGMON         &(jvmtiData->mutex) lock (0x0000000110013960): <unowned>
2LKREGMON         BCVD verifier lock (0x00000001100139F0): <unowned>
2LKREGMON         Thread public flags mutex lock (0x0000000110013A80): <unowned>

```

```

2LKREGMON      &newSem->pfmInfo.sem lock (0x0000000110013B10): <unowned>
...
3LKNOTIFYQ      Waiting to be notified:
3LKWAITNOTIFY    "HAManager.thread.pool : 1" (0x00000001136A8F00)
2LKREGMON      Thread public flags mutex lock (0x000000011204AF60): <unowned>
2LKREGMON      Thread public flags mutex lock (0x000000011204AFF0): <unowned>
2LKREGMON      JVM_RawMonitor lock (0x000000011204B110): <unowned>
...
2LKREGMON      Thread public flags mutex lock (0x0000000115356A50): <unowned>
NULL
NULL
-----
OSECTION      THREADS subcomponent dump routine
NULL          =====
NULL
1XMCURTHDINFO  Current Thread Details
NULL          -----
NULL
1XMTHDINFO     All Thread Details
NULL          -----
NULL
2XMFULLTHDDUMP Full thread dump J9SE VM (J2RE 5.0 IBM J9 2.3 AIX ppc64-64 build
20060501_06428_BHdSMr, native threads):
3XMTHREADINFO  "P=916005:0=0:CT" (TID:0x00000001104E5500,
sys_thread_t:0x0000000110014868, state:CW, native ID:0x0000000000FB0F9) prio=5
4XESTACKTRACE  at java/lang/Thread.sleep(Native Method)
4XESTACKTRACE  at java/lang/Thread.sleep(Thread.java:923)
4XESTACKTRACE  at com/ibm/ws/runtime/WsServerImpl.main(WsServerImpl.java:479)
4XESTACKTRACE  at com/ibm/ws/runtime/WsServer.main(WsServer.java:59)
4XESTACKTRACE  at sun/reflect/NativeMethodAccessorImpl.invoke0(Native Method)
4XESTACKTRACE  at
sun/reflect/NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:64)
4XESTACKTRACE  at
sun/reflect/DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
4XESTACKTRACE  at java/lang/reflect/Method.invoke(Method.java:615)
4XESTACKTRACE  at
com/ibm/wsspi/bootstrap/WSLauncher.launchMain(WSLauncher.java:183)
4XESTACKTRACE  at com/ibm/wsspi/bootstrap/WSLauncher.main(WSLauncher.java:90)
4XESTACKTRACE  at com/ibm/wsspi/bootstrap/WSLauncher.run(WSLauncher.java:72)
4XESTACKTRACE  at
org/eclipse/core/internal/runtime/PlatformActivator$1.run(PlatformActivator.java:226)
4XESTACKTRACE  at
org/eclipse/core/runtime/adaptor/EclipseStarter.run(EclipseStarter.java:376)
4XESTACKTRACE  at
org/eclipse/core/runtime/adaptor/EclipseStarter.run(EclipseStarter.java:163)

```



```

4XESTACKTRACE      at sun/reflect/NativeMethodAccessorImpl.invoke0(Native Method)
4XESTACKTRACE      at
sun/reflect/NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:64)
4XESTACKTRACE      at
sun/reflect/DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
4XESTACKTRACE      at java/lang/reflect/Method.invoke(Method.java:615)
4XESTACKTRACE      at
org/eclipse/core/launcher/Main.invokeFramework(Main.java:334)
4XESTACKTRACE      at org/eclipse/core/launcher/Main.basicRun(Main.java:278)
4XESTACKTRACE      at org/eclipse/core/launcher/Main.run(Main.java:973)
4XESTACKTRACE      at
com/ibm/wsspi/bootstrap/WSPreLauncher.launchEclipse(WSPreLauncher.java:245)
4XESTACKTRACE      at
com/ibm/wsspi/bootstrap/WSPreLauncher.main(WSPreLauncher.java:73)
3XMTTHREADINFO     "JIT Compilation Thread" (TID:0x00000001116BC300,
sys_thread_t:0x0000000110015228, state:CW, native ID:0x00000000010C019) prio=11
...
3XMTTHREADINFO     "Shared TCPChannel NonBlocking Accept Thread"
(TID:0x0000000115F02C00, sys_thread_t:0x0000000115562648, state:R, native
ID:0x00000000014608D) prio=5
4XESTACKTRACE      at sun/nio/ch/PollArrayWrapper.poll0(Native Method)
4XESTACKTRACE      at sun/nio/ch/PollArrayWrapper.poll(PollArrayWrapper.java:144)
4XESTACKTRACE      at
sun/nio/ch/PollSelectorImpl.doSelect(PollSelectorImpl.java:115)
4XESTACKTRACE      at
sun/nio/ch/SelectorImpl.lockAndDoSelect(SelectorImpl.java:99)
4XESTACKTRACE      at sun/nio/ch/SelectorImpl.select(SelectorImpl.java:110)
4XESTACKTRACE      at
com/ibm/ws/tcp/channel/impl/ChannelSelector.run(ChannelSelector.java:158)
4XESTACKTRACE      at java/lang/Thread.run(Thread.java:797)
NULL
-----
OSECTION          CLASSES subcomponent dump routine
NULL              =====
1CLTEXTCLLOS      Classloader summaries
1CLTEXTCLLS       2345678:
1=primordial,2=extension,3=shareable,4=middleware,5=system,6=trusted,7=application,8=
delegating
2CLTEXTCLLOADER   p---st-- Loader *System*(0x07000000000094D0)
3CLNMBRLOADEDLIB  Number of loaded libraries 5
3CLNMBRLOADEDCL   Number of loaded classes 3311
2CLTEXTCLLOADER   -x--st-- Loader
sun/misc/Launcher$ExtClassLoader(0x070000000001B220), Parent
*none*(0x0000000000000000)
3CLNMBRLOADEDLIB  Number of loaded libraries 0

```

```

3CLNMBRLOADEDCL  Number of loaded classes 85
2CLTEXTCLLOADER  -----ta- Loader
sun/misc/Launcher$AppClassLoader(0x07000000000222D0), Parent
sun/misc/Launcher$ExtClassLoader(0x070000000001B220)
3CLNMBRLOADEDLIB  Number of loaded libraries 0
3CLNMBRLOADEDCL  Number of loaded classes 174
2CLTEXTCLLOADER  -----t-- Loader
org/eclipse/core/launcher/Main$StartupClassLoader(0x070000000006D8E8), Parent
sun/misc/Launcher$AppClassLoader(0x07000000000222D0)
3CLNMBRLOADEDLIB  Number of loaded libraries 0
3CLNMBRLOADEDCL  Number of loaded classes 332
2CLTEXTCLLOADER  -----t-- Loader
org/eclipse/core/runtime/internal/adaptor/ContextFinder(0x0700000000078FE8), Parent
sun/misc/Launcher$AppClassLoader(0x07000000000222D0)
3CLNMBRLOADEDLIB  Number of loaded libraries 0
3CLNMBRLOADEDCL  Number of loaded classes 0
2CLTEXTCLLOADER  -----t-- Loader
org/eclipse/osgi/framework/adaptor/core/CDSBundleClassLoader(0x07000000000336DF8),
Parent sun/misc/Launcher$AppClassLoader(0x07000000000222D0)
3CLNMBRLOADEDLIB  Number of loaded libraries 0
3CLNMBRLOADEDCL  Number of loaded classes 109
2CLTEXTCLLOADER  -----t-- Loader
org/eclipse/osgi/framework/adaptor/core/CDSBundleClassLoader(0x0700000000058E650),
Parent sun/misc/Launcher$AppClassLoader(0x07000000000222D0)
3CLNMBRLOADEDLIB  Number of loaded libraries 0
3CLNMBRLOADEDCL  Number of loaded classes 7
2CLTEXTCLLOADER  -----t-- Loader
org/eclipse/osgi/framework/adaptor/core/CDSBundleClassLoader(0x070000000005901E8),
Parent sun/misc/Launcher$AppClassLoader(0x07000000000222D0)
3CLNMBRLOADEDLIB  Number of loaded libraries 0
3CLNMBRLOADEDCL  Number of loaded classes 28
...
1CLTEXTCLLIB  ClassLoader loaded libraries
2CLTEXTCLLIB  Loader *System*(0x07000000000094D0)
3CLTEXTLIB    /usr/IBM/WebSphere/AppServer/java/jre/bin/java
3CLTEXTLIB    /usr/IBM/WebSphere/AppServer/java/jre/bin/jciscar_23
3CLTEXTLIB    /usr/IBM/WebSphere/AppServer/java/jre/bin/zip
3CLTEXTLIB    /usr/IBM/WebSphere/AppServer/java/jre/bin/net
3CLTEXTLIB    /usr/IBM/WebSphere/AppServer/java/jre/bin/nio
2CLTEXTCLLIB  Loader
org/eclipse/osgi/framework/adaptor/core/CDSBundleClassLoader(0x0700000000081F688)
3CLTEXTLIB    /usr/IBM/WebSphere/AppServer/bin/Ws60ProcessManagement
3CLTEXTLIB    /usr/IBM/WebSphere/AppServer/bin/SystemData
3CLTEXTLIB    /usr/IBM/WebSphere/AppServer/bin/getClasses
3CLTEXTLIB    /usr/IBM/WebSphere/AppServer/bin/ibmaio

```

```

...
2CLTEXTCLLOAD Loader
org/eclipse/osgi/framework/adaptor/core/CDSBundleClassLoader(0x07000000010E3E48)
3CLTEXTCLASS com/ibm/mqservices/Trace(0x00000001138FE3E8)
3CLTEXTCLASS com/ibm/mqservices/CallStackTrace(0x00000001138FF240)
3CLTEXTCLASS com/ibm/mq/MQPoolServices(0x00000001138FF4A8)
3CLTEXTCLASS com/ibm/mq/MQSimpleConnectionManager$PSAdapter(0x00000001138FF9F8)
3CLTEXTCLASS com/ibm/mq/MQEnvironment$1(0x00000001138FFCD8)
3CLTEXTCLASS com/ibm/mq/MQPoolServicesEvent(0x00000001138FFF28)
3CLTEXTCLASS com/ibm/mq/StoredManagedConnection(0x0000000113863FB0)
3CLTEXTCLASS com/ibm/mq/ManagedConnectionStore(0x00000001138647A0)
3CLTEXTCLASS com/ibm/mq/PoolScavenger(0x0000000113864CD8)
3CLTEXTCLASS com/ibm/mq/MQSimpleConnectionManager(0x000000011379E9F0)
3CLTEXTCLASS com/ibm/mq/MQPoolServicesEventListener(0x000000011379F500)
3CLTEXTCLASS com/ibm/mq/MQConnectionEventListener(0x000000011379F6B8)
3CLTEXTCLASS com/ibm/mq/MQException(0x0000000113861248)
3CLTEXTCLASS com/ibm/mq/MQConnectionManager(0x0000000112C72658)
3CLTEXTCLASS com/ibm/mq/MQEnvironment(0x000000011379B6E8)

```

```

...
2CLTEXTCLLOAD Loader sun/reflect/DelegatingClassLoader(0x0700000005F85108)
3CLTEXTCLASS
sun/reflect/GeneratedSerializationConstructorAccessor70(0x0000000115E2E8D0)
NULL

```

```

-----
OSECTION      Javdump End section
NULL          ----- END OF DUMP
-----

```

JAVADUMP creates large files (a significant amount of repeated information has been left out of Example 5-36 on page 215). However, JAVADUMPs contain helpful pieces of information that promote your understanding of IBM J9 JVM on AIX implementation. Detailed explanations for interpreting a JAVADUMP can be found in the IBM Java Virtual Machine Infocenter documentation:

http://publib.boulder.ibm.com/infocenter/javasdk/v5r0/index.jsp?topic=/com.ibm.java.doc.diagnostics.50/diag/tools/javadump_tags_info.html

The first part of the file displays environmental and startup information for the Java part of the process, as shown in Example 5-37. This data tells you the operating system version used, the JVM version, the platform you are running on, and how the JAVADUMP file was generated.

Example 5-37 JAVADUMP: environmental and startup information

```

NULL
-----

```

```

OSECTION      TITLE subcomponent dump routine
NULL          =====
1TISIGINFO    Dump Event "user" (00004000) received
1TIDATETIME   Date:                2007/12/01 at 14:40:39
1TIFILENAME   Javacore filename:
/usr/IBM/WebSphere/AppServer/profiles/testwas/javacore.20071201.144039.123064.txt
NULL
-----
OSECTION      GPINFO subcomponent dump routine
NULL          =====
2XHOSLEVEL    OS Level           : AIX 5.3
2XHCPUS       Processors -
3XHCPUARCH    Architecture      : ppc64
3XNUMCPUS     How Many         : 1
NULL
1XHERROR2     Register dump section only produced for SIGSEGV, SIGILL or SIGFPE.
NULL
NULL
-----
OSECTION      ENVINFO subcomponent dump routine
NULL          =====
1CIJAVAVERSION J2RE 5.0 IBM J9 2.3 AIX ppc64-64 build j9vmap6423-20060504
1CIVMVERSION   VM build 20060501_06428_BHdSMr
1CIJITVERSION  JIT enabled - 20060428_1800_r8
1CIRUNNINGAS   Running as a standalone JVM

```

Next, you see the command line and arguments used to start the JVM, as displayed in Example 5-38 on page 225. This data is useful, particularly with WebSphere Application Server, because the complete environment is often applied by scripts or properties files and this shows what actual values are in use.

Examining the memory, you can see how the JVM works on AIX. You can see the size of the heap and how much of it is being used. Then you can see the AIX 64-bit memory model in use with each of the segments, and what their contents are.

Note the distinction between the heap, the object, the class, and the JIT code cache and JIT data cache memory segments. The JIT code and data segments are used to produce the code that actually runs.

Finally, you can see how the garbage collection is applied and when it is being applied (in this case, with a standard mark and sweep system).

Example 5-38 JAVADUMP: command line and arguments used to start the JVM

```

OSECTION      MEMINFO subcomponent dump routine

NULL          =====
1STHEAPFREE    Bytes of Heap Space Free: 247e330
1STHEAPALLOC   Bytes of Heap Space Allocated: 655cc00
NULL
1STSEGTYP      Internal Memory
NULL          segment      start      alloc      end
type      bytes
1STSEGMENT    00000001104E7870 0000000115AA3450 0000000115AB344C 0000000115AB3450
01000040 10000
1STSEGMENT    0000000111B4B518 0000000115A83410 0000000115A933F4 0000000115A93410
01000040 10000
...
1STSEGMENT    00000001104E7448 000000011056B810 000000011057B810 000000011057B810
01000040 10000
NULL
1STSEGTYP      Object Memory
NULL          segment      start      alloc      end
type      bytes
1STSEGMENT    00000001104E7AA8 0700000000000000 070000000655CC00 070000000655CC00
00000009 655cc00
NULL
1STSEGTYP      Class Memory
NULL          segment      start      alloc      end
type      bytes
1STSEGMENT    00000001160FE678 0000000115E2E890 0000000115E2EB70 0000000115E2EB70
00010040 2e4
1STSEGMENT    00000001160FE5E0 00000001161B5470 00000001161B5860 00000001161B6B50
00020040 16e0
...
1STSEGMENT    00000001104E82E8 0700000010000090 07000000130F8C70 07000000131FFFF0
00020104 31fff70
NULL
1STSEGTYP      JIT Code Cache
NULL          segment      start      alloc      end
type      bytes
1STSEGMENT    000000011051D108 000000011057B9B0 000000011057B9B0 0000000110D7B9B0
00000068 800000
NULL
1STSEGTYP      JIT Data Cache

```

NULL	segment	start	alloc	end
type bytes				
1STSEGMENT	000000011056B628	0000000110DB31B0	00000001110804F4	00000001115B31B0
00000048 800000				
NULL				
1STGCHTYPE	GC History			
3STHSTTYPE	14:28:57:601484000	GMT j9mm.53 -	GlobalGC end: workstackoverflow=0	
	overflowcount=0 weakrefs=17573 soft=2098 phantom=4 finalizers=239 newspace=0/0			
	oldspace=50488384/106286080 loa=0/0			
3STHSTTYPE	14:28:57:601144000	GMT j9mm.61 -	Class unloading end	
3STHSTTYPE	14:28:57:593173000	GMT j9mm.66 -	ClassLoader unload	
3STHSTTYPE	14:28:57:592714000	GMT j9mm.66 -	ClassLoader unload	
3STHSTTYPE	14:28:57:592308000	GMT j9mm.60 -	Class unloading start	
3STHSTTYPE	14:28:57:592282000	GMT j9mm.57 -	Sweep end	
3STHSTTYPE	14:28:57:575557000	GMT j9mm.56 -	Sweep start	
3STHSTTYPE	14:28:57:575530000	GMT j9mm.55 -	Mark end	
3STHSTTYPE	14:28:57:68626000	GMT j9mm.54 -	Mark start	
3STHSTTYPE	14:28:57:68410000	GMT j9mm.52 -	GlobalGC start: weakrefs=17872	
	soft=2120 phantom=36 finalizers=656 globalcount=34 scavengcount=0			
3STHSTTYPE	14:26:01:88564000	GMT j9mm.53 -	GlobalGC end: workstackoverflow=0	
	overflowcount=0 weakrefs=16865 soft=2021 phantom=4 finalizers=233			

Example 5-39 on page 227 displays the underlying thread synchronization at work and the Java monitors. This is important for identifying when a deadlock has occurred between threads. Note that on a single CPU machine, the internal synchronization is optimized away, although in a virtualized environment the real CPU to virtual CPU mapping means that synchronization code is always present.

In the following data, pay particular attention to the Eclipse/OSGI framework event manager because this is part of the “hidden” infrastructure that underlies WebSphere Application Server.

Also note that there are threads and associated monitors for monitoring ports, strings, JNI code, garbage collection, and native code. However, of particular interest here are the JIT and JVM thread-related monitors.

There are monitors that are used by threads used for profiling and sampling, and others for monitoring JIT queues. This shows the underlying asynchronous model used by the IBM J9 JVM for moving code from its basic byte code format to its “hot” native code implementation by recompiling and reoptimizing it on a background thread and moving it to the native code cache used by the JVM itself.

```

NULL
-----
0SECTION      LOCKS subcomponent dump routine
NULL          =====
NULL
1LKPOOLINFO    Monitor pool info:
2LKPOOLTOTAL   Current total number of monitors: 65
NULL
1LKMONPOOLDUMP Monitor Pool Dump (flat & inflated object-monitors):
2LKMONINUSE     sys_mon_t:0x0000000111CB7BE0 infl_mon_t: 0x0000000111CB7C30:
3LKMONOBJECT
org/eclipse/osgi/framework/eventmgr/EventManager$EventThread@0700000000334740/0700000
000334758: <unowned>
3LKNOTIFYQ      Waiting to be notified:
3LKWAITNOTIFY   "Framework Event Dispatcher" (0x0000000111DAC100)
2LKMONINUSE     sys_mon_t:0x0000000111CB7D90 infl_mon_t: 0x0000000111CB7DE0:
3LKMONOBJECT
org/eclipse/osgi/framework/eventmgr/EventManager$EventThread@0700000000335FD8/0700000
000335FF0: <unowned>
3LKNOTIFYQ      Waiting to be notified:
3LKWAITNOTIFY   "Start Level Event Dispatcher" (0x0000000111D41B00)
...
3LKMONOBJECT    java/lang/Object@0700000000C9F7E0/0700000000C9F7F8: owner
"server.startup : 0" (0x000000011457EC00), entry count 1
NULL
1LKREGMONDUMP  JVM System Monitor Dump (registered monitors):
2LKREGMON      Thread global lock (0x0000000110012400): <unowned>
2LKREGMON      NLS hash table lock (0x0000000110012490): <unowned>
2LKREGMON      portLibrary_j9sig_sync_monitor lock (0x0000000110012520):
<unowned>
2LKREGMON      portLibrary_j9sig_asynch_reporter_shutdown_monitor lock
(0x00000001100125B0): <unowned>
2LKREGMON      portLibrary_j9sig_async_monitor lock (0x0000000110012640):
<unowned>
2LKREGMON      Hook Interface lock (0x00000001100126D0): <unowned>
2LKREGMON      &(vm->bytecodeTableMutex) lock (0x0000000110012760): <unowned>
2LKREGMON      MM_SublistPool lock (0x00000001100127F0): <unowned>
2LKREGMON      MM_SublistPool lock (0x0000000110012880): <unowned>
...
2LKREGMON      MM_WorkPackets::inputList lock (0x0000000110012C70): <unowned>
2LKREGMON      MM_GCExtensions::gcStats lock (0x0000000110012D00): <unowned>
2LKREGMON      &vm->verboseStateMutex lock (0x0000000110012D90): <unowned>

```

```

2LKREGMON      VM thread list lock (0x0000000110012E20): Flat locked by
"[osthread]" (0x0000000110014D48), entry count 1
2LKREGMON      VM exclusive access lock (0x0000000110012EB0): <unowned>
2LKREGMON      VM Runtime flags Mutex lock (0x0000000110012F40): <unowned>
2LKREGMON      VM Extended method block flags Mutex lock (0x0000000110012FD0):
<unowned>
2LKREGMON      VM bind native lock (0x0000000110013060): <unowned>
2LKREGMON      VM class loader blocks lock (0x00000001100130F0): <unowned>
2LKREGMON      VM class table lock (0x0000000110013180): <unowned>
2LKREGMON      VM string table lock (0x0000000110013210): <unowned>
2LKREGMON      VM segment lock (0x00000001100132A0): <unowned>
2LKREGMON      VM JNI frame lock (0x0000000110013330): <unowned>
...
2LKREGMON      &newSem->pfmInfo.sem lock (0x0000000110013C30): <unowned>
2LKREGMON      JIT-MonitorTableMonitor lock (0x0000000110013CC0): <unowned>
2LKREGMON      JIT-MemoryAllocMonitor lock (0x0000000110013D50): <unowned>
2LKREGMON      JIT-ClassUnloadMonitor lock (0x0000000110013DE0): <unowned>
2LKREGMON      Hook Interface lock (0x0000000110013E70): <unowned>
2LKREGMON      JIT-jitConfig->mutex lock (0x0000000110013F00): <unowned>
2LKREGMON      JIT-AssumptionTableMutex lock (0x0000000110013F90): <unowned>
2LKREGMON      VM mem segment list lock (0x0000000110014020): <unowned>
2LKREGMON      VM mem segment list lock (0x00000001100140B0): <unowned>
2LKREGMON      ActivationTableMutex lock (0x0000000110014140): <unowned>
2LKREGMON      JIT-CompilationQueueMonitor lock (0x00000001100141D0): <unowned>
3LKNOTIFYQ      Waiting to be notified:
3LKWAITNOTIFY    "JIT Compilation Thread" (0x00000001116BC300)
2LKREGMON      JIT-CodeCacheListMutex lock (0x0000000110014260): <unowned>
2LKREGMON      JIT-CodeCacheMonitor-?? lock (0x00000001100142F0): <unowned>
2LKREGMON      Thread public flags mutex lock (0x0000000110014380): <unowned>
2LKREGMON      ValueProfilingMutex lock (0x0000000110014410): <unowned>
...
2LKREGMON      &refreshMutex lock (0x00000001100146E0): <unowned>
2LKREGMON      JIT sampling thread lock (0x0000000110014770): <unowned>
2LKREGMON      management fields lock lock (0x00000001118A4E00): <unowned>
2LKREGMON      &vm->managementData->notificationMonitor lock
(0x00000001118A4E90): <unowned>
2LKREGMON      JIT-InterpreterProfilingMonitor lock (0x00000001118A4F20):
<unowned>
2LKREGMON      JIT-QueueSlotMonitor-0 lock (0x00000001118A4FB0): <unowned>
2LKREGMON      JIT-QueueSlotMonitor-1 lock (0x00000001118A5040): <unowned>
2LKREGMON      JIT-QueueSlotMonitor-2 lock (0x00000001118A50D0): <unowned>
2LKREGMON      JIT-QueueSlotMonitor-3 lock (0x00000001118A5160): <unowned>
2LKREGMON      Thread public flags mutex lock (0x00000001118A51F0): <unowned>
2LKREGMON      JVM_RawMonitor lock (0x00000001118A5280): <unowned>
2LKREGMON      JVM_RawMonitor lock (0x00000001118A5310): <unowned>

```


2LKREGMON	JVM_RawMonitor lock (0x00000001118A53A0): <unowned>
2LKREGMON	JIT-QueueSlotMonitor-4 lock (0x00000001118A5430): <unowned>
2LKREGMON	JIT-QueueSlotMonitor-5 lock (0x00000001118A54C0): <unowned>

Example 5-40 displays the threads within the JVM and how they are being mapped down to the underlying native platform threads. It also shows the IDs to map to external tool pictures of the thread usage. The relative thread priorities can be useful in identifying thread starvation issues. With each thread, you also have the thread stack information.

Example 5-40 JAVADUMP: thread within the JVM, and mapping to platform threads

```

1XMTDINFO      All Thread Details
NULL          -----
NULL
2XMFULLTHDDUMP Full thread dump J9SE VM (J2RE 5.0 IBM J9 2.3 AIX ppc64-64 build
20060501_06428_BHdSMr, native threads):
3XMTHREADINFO  "P=916005:0=0:CT" (TID:0x00000001104E5500,
sys_thread_t:0x0000000110014868, state:CW, native ID:0x0000000000FB0F9) prio=5
4XESTACKTRACE  at java/lang/Thread.sleep(Native Method)
4XESTACKTRACE  at java/lang/Thread.sleep(Thread.java:923)
4XESTACKTRACE  at com/ibm/ws/runtime/WsServerImpl.main(WsServerImpl.java:479)
4XESTACKTRACE  at com/ibm/ws/runtime/WsServer.main(WsServer.java:59)
4XESTACKTRACE  at sun/reflect/NativeMethodAccessorImpl.invoke0(Native Method)
4XESTACKTRACE  at
sun/reflect/NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:64)
4XESTACKTRACE  at
sun/reflect/DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
4XESTACKTRACE  at java/lang/reflect/Method.invoke(Method.java:615)
4XESTACKTRACE  at
com/ibm/wsspi/bootstrap/WSLauncher.launchMain(WSLauncher.java:183)
4XESTACKTRACE  at com/ibm/wsspi/bootstrap/WSLauncher.main(WSLauncher.java:90)
4XESTACKTRACE  at com/ibm/wsspi/bootstrap/WSLauncher.run(WSLauncher.java:72)
4XESTACKTRACE  at
org/eclipse/core/internal/runtime/PlatformActivator$1.run(PlatformActivator.java:226)
4XESTACKTRACE  at
org/eclipse/core/runtime/adaptor/EclipseStarter.run(EclipseStarter.java:376)
4XESTACKTRACE  at
org/eclipse/core/runtime/adaptor/EclipseStarter.run(EclipseStarter.java:163)
4XESTACKTRACE  at sun/reflect/NativeMethodAccessorImpl.invoke0(Native Method)
4XESTACKTRACE  at
sun/reflect/NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:64)
4XESTACKTRACE  at
sun/reflect/DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
4XESTACKTRACE  at java/lang/reflect/Method.invoke(Method.java:615)

```

```

4XESTACKTRACE      at
org/eclipse/core/launcher/Main.invokeFramework(Main.java:334)
4XESTACKTRACE      at org/eclipse/core/launcher/Main.basicRun(Main.java:278)
4XESTACKTRACE      at org/eclipse/core/launcher/Main.run(Main.java:973)
4XESTACKTRACE      at
com/ibm/wsspi/bootstrap/WSPreLauncher.launchEclipse(WSPreLauncher.java:245)
4XESTACKTRACE      at
com/ibm/wsspi/bootstrap/WSPreLauncher.main(WSPreLauncher.java:73)
3XMTHREADINFO      "JIT Compilation Thread" (TID:0x00000001116BC300,
sys_thread_t:0x0000000110015228, state:CW, native ID:0x00000000010C019) prio=11

```

Finally, Example 5-41 displays classloader information. Although this is primarily a Java feature, you can observe the IBM shared classes implementation at work with the WebSphere Application Server Eclipse/OSGI implementation by noting what has been loaded by the CDSBundleClassLoader.

Example 5-41 JAVADUMP: classloader information

```

OSECTION          CLASSES subcomponent dump routine
NULL              =====
1CLTEXTCLLOS      Classloader summaries
1CLTEXTCLLSS      12345678:
1=primordial,2=extension,3=shareable,4=middleware,5=system,6=trusted,7=application,8=
delegating
2CLTEXTCLLOADERp---st-- Loader *System*(0x07000000000094D0)
3CLNMBRLOADEDLIBNumber of loaded libraries 5
3CLNMBRLOADEDCLNumber of loaded classes 3311
...
1CLTEXTCLLIB      ClassLoader loaded libraries
2CLTEXTCLLIB      Loader *System*(0x07000000000094D0)
3CLTEXTLIB        /usr/IBM/WebSphere/AppServer/java/jre/bin/java
3CLTEXTLIB        /usr/IBM/WebSphere/AppServer/java/jre/bin/jclscar_23
3CLTEXTLIB        /usr/IBM/WebSphere/AppServer/java/jre/bin/zip
3CLTEXTLIB        /usr/IBM/WebSphere/AppServer/java/jre/bin/net
3CLTEXTLIB        /usr/IBM/WebSphere/AppServer/java/jre/bin/nio
2CLTEXTCLLIB      Loader
org/eclipse/osgi/framework/adaptor/core/CDSBundleClassLoader(0x070000000081F688)
3CLTEXTLIB        /usr/IBM/WebSphere/AppServer/bin/ws60ProcessManagement
3CLTEXTLIB        /usr/IBM/WebSphere/AppServer/bin/SystemData
3CLTEXTLIB        /usr/IBM/WebSphere/AppServer/bin/getClasses
3CLTEXTLIB        /usr/IBM/WebSphere/AppServer/bin/ibmaio
1CLTEXTCLLOD      ClassLoader loaded classes
2CLTEXTCLLOAD      Loader *System*(0x07000000000094D0)
3CLTEXTCLASS      javax/management/remote/rmi/RMIConnectorServer(0x00000001164387C8)
3CLTEXTCLASS      javax/management/remote/rmi/RMIServer(0x0000000116439AA0)

```

```

3CLTEXTCLASS    javax/management/remote/rmi/RMIServerImpl(0x0000000116439CB8)
3CLTEXTCLASS    javax/management/remote/rmi/RMIIOPServerImpl(0x000000011643AB80)
3CLTEXTCLASS    sun/rmi/server/LoaderHandler$LoaderEntry(0x000000011643AFD0)
3CLTEXTCLASS    sun/rmi/server/LoaderHandler$LoaderKey(0x000000011643B218)
3CLTEXTCLASS    sun/rmi/server/LoaderHandler$2(0x000000011643B4B8)
3CLTEXTCLASS    com/ibm/oti/shared/SharedClassPermissionCollection(0x000000011643B778)
3CLTEXTCLASS    sun/rmi/server/LoaderHandler$1(0x000000011643BB60)
3CLTEXTCLASS    javax/management/remote/rmi/_RMIServerImpl_Tie(0x000000011643BED0)
...
2CLTEXTCLLOAD   Loader
org/eclipse/core/launcher/Main$StartupClassLoader(0x07000000006D8E8)
3CLTEXTCLASS    org/eclipse/osgi/internal/module/VersionHashMap(0x0000000111F97F00)
3CLTEXTCLASS    org/eclipse/osgi/internal/module/CyclicDependencyHashMap(0x0000000111F98458)
3CLTEXTCLASS    org/eclipse/osgi/internal/module/GroupingChecker(0x0000000111F98768)
3CLTEXTCLASS    org/eclipse/osgi/internal/module/ResolverExport(0x0000000111F98E78)
3CLTEXTCLASS    org/eclipse/osgi/internal/module/ResolverImport(0x0000000111F99498)
3CLTEXTCLASS    org/eclipse/osgi/internal/module/BundleConstraint(0x0000000111F99C18)
3CLTEXTCLASS    [Lorg/eclipse/osgi/internal/module/ResolverImport; (0x0000000111F9A200)
3CLTEXTCLASS    [Lorg/eclipse/osgi/internal/module/ResolverExport; (0x0000000111F9A350)
3CLTEXTCLASS    [Lorg/eclipse/osgi/internal/module/BundleConstraint; (0x0000000111F9A4A0)
3CLTEXTCLASS    [Lorg/eclipse/osgi/internal/module/VersionSupplier; (0x0000000111F9A5F0)
3CLTEXTCLASS    [Lorg/eclipse/osgi/internal/module/ResolverBundle; (0x0000000111F9A740)
3CLTEXTCLASS    org/osgi/framework/BundlePermission(0x0000000111F9A8C0)
3CLTEXTCLASS    org/osgi/framework/BundlePermissionCollection(0x0000000111F9AEF0)
...
3CLTEXTCLASS    org/osgi/framework/Bundle(0x0000000111B82DD8)
3CLTEXTCLASS    org/eclipse/core/runtime/adaptor/EclipseStarter$InitialBundle(0x0000000111B83200)
3CLTEXTCLASS    org/eclipse/osgi/framework/log/FrameworkLog(0x0000000111C2EB98)
3CLTEXTCLASS    org/eclipse/core/runtime/adaptor/EclipseStarter(0x0000000111B68448)
2CLTEXTCLLOAD   Loader
org/eclipse/core/runtime/internal/adaptor/ContextFinder(0x0700000000078FE8)
2CLTEXTCLLOAD   Loader
org/eclipse/osgi/framework/adaptor/core/CDSBundleClassLoader(0x0700000000336DF8)
3CLTEXTCLASS    org/eclipse/core/internal/runtime/PlatformURLMetaConnection(0x000000011214E390)
...
3CLTEXTCLASS    sun/reflect/GeneratedSerializationConstructorAccessor70(0x0000000115E2E8D0)
NULL
-----
OSECTION        Javadump End section

```

Example 5-41 on page 230 illustrates the mapping between real underlying processors and the virtual CPUs as seen by the operating system image in a virtualized environment. Remember that when the HyperVisor is loaded (always with a POWER5 processor and above), the CPU is always virtualized even if it is to whole CPU values.

The standard configuration for a newer POWER 5 or POWER 6 architecture machine is to run with Symmetric Multi-Threading (SMT) enabled and micropartitioning enabled. With SMT, multiple threads can be run concurrently under many circumstances on a single CPU. With micropartitioning, the CPU can be split into a number of virtual CPUs in portions of 0.1 or 0.01 of the processing power of a real CPU.

The JVM understands these and performs the appropriate optimizations. However, with these choices the processor cache is important because the real processor is split between virtual processors that may be running more than one operating system image concurrently, which incurs a small overhead.

Some applications, such as those running rules engines, are CPU-intensive, so dedicated processor partitions should be used for these. However, for standard Web applications, the overheads are insignificant when overall performance is considered so SMT and micropartitioning should be enabled, with the partitions running uncapped with appropriate entitlements discovered by testing. SMT can get an additional 30 percent performance out of a partition when enabled.

This is discussed here because processor folding optimizations from AIX 5.3 ML03 onward release unused virtual processor cycles back to the HyperVisor, and this can be defeated through misconfiguration and the JVM optimizations can be lost.

For example, adding .1 increments above whole CPU allocations to a partition leads to additional synchronization overheads and the loss of cache locality that counteract any benefit of the extra CPU allocation. (This was also explained earlier in the context of JVM architecture.)

5.4 WebSphere Application Server architecture

The following sections explain the various components that comprise the WebSphere Application Server product architecture.

5.4.1 Overview of WebSphere Application Server architecture

The WebSphere Application Server software architecture can be viewed in several ways: as bundles of Java libraries; as a group of cooperating services implemented by different components; or as a layered software architecture.

To understand the WebSphere Application Server environment and how it exploits the platform on which it runs, all three perspectives must be examined and understood. However, in this document we focus on layering and how WebSphere Application Server can be split into independently viewable parts. Each part has its own behavior and its own opportunities for tuning.

WebSphere Application Server is shipped in a number of different types of packages that are aimed at different uses, yet each has a common core and common code across platforms.

- ▶ The standalone WebSphere Application Server is designed primarily for environments where high availability and resilience are not required, and it is standalone. For truly stateless application environments, it is possible to produce a highly available infrastructure with multiple machines running standalone WebSphere Application Server. However, important features such as maintenance of Web user sessions mean that very few applications are truly stateless.
- ▶ WebSphere Application Server Network Deployment (WebSphere Application Server ND) edition builds upon the base product, primarily adding support for high availability and resilience. It is this version that we concentrate on here. This package includes deployment manager as an external component for managing a cluster, and the HAManager and Data Replication Services (DRS) to keep the cluster working in sync.
- ▶ WebSphere Application Server Extended Deployment (WebSphere Application Server XD) builds upon the Network Deployment edition of WebSphere Application Server, but with this package, the extensions are more external to the product. For example, WebSphere Application Server ND adds the high availability extensions into the core product. In contrast, WebSphere Application Server XD uses an external intelligent router and other application level code to support more intelligent clustering.

These editions have the same base functionality and code at their core, with the additional features added as separate deployable components or as features that are deployed onto the core product via the WebSphere Application Server Profile Manager.

In enterprise class POWER/AIX environments, the core WebSphere Application Server ND or WebSphere Application Server XD products are likely to be used. Because the WebSphere Application Server- XD product adds to the

WebSphere Application Server ND product externally, you need to understand the WebSphere Application Server and WebSphere Application Server ND picture.

When WebSphere Application Server is installed, most of the code is deployed as a set of Java jar libraries that are written as Eclipse/OSGI plug-ins that extend a core runtime. The core runtime is shipped in a file called `com.ibm.ws.runtime_6.1.0.jar` that is started by scripts and an Eclipse/OSGI runtime itself.

Eclipse/OSGI is comprised of extensible components called plug-ins or bundles, so the runtime provides a base on which other plug-ins or bundles build by extending “extension points” in the base. In the runtime, for example, there is code for generic support for containers, threading, and communications, and other code is shipped to extend this either in other parts of the WebSphere Application Server runtime, or in separate plug-ins or bundles. Thus, WebSphere Application Server consists of a set of JAR files that are Eclipse/OSGI plug-ins or bundles.

Much of this description can be seen with an understanding of Eclipse and OSGI plug-ins and bundles. Extensions, services, and extension points are described in a `plugin.xml` file or `MANIFEST.MF` file that is included inside the JAR file or directory holding the code. A JAR file is essentially the same as a zipped tar file, and you can find the `plugin.xml` or `MANIFEST.MF` file within it. If you open the `plugin.xml` file for `com.ibm.ws.runtime_6.1.0.jar`, you can see the base description for the startup of WebSphere Application Server and the extension points that it offers for other code to extend, such as the Web container implementation code in the `com.ibm.ws.webcontainer_2.0.0.jar` file.

For the Web container, the base WebSphere runtime in `com.ibm.ws.runtime_6.1.0.jar` offers the thread pool management, transport channel management, service registry, and base functional utility code that underlies the Web container, and the `com.ibm.ws.webcontainer_2.0.0.jar` file contains the Web container implementation code that sets the policy for behavior of the Web container functionality, and adds functions like the Apache Jasper JSP compiler functions and JSF code.

This type of implementation makes sense given that Web services make use of the same core functionality, so reusability is obtained without mixing the different concerns of usage. Note that the `com.ibm.ws.webcontainer_2.0.0.jar` file contains its own `plugin.xml` file that lists the extensions and extension points it uses and makes available, and the `MANIFEST.MF` file lists the OSGI Bundle Activator that it contains that allows the Eclipse/OSGI runtime to initialize it.

It is important to note that the XML shows that different code is run for the iSeries® (AS/400®) and zSeries® (S/390®) within the WebSphere Application

Server startup to accommodate differences in these platforms. But, for all other platforms, the core Java code is the same after the startup has completed and set up the appropriate environment. The AS/400 and S/390 tags in some of the sections of the plugin.xml file highlight these differences.

Figure 5-11 illustrates how WebSphere Application Server is extended to add new features.

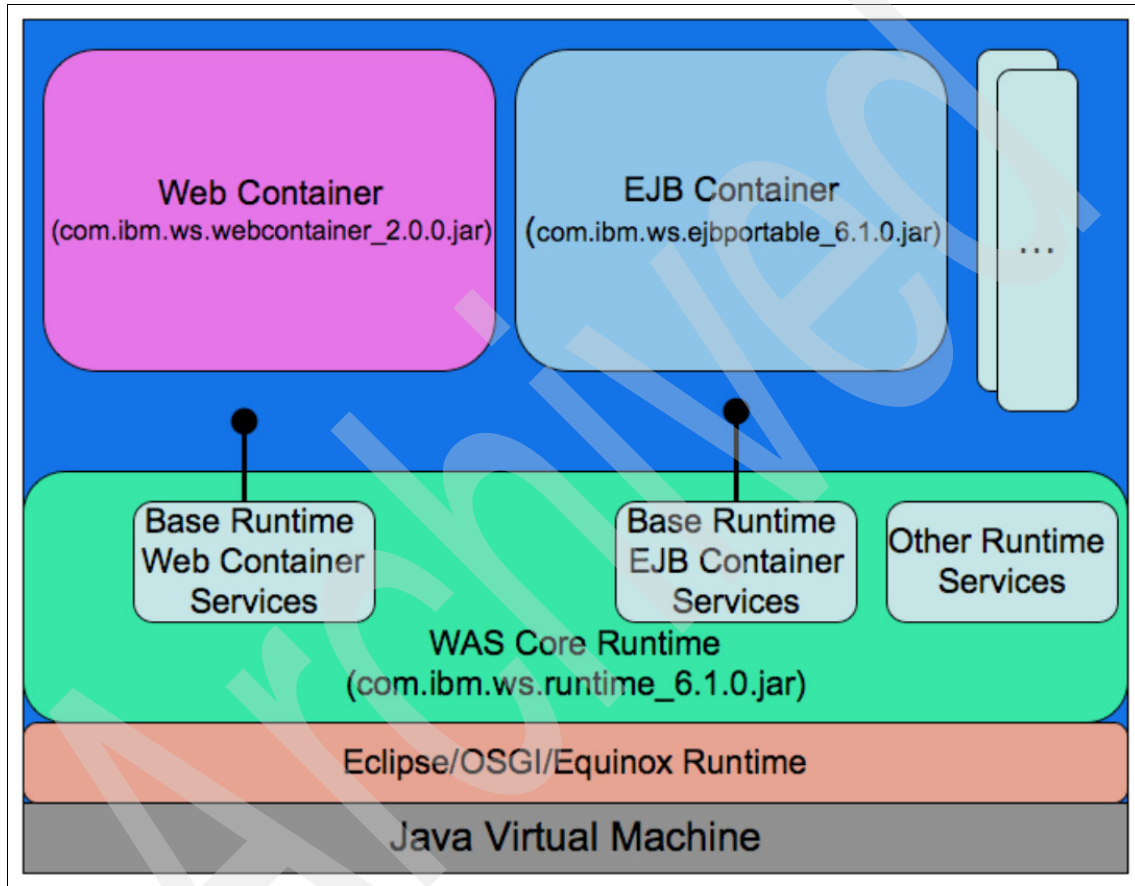


Figure 5-11 WebSphere Application Server deployment architecture in the JVM

Although this description gives you an understanding of deployment, it does not explain the base functionality that can be found in the core WebSphere Application Server runtime services and how to best tune the environment.

When the core Eclipse/OSGI runtime starts up, it examines the plugin.xml file to see what components to start as part of the base, and what to offer for extension. In this file is the startup order of the application environment that the containers

and the rest of the environment build upon. This core application server base contains code that knows what platform it is running on and sets up thread pools, communications channels, and other I/O and memory management that is appropriate for the platform it is running on, as well as generic services such as Java Management Extensions (JMX) management, Java Transaction Service (JTS) transaction management, and logging. The information in the `plugin.xml` file and the configuration in the `server.xml` file control the setup and usage of the environment and how this server instance behaves.

On AIX, the WebSphere Application Server runtime knows to make use of asynchronous I/O, so appropriate flags are set in the Java code. Java Native Interface (JNI) shared objects (`libibmaio.so` or `libibmaiodbg.so`) are loaded into the WebSphere Application Server address space to best use asynchronous I/O on the AIX platform. So, by tuning asynchronous I/O for the AIX platform, you effectively tune WebSphere Application Server.

The WebSphere Application Server runtime requires communications with the outside world over TCP/IP sockets, so it sets up a pool of sockets and code to manage listeners and socket clients called *channels*. These channels can be built into a chain, so an HTTP channel builds on a TCP channel, and so on.

Some channels have very specific coding that distinguishes them from other channels, such as the HTTP channels that support the Web container and Web services over TCP and SSL, the CORBA IIOP channel that forms the core of the EJB container communications via its central CORBA ORB, the Service Integration Bus (SIB) MQ messaging channel, and the internal Data Replication Service (DRS) channel used for fast communications between cluster members.

These channels are managed by the application server runtime itself and essentially sit outside of the containers. However, they pass requests, responses, packets, and messages into and out of the containers.

The IIOP communications are particularly interesting because a specialized JNI shared object (`libSelector.so`) is used for a reader thread for this task. Note the references in many parts of the WebSphere Application Server configuration to JFAP, either with reference to the Service Integration Bus or channels, which refers to the format and protocols used for messaging (that is, WebSphere MQ-type communications with embedded messaging).

There are both inbound and outbound chains of channels, which are managed independently to improve throughput and resilience. A request comes into the channel and passes through the chain before being handled by the container in a manner similar to a queue, with the container then routing the message as appropriate to the application module responsible for it for handling. Then, any response is put into the appropriate queue for the given outbound channel.

The management of these mappings uses the server.xml file for the WebSphere Application Server instance in tandem with the generic configuration in the com.ibm.ws.runtime_6.1.0.jar file plugin.xml file. Figure 5-12 illustrates WebSphere Application Server channels and thread pools.

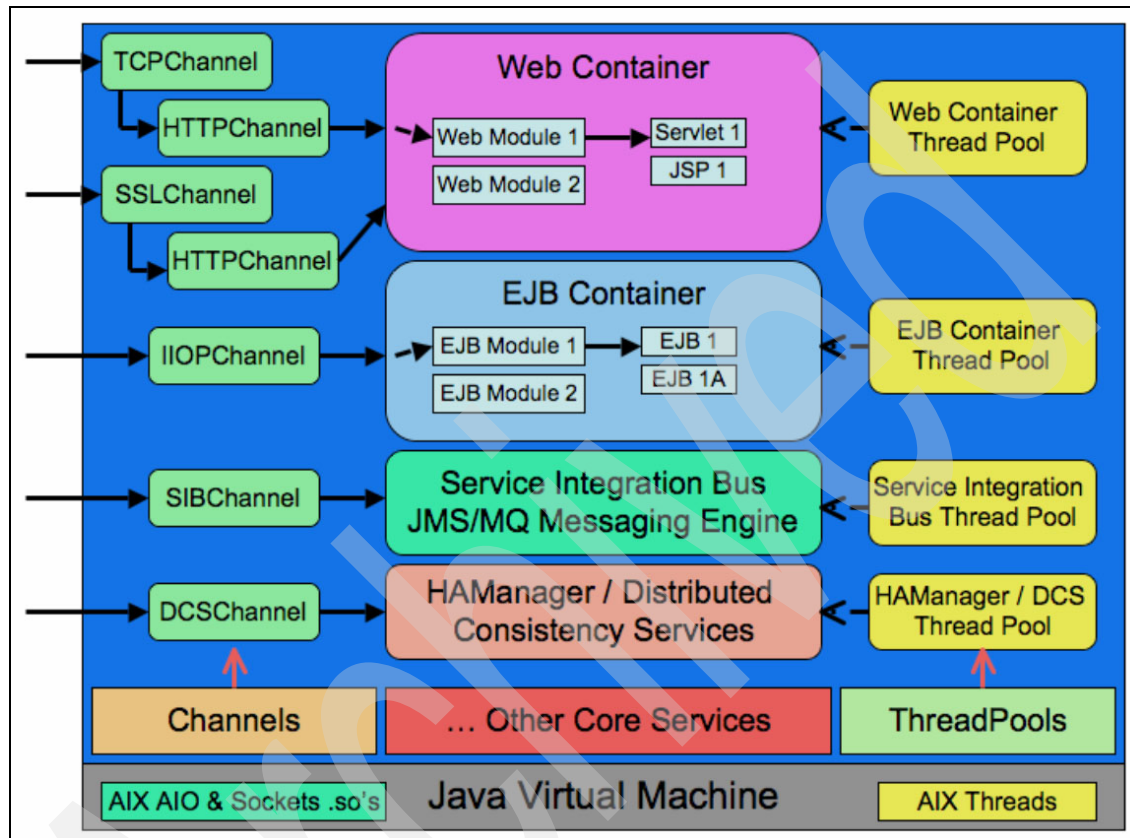


Figure 5-12 WebSphere Application Server channels and thread pools

Similarly, the application server sets up thread pools that are appropriate for the platform. In past WebSphere Application Server releases, this could be problematic given the 8:1 ratio between user mode application threads and kernel threads and synchronization difficulties this could cause with external processes such as those for WebSphere MQ Server. However, the use of the AIX System thread mode and 1:1 thread ratio has removed these difficulties.

The WebSphere Application Server process has Java code to manage the thread pools for the containers at one level, on which the containers and services build. But the core thread code is handled within the JVM itself and its AIX pthreads-based thread and monitor code, as discussed when examining the

JVM. Configuration of these thread pools is again held in the `server.xml` file for the WebSphere Application Server instance.

Other core services are implemented in the WebSphere Application Server runtime as Java code. However, the transaction management, logging, and RAS features of WebSphere Application Server exploit the platform, as described in the discussion of asynchronous I/O. This is unlike iSeries and zSeries platforms, which have specific code in the runtime.

The security mechanism for WebSphere Application Server is essentially built on industry standards for J2EE. For the Web container, the IETF RFC 2617 standard is used for authentication. The X509 standard is used for certificates. The SSL/TLS standards in RFC 2246 are used for transport security.

The EJB container standard part of J2EE is built around CORBA. For this reason the CORBA CSv2 security standard, which includes certificate and SSL standards and patterns for their specific usage, is used. These have specific implementations within the Java Virtual Machine and its libraries, but these are supplemented with additional support in WebSphere Application Server for functions specific to an application server environment, and which are all controlled using configuration in the `security.xml` file.

User information is usually mapped down to an Lightweight Directory Access Protocol (LDAP) directory. Java also includes the Java Authentication and Authorization Service (JAAS) for a custom implementation, and IBM ship code for the Java Cryptography Extensions (JCE) and the GSSAPI.

Keep in mind, though, that while WebSphere Application Server 6.1 runs on a virtual machine that has its own security mechanisms, it also sits on top of the platform security that also includes SSL, Kerberos, and LDAP, so the two layers of security can be used in combination.

It is common in an enterprise to use Kerberos and LDAP to secure AIX for the WebSphere Application Server, platform administrators, and WebSphere Application Server account, and then to have the WebSphere Application Server users who are accessing the applications running inside WebSphere Application Server also using Kerberos and LDAP, with both AIX and WebSphere Application Server sharing a common Kerberos keytab file.

As with all security-related matters, WebSphere Application Server should *not* run under the root account, and should not have its binaries writable. But in general, the combined security is very flexible and configurable for many different purposes. Configuration for security is in the `security.xml` file for the WebSphere Application Server instance, and this file needs careful control and configuration.

Underpinning WebSphere Application Server 6.1 architecture is something new for Version 6.1 and something that is important for a large number of IBM server-side infrastructure packages for the future. This is the Eclipse 3.1.2/OSGI (formerly the Open Standard Gateway Initiative, but now an obsolete name) runtime.

This is exactly the same runtime that underpins the well-known Eclipse 3.1.2 IDE, and it is known to Rational Application Developer users everywhere. So why is it used? Because the Eclipse runtime is a framework for bundling lightweight components that can provide extensions and dynamically extend other components (known as plug-ins) at runtime, all with minimum overhead.

The similar OSGI model provides bundles (similar to plug-ins) that provide services. The extensions, extension points, and services can all be enumerated via registries that use configuration files to get their metadata.

Thus, a plug-in or bundle is copied into a particular directory, provides a small set of boilerplate code (although this is not a hard requirement), and either a plugin.xml file or a MANIFEST.MF file that contains details of what extensions and services it provides and which ones it uses. The Java code itself is not loaded until it is required; only the plug-in configuration files are accessed and loaded by the registry to minimize memory usage.

An important consideration is that extensions and extension points carry version information with them to allow multiple versions of a component to co-reside without breaking any version dependencies for extensions.

5.4.2 Eclipse 3.1.2 and OSGI/Equinox Runtime

WebSphere Application Server uses the Eclipse 3.1.2 version of the Eclipse/OSGI environment for its runtime infrastructure to manage the loading of its main infrastructure components. Although this adds complexity, it allows IBM to provide extensions to the core WebSphere Application Server product (such as through the new feature packs) without affecting the core code. IBM, as well as vendors, have stated that many of their infrastructure packages (including some Tivoli products, the IBM Systems Console, and so on) will in the future be Eclipse/OSGI-based. Figure 5-13 on page 240 illustrates Eclipse and OSG/Equinox in the WebSphere Application Server runtime infrastructure.

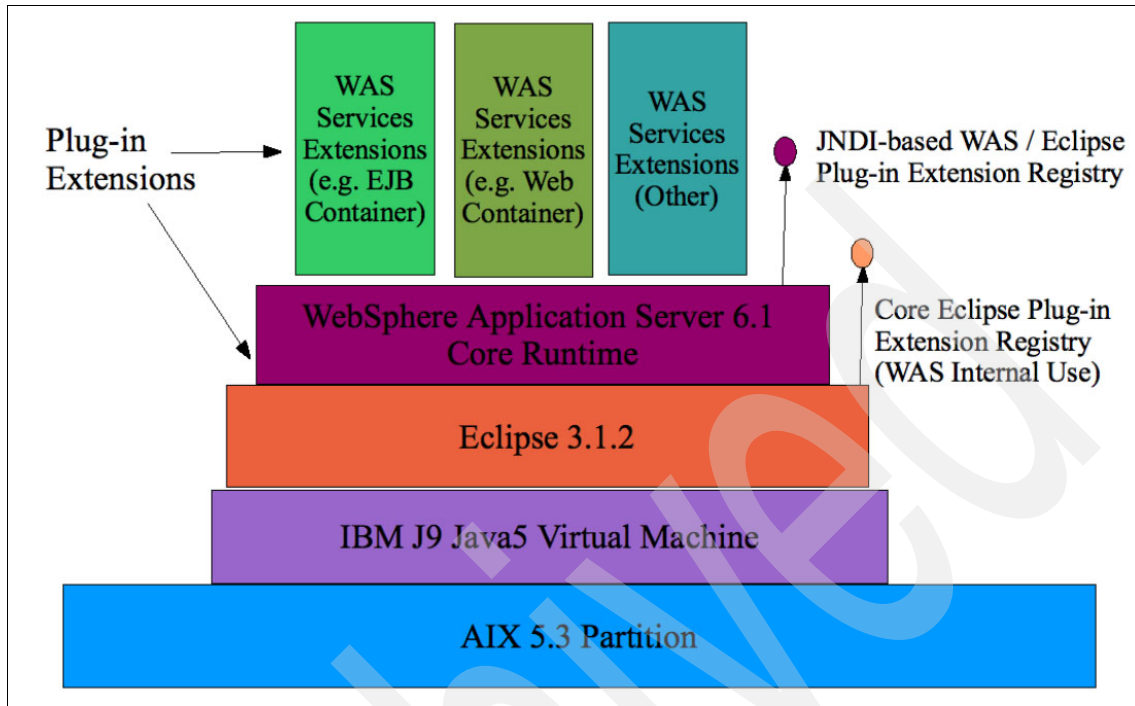


Figure 5-13 Eclipse and OSGI/Equinox in the WebSphere Application Server runtime infrastructure

To understand how this works, look at the “configuration” directory for the WebSphere Application Server installation and examine the config.ini file that lists what base components of Eclipse are required to get the internal WebSphere Application Server extension registry up and running from the Eclipse 3.1.2 runtime.

Then examine the startup scripts for WebSphere Application Server to see the Eclipse runtime JAR files being accessed. Finally, copy the com.ibm.ws.runtime_6.1.0.jar file from a WebSphere Application Server installation and open its plugin.xml file to see the component configurations used to create the underlying WebSphere Application Server product.

With the later Eclipse environments, the similar OSGI (formerly the Open Service Gateway Initiative) platform R4 services and bundles are also supported, using a framework implementation called Equinox. Eclipse allows plug-ins to provide runtime extensions based on what they find around them, and OSGI adds the dynamic-aware behavior to control runtime addition and removal of bundles. All of this functionality allows applications based on it to be extended and modified without a restart.

OSGI started out as a consortium working on embedded and mobile initiatives, and then defined the dynamic platform and how the applications would group Java code, native code, and other resources to form the deployable bundles. A manifest file (MANIFEST.MF) describes the bundle contents, what its dependencies are, what packages it exports, and if it is to be initialized before the first class it contains is accessed.

In earlier, non-Equinox Eclipse environments, the extensions and extension points are all covered in the plugin.xml file. The current WebSphere Application Server 6.1 runtime is based on the hybrid Eclipse 3.1.2 implementation, so both mechanisms are used by different parts of the environment.

In OSGI, bundles detect being started and stopped to perform initialization and shutdown in a *bundle activator*, and register services in a *service registry*. A *service* is simply a Java class that usually exposes a well-known interface that is made available for others to bind to and use.

The current OSGI Release 4 standard also defines other features of the platform for use on the server side, such as a lightweight HTTP listener. The current Eclipse implementation of OSGI is Equinox, which is used by WebSphere Application Server. However, there are other implementations of the standard such, as Apache Felix. For more details, visit:

<http://www.osgi.org>

The dynamic extensions and services mechanisms allow the WebSphere Application Server environment that runs on top of Eclipse to be dynamically extensible without changing the core WebSphere Application Server code. Code, including that from standard J2EE applications with the appropriate additional code to support the Eclipse registry and extension mechanism, uses the registry to find what it wants to extend and then adds its own code.

IBM is now providing feature packs to add new features to WebSphere Application Server 6.1 without changing the base code, and this mechanism is key to this type of extension. Platform-specific features are also being added, which will allow new AIX features to be taken advantage of. This is best seen with the startup of the WebSphere Application Server containers.

WebSphere Application Server effectively has an internal Eclipse registry, and one that is accessed using Java Naming and Directory Interface (JNDI). The internal one is core to the Eclipse platform itself, with some low level WebSphere Application Server Eclipse plug-in code.

According to the “Official Eclipse 3.0 FAQs”, the simplest Eclipse application (as opposed to an OSGI bundle or plug-in), consists of the following elements:

- ▶ A startup.jar file that contains a single class responsible for starting the Eclipse platform via a plugin, usually org.eclipse.core.runtime_XXX.jar. In the case of Eclipse 3.1.2, and thus WebSphere Application Server, the launcher class is org.eclipse.core.launcher. The main work is performed in a method called “basicRun”, which checks the security, paths, and configuration before performing the main Eclipse initialization by loading plug-ins named in the configuration config.ini file. In Eclipse 3.1.2, this performs a significant amount of OSGI-specific configuration checks.
- ▶ A configurator plug-in, which is org.eclipse.update.configurator by default.
- ▶ The org.eclipse.core.runtime plug-in, which is responsible for enumerating the plugin.xml files for the plug-ins in the /plugins directory and setting up the extension registry.
- ▶ Runtime plug-ins to support the platform, which are now provided by the org.eclipse.osgi (org.eclipse.osgi; org.eclipse.osgi.util; and org.eclipse.osgi.services) (now Equinox) plug-ins.

The startup for this base platform is controlled by a config.ini file that is found in the /configuration directory under the WebSphere Application Server installation directory structure. Here, you can examine the startup scripts, the documentation, the logs, and drill down into the JAR file contents. Remember, a JAR file is just a special type of zip file, and you can view the contents of zip files using Windows Explorer or other similar browsing tools. You can refer to the Eclipse startup and registry documentation for more information about this subject:

<http://www.eclipse.org>

Understanding this task can help in diagnosing startup problems. You can refer to WebSphere Application Server documentation to some extent, with much of the rest available as Open Source code. WebSphere Application Server uses the same Eclipse 3.1.2 runtime for AIX/Motif, which can be downloaded in binary or source form:

<http://www.eclipse.org>

In order to allow application vendors to provide their own extensions and plug-ins, a public version of the Eclipse registry is accessible via JNDI, which exposes the components available for public use and extension.

5.4.3 WebSphere Application Server-specific JNI native shared object libraries

WebSphere Application Server-specific libraries can be found under the WebSphere Application Server installation directory and the /bin subdirectory, where the startServer.sh script can be found that starts instances.

The shared objects in this directory reveal how WebSphere Application Server accesses operating system functionality, thus bypassing the JVM. To find out how these are used as JNI objects you can use **strings -a** to look for the Java_XXXXX entries and determine how each maps to the Java world.

Example 5-42 WebSphere Application Server-specific shared objects listing

```
# ls -l *.so
-rwxr-xr-x 1 root system 6556 06 Oct 2006 libgetClasses.so
-rwxr-xr-x 1 root system 48530 06 Oct 2006 libibmaiodbg.so
-rwxr-xr-x 1 root system 28317 06 Oct 2006 libibmaio.so
-rwxr-xr-x 1 root system 60175 06 Oct 2006 libNativeFile.so
-rwxr-xr-x 1 root system 24472 06 Oct 2006 libpmiJvmpiProfiler.so
-rwxr-xr-x 1 root system 30381 06 Oct 2006 libpmiJvmtiProfiler.so
-rwxr-xr-x 1 root system 11860 06 Oct 2006 libSelector.so
-rwxr-xr-x 1 root system 16250 06 Oct 2006 libSystemData.so
-rwxr-xr-x 1 root system 21720 06 Oct 2006 libUnixRegistryImpl.so
-rwxr-xr-x 1 root system 68252 06 Oct 2006 libWs60ProcessManagement.so
```

Table 5-5 lists the WebSphere Application Server-specific shared objects and functional descriptions.

Table 5-5 WebSphere Application Server-specific shared objects

Name	Java class	Functional description
libgetClasses.so	com.ibm.ws.classloader.ClassLoaderDump	Java Classloader Debug Access
libibmaiodbg.so	com.ibm.io.async.AsyncLibrary	Asynchronous I/O mapping - Debug Version
libibmaio.so	com.ibm.io.async.AsyncLibrary	Asynchronous I/O mapping - Normal Version
libNativeFile.so	com.ibm.io.file.UnixNativeFile	Access to native UNIX file system structures - including /proc file system
libpmiJvmpiProfiler.so	com.ibm.ws.pmi.server.jvmpi.JvmpiJni	Java JVMPi performance statistics-gathering management

Name	Java class	Functional description
libpmiJvmtiProfiler.so	com.ibm.ws.pmi.server.jvmpi.JvmpiJni	Java JVMPI/JMTI performance statistics-gathering management. Maps JVMTI data to JVMPI.
libSelector.so	com.ibm.ws.orbimpl.transport.JNIReaderThread	Object request broker connection management. Important because most J2EE security uses the CORBA CSrv2 functionality and much J2EE internal communication uses CORBA IIOP, so this shows that WebSphere Application Server bypasses JVM sockets functionality for some of the CORBA (and thus, EJB container security) functions and talks directly to AIX networking code.
libSystemData.so	com.ibm.ws.pmi.server.system.SystemData	General system level performance statistics gathering, that is, CPU and memory stats
libUnixRegistryImpl.so	com.ibm.ws.security.registry.unix.UnixRegistryImpl	Password file access and UNIX authentication
libWs60ProcessManagement.so	com.ibm.ws.process.EnvUtilGlue_native & com.ibm.ws.process.UnixProcessGlue	Access the UNIX environment and process management (that is, start and stop).

5.4.4 WebSphere Application Server - startup and operation

The following sections describe WebSphere Application Server system operations.

Runtime startup

It is important, for problem determination, to understand how WebSphere Application Server starts up. This can help in understanding the configuration and runtime issues highlighted in the startServer.log file that traces startup of WebSphere Application Server.

Consider the following examples showing stack traces from failed startups. In Example 5-43, the XML cell configuration files had invalid permissions.

Example 5-43 Stack trace: XML cell configuration file with invalid permissions

```
[14/06/07 21:52:56:874 GMT+00:00] 0000000a FileDocument E ADMR0104E: The system is unable to
read document cells/spock44pNode01Cell/cell.xml: java.io.IOException: The file access
permissions do not allow the specified action.
```



```

at java.io.UnixFileSystem.createFileExclusively(Native Method)
at java.io.File.checkAndCreate(File.java:1380)
at java.io.File.createTempFile(File.java:1469)
at com.ibm.ws.management.repository.FileDocument.createTempFile(FileDocument.java:562)
at com.ibm.ws.management.repository.FileDocument.read(FileDocument.java:498)
at
com.ibm.ws.management.repository.FileRepository.extractInternal(FileRepository.java:986)
at com.ibm.ws.management.repository.FileRepository.extract(FileRepository.java:958)
at com.ibm.ws.management.repository.FileRepository.extract(FileRepository.java:916)
at com.ibm.ws.management.repository.FileRepository.extract(FileRepository.java:906)
at com.ibm.ws.taskmanagement.task.TaskContainer.<init>(TaskContainer.java:200)
at
com.ibm.ws.taskmanagement.task.TaskManagementMBean.initialize(TaskManagementMBean.java:74)
at
com.ibm.ws.runtime.component.ContainerImpl.initializeComponent(ContainerImpl.java:1338)
at
com.ibm.ws.runtime.component.ContainerImpl.initializeComponents(ContainerImpl.java:1136)
at com.ibm.ws.runtime.component.ServerImpl.initialize(ServerImpl.java:347)
at com.ibm.ws.runtime.WsServerImpl.bootServerContainer(WsServerImpl.java:177)
at com.ibm.ws.runtime.WsServerImpl.start(WsServerImpl.java:139)
at com.ibm.ws.runtime.WsServerImpl.main(WsServerImpl.java:460)
at com.ibm.ws.runtime.WsServer.main(WsServer.java:59)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:64)
at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:615)
at com.ibm.wsspi.bootstrap.WSLauncher.launchMain(WSLauncher.java:183)
at com.ibm.wsspi.bootstrap.WSLauncher.main(WSLauncher.java:90)
at com.ibm.wsspi.bootstrap.WSLauncher.run(WSLauncher.java:72)
at
org.eclipse.core.internal.runtime.PlatformActivator$1.run(PlatformActivator.java:226)
at org.eclipse.core.runtime.adaptor.EclipseStarter.run(EclipseStarter.java:376)
at org.eclipse.core.runtime.adaptor.EclipseStarter.run(EclipseStarter.java:163)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:64)
at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:615)
at org.eclipse.core.launcher.Main.invokeFramework(Main.java:334)
at org.eclipse.core.launcher.Main.basicRun(Main.java:278)
at org.eclipse.core.launcher.Main.run(Main.java:973)
at com.ibm.wsspi.bootstrap.WSPreLauncher.launchEclipse(WSPreLauncher.java:245)
at com.ibm.wsspi.bootstrap.WSPreLauncher.main(WSPreLauncher.java:73)

```

In Example 5-44 on page 246, the WebSphere Application Server Kerberos and LDAP configuration attempted to connect to an Active Directory Server that was inactive.

Example 5-44 Stack trace: Active Directory server failed connection

```
14/06/07 21:54:31:986 GMT+00:00] 0000000a exception      E
com.ibm.ws.wim.adapter.ldap.LdapConnection getDirContext
                                com.ibm.websphere.wim.exception.WIMSystemException: CWWIM4520E
The 'javax.naming.CommunicationException: 192.168.200.3:389 [Root exception is
java.net.ConnectException: A remote host refused an a
ttempted connect operation.].' naming exception occurred during processing.
    at com.ibm.ws.wim.adapter.ldap.LdapConnection.getDirContext(LdapConnection.java:1348)
    at com.ibm.ws.wim.adapter.ldap.LdapConnection.search(LdapConnection.java:2451)
    at
com.ibm.ws.wim.adapter.ldap.LdapConnection.checkSearchCache(LdapConnection.java:2374)
    at com.ibm.ws.wim.adapter.ldap.LdapConnection.search(LdapConnection.java:2550)
    at com.ibm.ws.wim.adapter.ldap.LdapConnection.searchEntities(LdapConnection.java:2694)
    at com.ibm.ws.wim.adapter.ldap.LdapAdapter.search(LdapAdapter.java:2721)
    at com.ibm.ws.wim.ProfileManager.searchRepository(ProfileManager.java:4073)
    at com.ibm.ws.wim.ProfileManager.searchImpl(ProfileManager.java:891)
    at com.ibm.ws.wim.ProfileManager.genericProfileManagerMethod(ProfileManager.java:245)
    at com.ibm.ws.wim.ProfileManager.search(ProfileManager.java:333)
    at com.ibm.websphere.wim.ServiceProvider.search(ServiceProvider.java:447)
    at com.ibm.ws.wim.registry.util.UniqueIdBridge.getUniqueId(UniqueIdBridge.java:151)
    at com.ibm.ws.wim.registry.WIMUserRegistry$6.run(WIMUserRegistry.java:349)
    at com.ibm.ws.security.auth.ContextManagerImpl.runAs(ContextManagerImpl.java:3731)
    at
com.ibm.ws.security.auth.ContextManagerImpl.runAsSystem(ContextManagerImpl.java:3813)
    at
com.ibm.ws.wim.security.authz.jacc.JACCSecurityManager.runAsSuperUser(JACCSecurityManager.java:
484)
    at
com.ibm.ws.wim.security.authz.ProfileSecurityManager.runAsSuperUser(ProfileSecurityManager.java
:961)
    at com.ibm.ws.wim.registry.WIMUserRegistry.getUniqueId(WIMUserRegistry.java:338)
    at
com.ibm.ws.security.registry.UserRegistryImpl.getUniqueId(UserRegistryImpl.java:446)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:64)
    at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:615)
    at com.ibm.rmi.util.ProxyUtil$2.run(ProxyUtil.java:654)
    at java.security.AccessController.doPrivileged(AccessController.java:241)
    at com.ibm.rmi.util.ProxyUtil.invokeWithPrivilege(ProxyUtil.java:650)
    at com.ibm.CORBA.iiop.ClientDelegate.invoke(ClientDelegate.java:1118)
    at $Proxy1.getUniqueId(Unknown Source)
    at
com.ibm.websphere.security._UserRegistry_Stub.getUniqueId(_UserRegistry_Stub.java:380)
    at
com.ibm.ws.security.role.RoleBasedConfiguratorImpl.fillMissingAccessIds(RoleBasedConfiguratorIm
pl.java:706)
```

```

        at
com.ibm.ws.security.role.RoleBasedConfiguratorImpl.loadApplication(RoleBasedConfiguratorImpl.java:270)
        at
com.ibm.ws.security.core.distSecurityComponentImpl.configureRoleBasedAuthz(distSecurityComponentImpl.java:1224)
        at
com.ibm.ws.security.core.distSecurityComponentImpl.initialize(distSecurityComponentImpl.java:377)
        at
com.ibm.ws.security.core.distSecurityComponentImpl.startSecurity(distSecurityComponentImpl.java:304)
        at
com.ibm.ws.security.core.SecurityComponentImpl.startSecurity(SecurityComponentImpl.java:101)
        at
com.ibm.ws.security.core.ServerSecurityComponentImpl.start(ServerSecurityComponentImpl.java:281)
    )
        at com.ibm.ws.runtime.component.ContainerImpl.startComponents(ContainerImpl.java:977)
        at com.ibm.ws.runtime.component.ContainerImpl.start(ContainerImpl.java:673)
        at
com.ibm.ws.runtime.component.ApplicationServerImpl.start(ApplicationServerImpl.java:191)
        at com.ibm.ws.runtime.component.ContainerImpl.startComponents(ContainerImpl.java:977)
        at com.ibm.ws.runtime.component.ContainerImpl.start(ContainerImpl.java:673)
        at com.ibm.ws.runtime.component.ServerImpl.start(ServerImpl.java:485)
        at com.ibm.ws.runtime.WsServerImpl.bootServerContainer(WsServerImpl.java:191)
        at com.ibm.ws.runtime.WsServerImpl.start(WsServerImpl.java:139)
        at com.ibm.ws.runtime.WsServerImpl.main(WsServerImpl.java:460)
        at com.ibm.ws.runtime.WsServer.main(WsServer.java:59)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:64)
        at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:615)
        at com.ibm.wsspi.bootstrap.WSLauncher.launchMain(WSLauncher.java:183)
        at com.ibm.wsspi.bootstrap.WSLauncher.main(WSLauncher.java:90)
        at com.ibm.wsspi.bootstrap.WSLauncher.run(WSLauncher.java:72)
        at
org.eclipse.core.internal.runtime.PlatformActivator$1.run(PlatformActivator.java:226)
        at org.eclipse.core.runtime.adaptor.EclipseStarter.run(EclipseStarter.java:376)
        at org.eclipse.core.runtime.adaptor.EclipseStarter.run(EclipseStarter.java:163)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:64)
        at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:615)
        at org.eclipse.core.launcher.Main.invokeFramework(Main.java:334)
        at org.eclipse.core.launcher.Main.basicRun(Main.java:278)
        at org.eclipse.core.launcher.Main.run(Main.java:973)
        at com.ibm.wsspi.bootstrap.WSPreLauncher.launchEclipse(WSPreLauncher.java:245)

```

```
at com.ibm.wsspi.bootstrap.WSPreLauncher.main(WSPreLauncher.java:73)
```

For both Example 5-43 on page 244 and Example 5-44 on page 246, the bottom of the stack is the same.

It highlights the underlying Eclipse/OSGI runtime startup that launches the WebSphere Application Server runtime; the bottom of the stack is shown in Example 5-45.

Example 5-45 Stack trace: Eclipse/OSGI runtime startup

```
at com.ibm.wsspi.bootstrap.WSLauncher.launchMain(WSLauncher.java:183)
    at com.ibm.wsspi.bootstrap.WSLauncher.main(WSLauncher.java:90)
    at com.ibm.wsspi.bootstrap.WSLauncher.run(WSLauncher.java:72)
    at
org.eclipse.core.internal.runtime.PlatformActivator$1.run(PlatformActivator.java:226)
    at org.eclipse.core.runtime.adaptor.EclipseStarter.run(EclipseStarter.java:376)
    at org.eclipse.core.runtime.adaptor.EclipseStarter.run(EclipseStarter.java:163)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:64)
    at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:615)
    at org.eclipse.core.launcher.Main.invokeFramework(Main.java:334)
    at org.eclipse.core.launcher.Main.basicRun(Main.java:278)
    at org.eclipse.core.launcher.Main.run(Main.java:973)
    at com.ibm.wsspi.bootstrap.WSPreLauncher.launchEclipse(WSPreLauncher.java:245)
    at com.ibm.wsspi.bootstrap.WSPreLauncher.main(WSPreLauncher.java:73)
```

This stack traces start with the `com.ibm.wsspi.bootstrap.WSPreLauncher` class and its main method. All Java processes are started by a class with a main method as its entry point.

This main method calls the `launchEclipse` method to initiate the Eclipse/OSGI runtime startup. The `org.eclipse.core.*` classes are used to launch the Eclipse/OSGI runtime.

Finally, the `com.ibm.wsspi.bootstrap.WSLauncher` class has a `launchMain` method that starts the WebSphere Application Server runtime. The WebSphere Application Server runtime comes into play from the `WsServer` classes upwards until the containers themselves have started, as illustrated in Example 5-46.

Example 5-46 Stack trace snippet: WebSphere Application Server runtime startup

```
at
com.ibm.ws.security.core.ServerSecurityComponentImpl.start(ServerSecurityComponentImpl.java:281)
    at com.ibm.ws.runtime.component.ContainerImpl.startComponents(ContainerImpl.java:977)
```

```

at com.ibm.ws.runtime.component.ContainerImpl.start(ContainerImpl.java:673)
at
com.ibm.ws.runtime.component.ApplicationServerImpl.start(ApplicationServerImpl.java:191)
at com.ibm.ws.runtime.component.ContainerImpl.startComponents(ContainerImpl.java:977)
at com.ibm.ws.runtime.component.ContainerImpl.start(ContainerImpl.java:673)
at com.ibm.ws.runtime.component.ServerImpl.start(ServerImpl.java:485)
at com.ibm.ws.runtime.WsServerImpl.bootServerContainer(WsServerImpl.java:191)
at com.ibm.ws.runtime.WsServerImpl.start(WsServerImpl.java:139)
at com.ibm.ws.runtime.WsServerImpl.main(WsServerImpl.java:460)
at com.ibm.ws.runtime.WsServer.main(WsServer.java:59)

```

Although this seems relatively straightforward, recognize that these examples sample the stack from a single thread. A real implementation is more complex. The extra complexity comes from the use of the Eclipse/OSGI runtime, but brings with it benefits in extensibility and memory management improvements.

To understand how WebSphere Application Server starts up, examine the `startServer.sh` file from the `/usr/IBM/WebSphere/AppServer/bin` directory. Although there are similarly named files for the profiles that start the instances, this file does most of the work as the profile versions reference it.

The setup is complex, particularly the setup of the command line, which is accomplished by a combination of this shell script and the `setupCmdLine.sh` script that sets up the classpath and other directories, including the `lib` directory `startup.jar` and `bootstrap.jar` files on the classpath.

This `startup.jar` file is the same `startup.jar` file that comes with the Eclipse 3.1.2 runtime for AIX/Motif and, as mentioned previously, starts the Eclipse platform runtime. Here is the actual command line that starts up WebSphere Application Server:

```

"$JAVA_HOME"/bin/java \"$OSGI_INSTALL\" \"$OSGI_CFG\" \"$X_ARGS\"
\\$WebSphere Application Server _DEBUG \\$CONSOLE_ENCODING \\$D_ARGS
\\-classpath \"$WebSphere Application Server _CLASSPATH\"
\\$USER_INSTALL_PROP \\$JVM_EXTRA_CMD_ARGS
\\com.ibm.ws.bootstrap.WSLauncher \\$SHELL \"$CONFIG_ROOT\" \"$WebSphere
Application Server _CELL\" \"$WebSphere Application Server _NODE\" \"$@"
$WORKSPACE_ROOT_PROP

```

By examining at the `ps` command output, you can find the expanded version of this command line, with all variables substituted, as shown in Example 5-47.

Example 5-47 Sample full WebSphere Application Server startup command syntax

```

/usr/IBM/WebSphere/AppServer/java/bin/java -Declipse.security -Dwas.status.socket=34929
-Dosgi.install.area=/usr/IB

```

```

M/WebSphere/AppServer
-Dosgi.configuration.area=/usr/IBM/WebSphere/AppServer/profiles/testwas/configuration
-Djava.awt.headless=true -Dosgi.framework.extensions=com
.ibm.cds -Xshareclasses:name=webspherev61_%g,groupAccess,nonFatal -Xscmx50M
-Xbootclasspath/p:/usr/IBM/WebSphere/AppServer/java/jre/lib/ext/ibmorb.jar:/usr/IBM/WebS
phere/AppServer/java/jre/lib/ext/ibmext.jar -classpath
/usr/IBM/WebSphere/AppServer/profiles/testwas/properties:/usr/IBM/WebSphere/AppServer/propertie
s:/usr/IBM/Web
Sphere/AppServer/lib/startup.jar:/usr/IBM/WebSphere/AppServer/lib/bootstrap.jar:/usr/IBM/WebSph
ere/AppServer/lib/j2ee.jar:/usr/IBM/WebSphere/AppServer/lib/lmproxy.j
ar:/usr/IBM/WebSphere/AppServer/lib/urlprotocols.jar:/usr/IBM/WebSphere/AppServer/deploytool/it
p/batchboot.jar:/usr/IBM/WebSphere/AppServer/deploytool/itp/batch2.ja
r:/usr/IBM/WebSphere/AppServer/java/lib/tools.jar -Dibm.websphere.internalClassAccessMode=allow
-Xms50m -Xmx256m -Dws.ext.dirs=/usr/IBM/WebSphere/AppServer/java/lib
:/usr/IBM/WebSphere/AppServer/profiles/testwas/classes:/usr/IBM/WebSphere/AppServer/classes:/us
r/IBM/WebSphere/AppServer/lib:/usr/IBM/WebSphere/AppServer/installedC
hannels:/usr/IBM/WebSphere/AppServer/lib/ext:/usr/IBM/WebSphere/AppServer/web/help:/usr/IBM/Web
Sphere/AppServer/deploytool/itp/plugins/com.ibm.etools.ejbdeploy/runt
ime -Dderby.system.home=/usr/IBM/WebSphere/AppServer/derby
-Dcom.ibm.itp.location=/usr/IBM/WebSphere/AppServer/bin
-Djava.util.logging.configureByServer=true -Duser
.install.root=/usr/IBM/WebSphere/AppServer/profiles/testwas
-Djavax.management.builder.initial=com.ibm.ws.management.PlatformMBeanServerBuilder
-Dwas.install.root=/
usr/IBM/WebSphere/AppServer
-Dpython.cachedir=/usr/IBM/WebSphere/AppServer/profiles/testwas/temp/cachedir
-Djava.util.logging.manager=com.ibm.ws.bootstrap.WsLogMana
ger -Dserver.root=/usr/IBM/WebSphere/AppServer/profiles/testwas
-Dcom.ibm.ws.security.spnego.isEnabled=true
-Djava.security.auth.login.config=/usr/IBM/WebSphere/App
Server/profiles/testwas/properties/wsjaas.conf
-Djava.security.policy=/usr/IBM/WebSphere/AppServer/profiles/testwas/properties/server.policy
com.ibm.wsspi.bootstrap
.WSPreLauncher -nosplash -application com.ibm.ws.bootstrap.WSlauncher
com.ibm.ws.runtime.WsServer /usr/IBM/WebSphere/AppServer/profiles/testwas/config spock44pNode0
1Cell spock44pNode02 server1

```

In Example 5-47 on page 249, note the

-Dosgi.framework.extensions=com.ibm.cds system property entry that tells the OSGI framework to enable the com.ibm.cds classloader to facilitate AIX shared classes and thus, improve memory utilization and performance. This com.ibm.cds jar file is loaded by the startup.jar file during its initialization.

Also notice that the com.ibm.wsspi.bootstrap.WSPreLauncher entry at the start of the bold section matches the bottom of our stack dump, as shown in Example 5-45 on page 248.

Many of the command line setup defaults for the AIX platform can be found in the `aix.systemlaunch.properties` file inside the `com.ibm.ws.runtime_6.1.0.jar` JAR file, as shown in Example 5-48.

Example 5-48 Command line setup defaults from `aix.systemlaunch.properties` file

```
# Default system properties

# Default environment settings
com.ibm.websphere.management.launcher.defaultEnvironment=EXTSHM=ON

# Default JVM options
com.ibm.websphere.management.launcher.defaultJvmOptions=-Declipse.security -Dosgi.install.area=${WebSphere Application Server _INSTALL_ROOT}
-Dosgi.configuration.area=${USER_INSTALL_ROOT}/configuration
-Djava.awt.headless=true -Dosgi.framework.extensions=com.ibm.cds
-Xshareclasses:name=webspherev61_%g,groupAccess,nonFatal -Xscmx50M
```

Key elements in the command line are the last few parts that identify the actual Java class that runs the Eclipse/OSGI runtime (`com.ibm.wsspi.bootstrap.WSPreLauncher` from `lib/bootstrap.jar`); the Eclipse application plug-in that it starts (`com.ibm.ws.bootstrap.WSLauncher` from `lib/bootstrap.jar`); and the core runtime that is then loaded to provide the base environment for the rest of WebSphere Application Server (`com.ibm.ws.runtime.WsServer` from `plugins/com.ibm.ws.runtime_6.1.0.jar`). The configuration to apply to this for the cell, node, and instance are then applied.

The `WSPreLauncher` class and its `Main` method is what runs as the main Java command. It sets up all of the environment variables and properties it needs to be able to run the Eclipse/OSGI runtime environment.

It then runs the Eclipse/OSGI startup, `org.eclipse.core.launcher.Main` from `startup.jar`, as can be derived from the command line shown. This standard Eclipse Java code sets up the security, the remainder of the configuration and environment, and then reads the base Eclipse/OSGI configuration from the `configuration/config.ini` file.

The `config.ini` file from the standard Eclipse 3.1.2 environment says to load and start `org.eclipse.core.runtime` and `org.eclipse.update.configurator` from the `plugins` directory as standard, and the WebSphere Application Server configuration is no different. The Eclipse/OSGI runtime run from `startup.jar` loads further plug-ins from the `/plugins` directory controlled by the `configuration/config.ini` file, and further configuration in the `plugin.xml` files.

One of the command line parameters that the `WSPreLauncher` code passes to the `startup.jar` `Main` class is the `-application` parameter, which identifies the

Eclipse Rich Client Platform application to run, as for any other Eclipse RCP application (in this case, `com.ibm.ws.bootstrap.WSLauncher`).

The `WSLauncher` class sets up the configuration of the plug-ins, identifies the WebSphere Application Server runtime environment (in this case, as AIX), checks licenses (using the “TimeBomb” class), and sets up classloaders for the Java environment. It also ensures the profile name passed on the command line after it is available.

This class implements the normal `Runnable` interface so it can run on a separate thread, but it is not a true Eclipse/OSGI executable. The `WSLauncher` class simply loads the main runtime environment in the plug-in that it finds in the path that is named on the command line.

Now if you run the `WSLauncher` class itself as a plain Java application, you would receive the message: `usage: WSLauncher <className>`. So, you can see that the real work is done by the `com.ibm.ws.runtime.WsServer` class passed as a parameter to it.

To understand what happens next, open up the `com.ibm.ws.runtime_6.1.0.jar` JAR file that contains the runtime for WebSphere Application Server and look at the directory structure within it. (Remember, it is essentially a zip file.)

The `WsServer` class is a singleton that passes the real work on to `WsServerImpl` class, using the strategy pattern to act as its interface. This is a real Eclipse plug-in class that sets up the rest of the environment, that is composed of Eclipse plug-ins with their extensions and extension points, with significant access to the Eclipse platform extension registry.

Interestingly for this class is the amount of code dedicated to making the z/OS® implementation appear to function the same as other versions and to provide z/OS extensions, all referred to using `ws390` and z/OS methods and classes. In this class, the components and containers that form the core of a J2EE environment are started. Then, a number of other classes are used to perform tasks, such as create thread pools, set up JMX management, and so on, from the components that are started. The configuration in the `server.xml` file provides the specific setup of the application server instance and the components it loads using the `plugin.xml` to identify the components that can be loaded in the container and the order in which they load.

However, the threads of execution do not reveal much about what has been happening in the background to create the runtime environment for WebSphere Application Server. Remember, the Eclipse/OSGI runtime is now running, as started in `WSPreLauncher`, and it has read its configuration file. It has been reading the `plugin.xml` and `MANIFEST.MF` files of the plug-in jar files in the `plugins` directory to determine what needs to be loaded. This is all happening in

the background on a thread within Eclipse, which is why the WSLauncher environment identified by the **-application** command line argument has its own thread.

The key to knowing where code executes next lies in understanding the Eclipse/OSGI environment. Every OSGI bundle that is a runnable application has an Activator class that supports the start and stop methods to hook the startup and shutdown of the “application” (the `com.ibm.ws.runtime.WsServer` environment, in this case).

Looking at the MANIFEST.MF file that is part of the `com.ibm.ws.runtime_6.1.0.jar` file, notice that the startup code is in a class called `com.ibm.ws.runtime.component.RuntimeBundleActivator`; see Example 5-49.

Example 5-49 MANIFEST.MF: RuntimeBundleActivator

```
...
Bundle-Activator: com.ibm.ws.runtime.component.RuntimeBundleActivator
Bundle-Localization: plugin
Bundle-ManifestVersion: 2
Bundle-Name: WS_Server
Bundle-SymbolicName: com.ibm.ws.runtime; singleton:=true
Bundle-Vendor: IBM
Bundle-Version: 6.1.0
Eclipse-AutoStart: false
J2EE-DeploymentFactory-Implementation-Class:
com.ibm.ws.management.application.j2ee.deploy.spi.factories.DeploymentF
actoryImpl
Manifest-Version: 1.0
Require-Bundle: com.ibm.ws.bootstrap; visibility:=reexport,
org.eclipse.jdt.core
...
```

Also note that the application server runtime is a singleton; that is, only one copy can be loaded, as would be expected for a single application server instance. And the `com.ibm.ws.bootstrap` bundle code is re-exported for use by components elsewhere. Finally, notice that the Java Developer tools (`org.eclipse.jdt.core`) are actually used within WebSphere Application Server itself.

IBM uses the strategy pattern to implement most of WebSphere Application Server so the interfaces exposed to the outside world usually are wrappers for other classes and interfaces. In accordance with this, most of the implementation for the `RuntimeBundleActivator` above can be found in `com.ibm.wsspi.bootstrap.osgi.WsBundleActivator` in

com.ibm.ws.bootstrap_6.1.0.jar, as can be seen in its MANIFEST.MF file; see Example 5-50.

Example 5-50 Implementation of the RuntimeBundleActivator

```
Bundle-Activator: com.ibm.wsspi.bootstrap.osgi.WsBundleActivator
Bundle-Localization: plugin
Bundle-ManifestVersion: 2
Bundle-Name: WebSpherer Bootstrap Plug-in
Bundle-SymbolicName: com.ibm.ws.bootstrap; singleton:=true
Bundle-Vendor: IBM
Bundle-Version: 6.1.0
Eclipse-AutoStart: true
Manifest-Version: 1.0
Import-Package: org.eclipse.core.runtime,
    org.osgi.framework
Export-Package: com.ibm.ws.bootstrap,
    com.ibm.ws.runtime.service.impl,
    com.ibm.wsspi.bootstrap,
    com.ibm.wsspi.bootstrap.osgi,
    com.ibm.wsspi.extension,
    com.ibm.wsspi.runtime.service
```

Remember that the com.ibm.ws.bootstrap_6.1.0.jar file that this MANIFEST.MF file comes from was mentioned in the Require-Bundle entry in the MANIFEST.MF file of the com.ibm.ws.runtime_6.1.0.jar file. This signifies that it is a dependency, so you can be sure that the underlying code for the implementation is already loaded and available.

This simple OSGi activator code implementation simply registers and unregisters the bundle context with the WebSphere services registry. So what actually runs? The Eclipse applications registered in the plugin.xml files run.

When the Eclipse runtime loads, it enumerates all of the jar files and directories under the /plugins directory and examines the plugin.xml or MANIFEST.MF file to set up extension points or extensions that extend other extension points.

In the OSGi world, these plug-ins are bundles that also offer services. The extensions and extension points are handled by the platform extension registry. The services are handled by the OSGi service registry.

During startup of the WebSphere Application Server runtime both are used, but it is not until you get into the com.ibm.ws.runtime_6.1.0.jar code that the service registry gets more heavily used.

Although the plug-ins are being evaluated, those with the Eclipse-AutoStart:true setting (such as those for com.ibm.wsspi.bootstrap.osgi.WsBundleActivator in com.ibm.ws.bootstrap_6.1.0.jar) are actually loaded and their startup code executed. Those without that setting simply have their XML descriptions loaded for later use when referenced.

If you look inside the plugin.xml file for com.ibm.ws.bootstrap_6.1.0.jar, you find that another WSLauncher class is referenced, as shown in Example 5-51, that sets up the Eclipse registry environment for WebSphere Application Server. This was seen in our stack trace earlier, after the EclipseStarter and PlatformActivator classes were brought into play.

Example 5-51 WSLauncher class is referenced to set up registry environment

```
...
<extension
    id="WSLauncher"
    point="org.eclipse.core.runtime.applications">
    <application>
        <run class="com.ibm.wsspi.bootstrap.WSLauncher" />
    </application>
</extension>
...
```

This extension point tells you that the class com.ibm.wsspi.bootstrap.WSLauncher is designed to be launched as an Eclipse application by the runtime, and it extends the Eclipse IPlatformRunnable interface to allow Eclipse to execute its run method.

Inside this WSLauncher class (which is similar in structure to the earlier WSLauncher class), the run method eventually calls a launchMain method which finds classes that extend the extension point com.ibm.ws.bootstrap.applications and allows them to be run. This time it is the IBM bootstrap environment that executes these mini-applications rather than the native Eclipse runtime, so the dependencies are different. Standard Java reflection code is used to load each of these mini-applications, as you can see in the stack traces.

Eclipse allows individual Eclipse applications, like applets within an application, to run as separate threads of control. These all extend the org.eclipse.core.runtime.applications extension point. If you enumerate all of the extensions found in the internal Eclipse platform registry, you can determine what WebSphere Application Server runs as separate distinct Eclipse applications; see Example 5-52 on page 256.

Example 5-52 Extensions in Eclipse platform registry

```
ID: org.eclipse.core.runtime.applications
Extension:com.ibm.ws.bootstrap.WSLauncher
Extension:com.ibm.ws.debug.osgi.StartConsole
Extension:com.ibm.ws.debug.osgi.Noop
Extension:com.ibm.ws.runtime.startWsServer
Extension:org.eclipse.ant.core.antRunner
```

The keys to WebSphere Application Server startup itself here are the `com.ibm.ws.bootstrap.WSLauncher` and `com.ibm.ws.runtime.startWsServer` classes. Ultimately, the latter of these results, due to the strategy pattern implementation, makes use of code inside the `com.ibm.ws.runtime.WsServerImpl` class that boots the containers within WebSphere Application Server and makes platform-specific decisions as how the environment should be configured. These applications extend the `org.eclipse.core.runtime.applications` extension point and so are designed to run on the base Eclipse platform.

Finally, you can distinguish the mini-applications that can run on the base Eclipse platform from those that run on the WebSphere Application Server runtime platform. This can be determined by enumerating the Eclipse registry to examine the extensions for the `com.ibm.ws.bootstrap.applications` extension point, as shown in Example 5-53.

Example 5-53 Examining extensions for com.ibm.ws.bootstrap.applications extension point

```
ID: com.ibm.ws.bootstrap.applications
Extension:com.ibm.ws.migration.WebSphere Application Server PreUpgrade
Extension:com.ibm.ws.migration.WebSphere Application Server PostUpgrade
Extension:com.ibm.ws.migration.ConvertScriptCompatibility
Extension:com.ibm.ws.migration.ClientUpgrade
Extension:com.ibm.ws.runtime.RetrieveSigners
Extension:com.ibm.ws.runtime.WsAdmin
Extension:com.ibm.ws.runtime.CollectManagedObjectMetadata
Extension:com.ibm.ws.runtime.WsServerLauncher
Extension:com.ibm.ws.runtime.WsServer
Extension:com.ibm.ws.runtime.WsServerStop
Extension:com.ibm.ws.runtime.ServerStatus
Extension:com.ibm.ws.runtime.LaunchBatchCompiler
Extension:com.ibm.ws.runtime.LaunchWSAnt
Extension:com.ibm.ws.runtime.BackupConfigUtility
Extension:com.ibm.ws.runtime.RestoreConfigUtility
Extension:com.ibm.ws.runtime.FindEJBTimersCommand
```

```
Extension:com.ibm.ws.runtime.CancelEJBTimersCommand
Extension:com.ibm.ws.runtime.LaunchClient
Extension:com.ibm.ws.runtime.LaunchClientApi
Extension:com.ibm.ws.runtime.NodeFederationUtility
Extension:com.ibm.ws.runtime.NodeUninstallPrep
Extension:com.ibm.ws.runtime.NodeCleanupUtility
Extension:com.ibm.ws.runtime.NodeRemovalUtility
Extension:com.ibm.ws.runtime.NodeSyncUtility
Extension:com.ibm.ws.runtime.NodeRenameUtility
Extension:com.ibm.ws.runtime.DumpExtensionRegistry
Extension:com.ibm.ws.runtime.WsProfile
Extension:com.ibm.ws.runtime.WsProfileAdminListener
Extension:com.ibm.uddi.UDDIValueSet
```

Notice the `com.ibm.ws.runtime.WsServer` and `WsServerLauncher` classes that show how the core WebSphere Application Server container environment runs on top of the core WebSphere Application Server runtime, which itself runs on top of the Eclipse/OSGI platform runtime. Many of these extensions can be seen configured in the `plugin.xml` file for `com.ibm.ws.runtime_6.1.0.jar`.

In summary, there are two threads of control involved in the WebSphere Application Server startup up to this point:

- ▶ One thread of control sets up the environment, loads the Eclipse/Equinox/OSGI runtime, and then moves on to do further launching of the environment.
- ▶ After the Eclipse runtime has been started, the second thread of control enumerates the WebSphere Application Server plug-ins to find Eclipse RCP application plug-ins, extensions, and extension points from the enumerated Eclipse plug-ins, and OSGI bundles and services from those plug-ins. For each of these, the `MANIFEST.MF` or `plugin.xml` file is read to determine what must be started, what may be started later on request, and what is available to be built on.

All of this is used to set up the WebSphere Application Server thin registry or public extension registry in accord with the Eclipse runtime rules, and this can be accessed via JNDI from WebSphere Application Server applications, and the OSGI service registry.

Although the complex layout, duplication, and concurrency leads to confusion regarding how this works and to the impression that this is a work in progress, it is this extensible architecture that allows IBM to extend WebSphere Application Server with new features. And it will form the basis of a future extensible and more efficient WebSphere Application Server runtime.

The WebSphere Application Server runtime start up process is diagrammed in Figure 5-14.

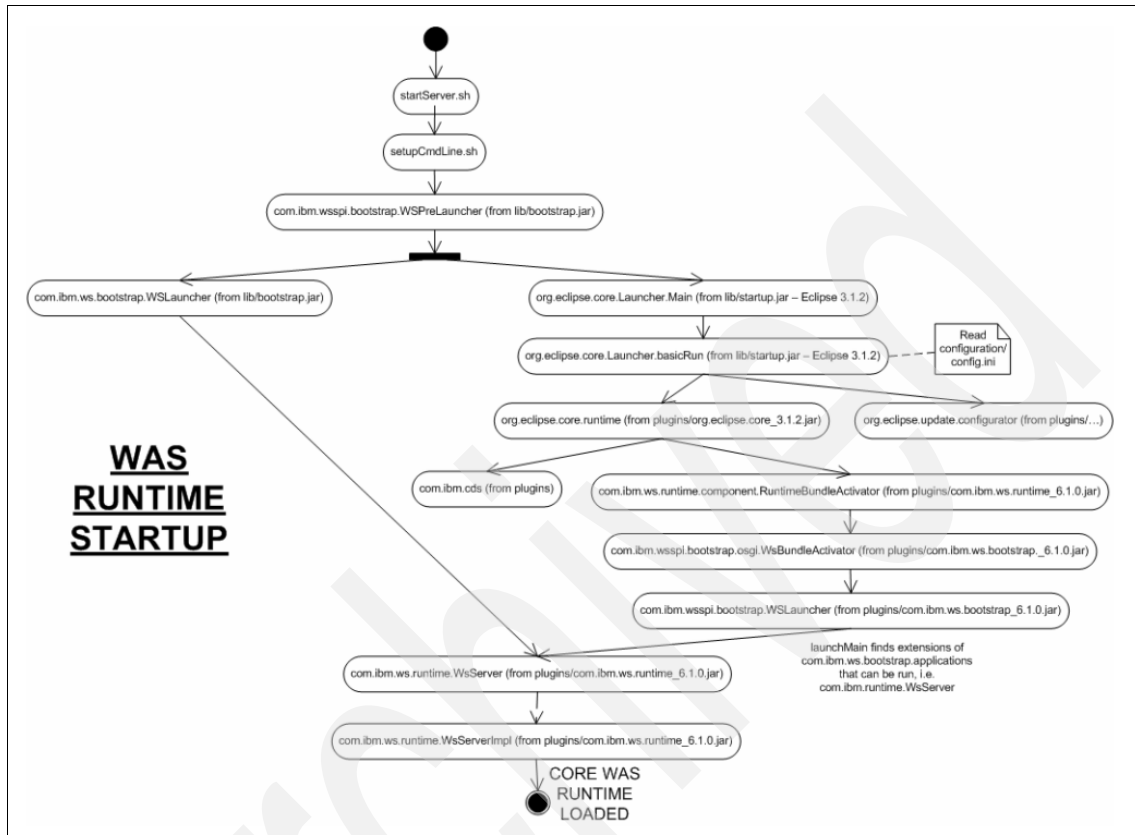


Figure 5-14 WebSphere Application Server runtime startup

At this point you have been shown how the Eclipse registry could be enumerated at runtime, but the process itself has not been explained. To do so, we must generate a plug-in of our own and persuade the WebSphere Application Server Eclipse platform environment to execute it. Note that this activity is *not* supported for WebSphere Application Server by IBM; we are simply demonstrating how the environment works.

We needed to use an Eclipse plug-in development environment, which is freely available from:

<http://www.eclipse.org>

First we generate a standard plug-in. Then, in the bundle Activator, we change the start method and use the Eclipse registry APIs to do the enumeration, as shown in Example 5-54.

Example 5-54 Custom plug-in

```
public void start(BundleContext context) throws Exception {
    super.start(context);
    System.out.println("Welcome:WebSphere Application Server View called...");

    // Get the details of the extension register and enumerate it
    try {
        FileWriter outlog = new FileWriter("/tmp/logs/WebSphere Application Server
View.log", true);
        IExtensionRegistry reg = Platform.getExtensionRegistry();
        outlog.write("Registry is: " + reg.toString() + "\n");
        for (IExtensionPoint point: reg.getExtensionPoints()){
            if (!point.getUniqueIdentifier().toLowerCase().equals("null")) {
                outlog.write("ID: " + point.getUniqueIdentifier() + "\n");
            }
        }
        for (IExtension ext: point.getExtensions()){
            if (!ext.getUniqueIdentifier().toLowerCase().equals("null")) {
                outlog.write("-->Extension:" + ext.getUniqueIdentifier() + "\n");
            }
        }
    }

    // Make sure that things get written
    outlog.flush();

    // Finally enumerate the bundles
    Bundle bundles[] = context.getBundles();
    for (Bundle bundle: bundles) {
        outlog.write("Bundle Name:"
            + ((bundle.getSymbolicName() != null) ? bundle.getSymbolicName() : "null")
            + ", ID: " + bundle.getBundleId() + ", State: " + bundle.getState()
            + "\n");
        for (ServiceReference sr: bundle.getRegisteredServices()) {
            outlog.write("\t-->Service Reference:" + sr.toString() + "\n");
            if (sr.getUsingBundles() != null) {
                for (Bundle b: sr.getUsingBundles()) {
                    outlog.write("\t---->Used By: " + ((b.getSymbolicName() != null)
                        ? b.getSymbolicName() : "null") + "\n");
                }
            }
        }
    }
}
```

```

    }
    }
    outlog.flush();
    outlog.close();
} catch (IOException ioe) {
    ioe.printStackTrace();
}
}
}

```

When we were ready to deploy, we exported the plug-in and place it in the /plugins directory under WebSphere Application Server. To ensure that our plug-in started when WebSphere Application Server started, we added an entry to the configuration/config.ini file, as shown in Example 5-55.

Example 5-55 Starting the custom plug-in WebSphere Application Server View

```

...
# The startlevel indicates the OSGi start level at which the bundle should run.
osgi.bundles=org.eclipse.core.runtime@2:start,
org.eclipse.update.configurator@3:start, WebSphere Application Server View@4:start
...

```

When WebSphere Application Server was now run, output was generated which showed the extension points that allowed the core Web services to be built upon; see Example 5-56. Extensions in feature packs can build on these extension points.

Example 5-56 WebSphere Application Server View plug-in output

```

ID: com.ibm.wsspi.extension.webservices
-->Extension:com.ibm.ws.runtime.wsaddressing_impl-webservices
-->Extension:com.ibm.ws.runtime.dynacache-webservices
-->Extension:com.ibm.ws.runtime.pmi_rm-webservices
-->Extension:com.ibm.ws.runtime.security_wssecurity-webservices
-->Extension:com.ibm.ws.runtime.webservices-webservices
-->Extension:com.ibm.ws.runtime.il8n-webservices
-->Extension:com.ibm.ws.runtime.transaction_impl-webservices

```

The new WebSphere Application Server 6.1 osgiConsole.sh script loads a command line that allows bundles to be enumerated, started, and stopped. It gives you a view of what WebSphere Application Server has loaded in terms of the OSGi view of bundles and services. It also allows you to see the relationships between the bundles and the services they register, as demonstrated in Example 5-57 on page 261.


```
osgi> status
Framework is launched.
```

```
id      Bundle Location
State   Bundle File Name
0       System Bundle
  ACTIVE org.eclipse.osgi.framework.internal.core.SystemBundleData@d580d58
1       initial@reference:file:plugins/com.ibm.cds_1.0.0.jar/
  RESOLVED BundleData for com.ibm.cds (1)
2       initial@reference:file:plugins/org.eclipse.core.runtime_3.1.2.jar/
  ACTIVE   BundleData for org.eclipse.core.runtime (2)
3       initial@reference:file:plugins/org.eclipse.update.configurator_3.1.0.jar/
  ACTIVE   BundleData for org.eclipse.update.configurator (3)
4       update@plugins/com.ibm.ws.bootstrap_6.1.0.jar
  ACTIVE   BundleData for com.ibm.ws.bootstrap (4)
5       update@plugins/com.ibm.ws.debug.osgi_6.1.0.jar
  ACTIVE   BundleData for com.ibm.ws.debug.osgi (5)
6       update@plugins/com.ibm.ws.ejbportable_6.1.0.jar
  RESOLVED BundleData for com.ibm.ws.ejbportable (6)
7       update@plugins/com.ibm.ws.emf_2.1.0.jar
  RESOLVED BundleData for com.ibm.ws.emf (7)
8       update@plugins/com.ibm.wsspi.extension_6.1.0.jar
  RESOLVED BundleData for com.ibm.wsspi.extension (8)
9       update@plugins/com.ibm.ws.runtime.gateway_6.1.0.jar
  ACTIVE   BundleData for com.ibm.ws.runtime.gateway (9)
10      update@plugins/com.ibm.ws.migration_6.1.0.jar
  RESOLVED BundleData for com.ibm.ws.migration (10)
11      update@plugins/org.eclipse.jdt.core_3.1.2.jar
  RESOLVED BundleData for org.eclipse.jdt.core (11)
12      update@plugins/com.ibm.ws.security.crypto_6.1.0.jar
  RESOLVED BundleData for com.ibm.ws.security.crypto (12)
13      update@plugins/com.ibm.ws.runtime_6.1.0.jar
  ACTIVE   BundleData for com.ibm.ws.runtime (13)
...
Registered Services
{org.osgi.service.packageadmin.PackageAdmin}={service.ranking=2147483647,
service.pid=0.org.eclipse.osgi.framework.internal.core.PackageAdminImpl,
service.vendor=Eclipse.org, service.id=1}
{org.osgi.service.startlevel.StartLevel}={service.ranking=2147483647,
service.pid=0.org.eclipse.osgi.framework.internal.core.StartLevelManager,
service.vendor=Eclipse.org, service.id=2}
{javax.xml.parsers.SAXParserFactory}={service.id=3}
```

```

{javax.xml.parsers.DocumentBuilderFactory}={service.id=4}
{org.eclipse.osgi.service.datalocation.Location}={type=osgi.user.area, service.id=5}
{org.eclipse.osgi.service.datalocation.Location}={type=osgi.instance.area,
service.id=6}
{org.eclipse.osgi.service.datalocation.Location}={type=osgi.configuration.area,
service.id=7}
{org.eclipse.osgi.service.datalocation.Location}={type=osgi.install.area,
service.id=8}
{org.eclipse.osgi.service.environment.EnvironmentInfo}={service.ranking=2147483647,
service.pid=0.org.eclipse.core.runtime.internal.adaptor.EclipseEnvironmentInfo,
service.vendor=Eclipse.org, service.id=9}
{org.eclipse.osgi.service.resolver.PlatformAdmin}={service.ranking=2147483647,
service.pid=0.org.eclipse.osgi.framework.adaptor.core.StateManager,
service.vendor=Eclipse.org, service.id=10}
{org.eclipse.osgi.service.pluginconversion.PluginConverter}={service.ranking=21474836
47, service.pid=0.org.eclipse.core.runtime.internal.adaptor.PluginConverterImpl,
service.vendor=Eclipse.org, service.id=11}
{org.eclipse.osgi.service.urlconversion.URLConverter}={service.ranking=2147483647,
service.pid=0.org.eclipse.core.runtime.internal.adaptor.URLConverterImpl,
service.vendor=Eclipse.org, service.id=12}
{org.eclipse.osgi.framework.console.CommandProvider}={service.ranking=2147483647,
service.pid=0.org.eclipse.core.runtime.internal.adaptor.EclipseCommandProvider,
service.vendor=Eclipse.org, service.id=13}
{org.eclipse.osgi.framework.log.FrameworkLog}={service.ranking=2147483647,
service.pid=0.org.eclipse.core.runtime.adaptor.EclipseLog,
service.vendor=Eclipse.org, service.id=14}
{org.eclipse.osgi.framework.log.FrameworkLog}={service.ranking=-2147483648,
performance=true, service.pid=46org.eclipse.core.runtime.adaptor.EclipseLog,
service.vendor=Eclipse.org, service.id=15}
{org.eclipse.osgi.service.localization.BundleLocalization}={service.ranking=214748364
7, service.pid=0.org.eclipse.core.runtime.internal.adaptor.BundleLocalizationImpl,
service.vendor=Eclipse.org, service.id=16}
{org.eclipse.osgi.framework.console.CommandProvider}={service.ranking=2147483647,
service.id=17}
{java.lang.Runnable}={name=splashscreen, service.id=18}
{org.eclipse.core.runtime.IExtensionRegistry}={service.id=19}
{org.osgi.service.url.URLStreamHandlerService}={url.handler.protocol=[Ljava.lang.Stri
ng;@42154215, service.id=20}
{org.eclipse.osgi.service.runnable.ParameterizedRunnable}={eclipse.application=default,
service.id=21}
{org.eclipse.update.configurator.IPlatformConfigurationFactory}={service.id=22}

```

The output in Example 5-57 on page 261 shows that most of the OSGI services are registered by the Eclipse/OSGI/Equinox runtime itself, with a particular

emphasis on XML (javax.xml.parsers.*) and URL management-related services that provide a general set of infrastructure functions to the rest of the environment.

Note also that the core WebSphere Application Server runtime is bundle 13 in this running WebSphere Application Server environment. From the OSGI Console, a garbage collection can also be manually requested by using the `gc` command, although the supportability and real use of this in a production setting is yet to be determined.

Although for demonstration purposes in this book, we looked at the unsupported use of Eclipse/OSGI bundles below WebSphere Application Server as a mechanism for seeing how WebSphere Application Server loading works, IBM does support the use of Eclipse/OSGI bundles running in WebSphere Application Server on top of the exported WebSphere Application Server registry. This is covered in the WebSphere Application Server 6.1 documentation:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.express.iseries.doc/info/iserieexp/ae/welc_ref_dev_javadoc.html

So, from the WebSphere Application Server startup we obtained a view of the new Eclipse/OSGI architecture and saw how it provides an extensible base. The complexity of the startup path is due to this. Based on statements of direction from IBM as a whole, this is a work in progress as the Eclipse/OSGI (Equinox) architecture is adopted more widely.

Container startup

Previously, in 5.4.4, “WebSphere Application Server - startup and operation” on page 244, you examined the first part of the WebSphere Application Server startup sequence, which involved shell scripts and the Eclipse/OSGI/Equinox runtime, and learned how this starts the WebSphere Application Server runtime. This section examines how the WebSphere Application Server runtime startup creates the containers that form the basis of J2EE.

It is important to understand is that the core Eclipse/OSGI model is used extensively during the startup of the containers. Essentially, common code is provided in the `WsContainer/ContainerImpl` and `WsComponent` classes.

The `ContainerImpl` class is an implementation class that loads the `WsComponent` derived classes described in the `com.ibm.ws.runtime_6.1.0.jar` plugin.xml file for the given container, and maps the appropriate threads, transports and other runtime artifacts to the given environment. The target container that is being started is passed to the `WsContainer/ContainerImpl` implementation by the caller.

First, the core server environment itself is started up (that is, the management and runtime infrastructure). Then the application server containers are started. This can be seen in the stacks we saw earlier, which are displayed again here in Example 5-58, Example 5-59, and Figure 5-15 on page 265.

Example 5-58 WebSphere Application Server component startup

```
at com.ibm.ws.runtime.component.ContainerImpl.startComponents(ContainerImpl.java:977)
    at com.ibm.ws.runtime.component.ContainerImpl.start(ContainerImpl.java:673)
    at
com.ibm.ws.runtime.component.ApplicationServerImpl.start(ApplicationServerImpl.java:1
91)
    at
com.ibm.ws.runtime.component.ContainerImpl.startComponents(ContainerImpl.java:977)
    at com.ibm.ws.runtime.component.ContainerImpl.start(ContainerImpl.java:673)
    at com.ibm.ws.runtime.component.ServerImpl.start(ServerImpl.java:485)
    at com.ibm.ws.runtime.WsServerImpl.bootServerContainer(WsServerImpl.java:191)
    at com.ibm.ws.runtime.WsServerImpl.start(WsServerImpl.java:139)
    at com.ibm.ws.runtime.WsServerImpl.main(WsServerImpl.java:460)
```

Example 5-59 continues the component startup display.

Example 5-59 WebSphere Application Server component startup

```
at
com.ibm.ws.runtime.component.ContainerImpl.initializeComponent(ContainerImpl.java:133
8)
    at
com.ibm.ws.runtime.component.ContainerImpl.initializeComponents(ContainerImpl.java:11
36)
    at com.ibm.ws.runtime.component.ServerImpl.initialize(ServerImpl.java:347)
    at com.ibm.ws.runtime.WsServerImpl.bootServerContainer(WsServerImpl.java:177)
    at com.ibm.ws.runtime.WsServerImpl.start(WsServerImpl.java:139)
    at com.ibm.ws.runtime.WsServerImpl.main(WsServerImpl.java:460)
    at com.ibm.ws.runtime.WsServer.main(WsServer.java:59)
```

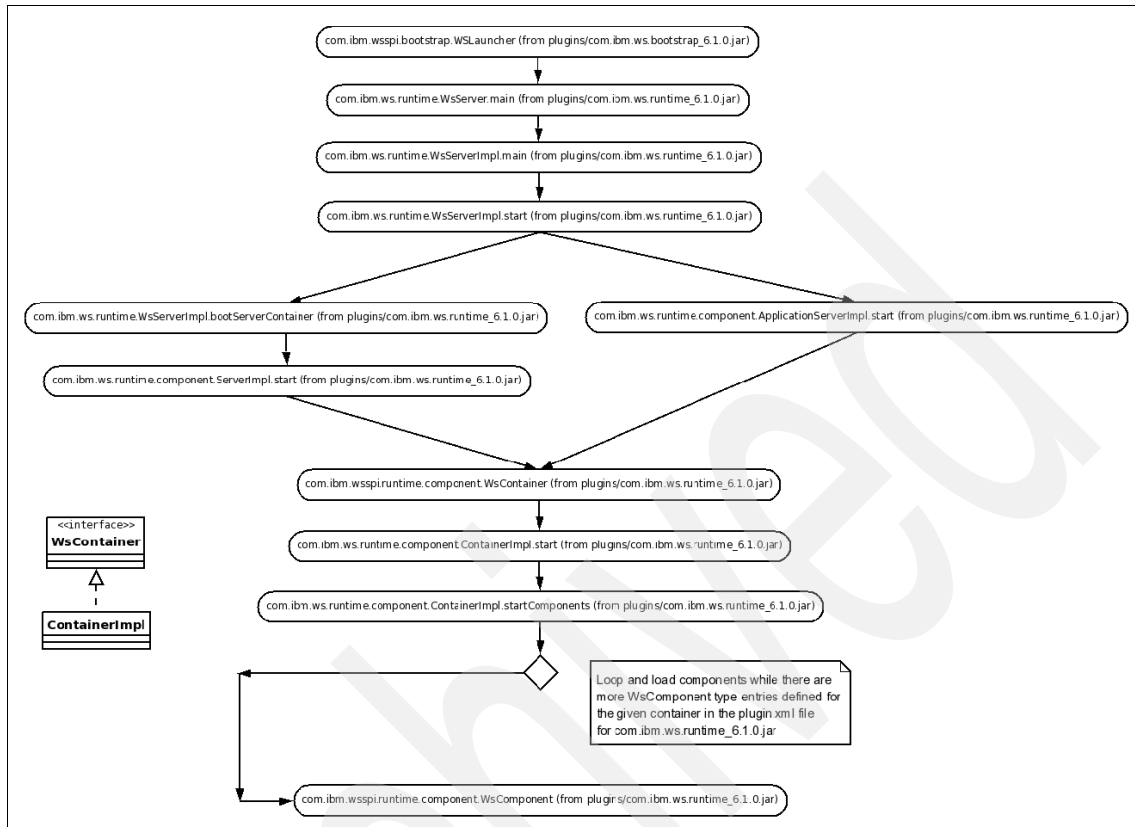


Figure 5-15 WebSphere Application Server container startup

Each component named in the appropriate section of the plugin.xml file related to the given container is initialized individually by the ContainerImpl class.

Now consider the two core containers. The EJB container is based around a CORBA ORB core and uses IIOP as a transport, with a requirement for thread pool management to handle multiple concurrent remote EJB requests. The Web container is based around a servlet engine core that uses HTTP as a transport, with a requirement for thread pool management to handle multiple concurrent HTTP requests. The commonality is why the mapping is done using the same core implementation to map configurations of containers, transports, and components in the plugin.xml file.

In the Web container, `com.ibm.wsspi.runtime.component.WsContainer/ContainerImpl` starts up a number of components (that support the `com.ibm.wsspi.runtime.WsComponent` interface) for the given container. The components are themselves Eclipse

plug-ins. The extension registry is enumerated to find them. When they are loaded, the service registry is also used.

The extensions and extension points are enumerated to get their configuration information (that is, startup order, platform, recovery information, and so on), and then they are sorted on the startup order information and started (using the start method for each component), with a preference for asynchronous startup if the component supports it. The WsContainer class can also start, stop, and initialize the components.

When the Web container starts up, it has a default response buffer size of 32 K but reads its configuration to resize this response buffer size. It also sets up a fixed size pool of 100 WCChannelLinks to represent each virtual connection, so this is a hard limit. These are inbound request/response connections only.

Each WCChannelLink uses underlying classes and services to read the request, process it, and respond. However, it tries to do as much asynchronously as possible to minimize latency and resource constraints. The underlying methods favored by servlet coders eventually map to the WCCRequestImpl and WCCResponseImpl classes for handling, where they read and write the underlying request and response streams. At a higher level the servlet classes and methods themselves are mapped to the classes in `com.ibm.ws.http.*` package classes that implement the standard classes. This is all controlled in the `com.ibm.ws.webcontainer.channel.WCChannel` class.

The `com.ibm.ws.http.HTTPServer` class sets up a thread pool and an object pool of connections, and then does all of the things normally expected of a Web server in responding to connections. This is the internal HTTP server for representing J2EE JSP and servlet-based Web applications that must be fronted by a separate HTTP server process for support and security reasons. The `HttpTransport` class does a significant amount of the underlying work.

The `com.ibm.ws.http.channel.impl.HttpChannelConfig` reads the HTTP setup parameters, which are distinct from those of the Web container it runs on, and sets those up.

The channel factory makes use of the underlying TCP connection factory in `com.ibm.wsspi.tcp.channel` to get its environment. The `com.ibm.ws.http.channel` package classes make use of the `com.ibm.wsspi.http.channel` and `com.ibm.wsspi.tcp.channel` implementations to do the real work, using the strategy pattern interface which is seen throughout the WebSphere Application Server code.

The `com.ibm.ws.tcp.channel.impl` package code again uses the underlying implementation in the equivalent `wsspi` classes, but makes use of Asynchronous IO wherever possible, and using native platform code if available.

So, how is all of this related? The XML file used to start the containers from WsServerImpl starts the WsContainer/ContainerImpl containers that load the components using the extension registry mentioned. The containers load a component manager that uses the name of the subsystem to load the appropriate configuration section from the plugin.xml file for that subsystem, and then enumerate the “component” entries to load each component within it.

So, the base server that provides basic policy-less services (like the thread pool manager, the HAManager, the EndPoint Manager, and so on) loads the components in the section `com.ibm.wsspi.extension.server-startup`. One of these is the Application Server subsystem, which starts loading as a result of the entry in this Server subsystem configuration.

Then the core application server services that provide the Web services support, the core object pool, the dynacache, the extension registry, and the EJBContainer environment, are loaded using the configuration in the `com.ibm.wsspi.extension.application-server.startup` section to identify and load the components from that environment in the appropriate order.

The EJB container starts loading at this point in the same way as it is named as a component in this section, and starts immediately after the client services startup. The EJB container is followed by the HA Manager subsystems (Group Manager, Agent, and so on), the Web container, the scheduler service, and finally the administration service (JMX connectors and the plug-in configuration service for the Web server plug-in).

The overall components that a WebSphere Application Server instance loads and runs are listed in `com.ibm.wsspi.extension.server-components`.

These details are all held in the `com.ibm.ws.runtime_6.1.0.jar` plugin in the `plugin.xml` file.

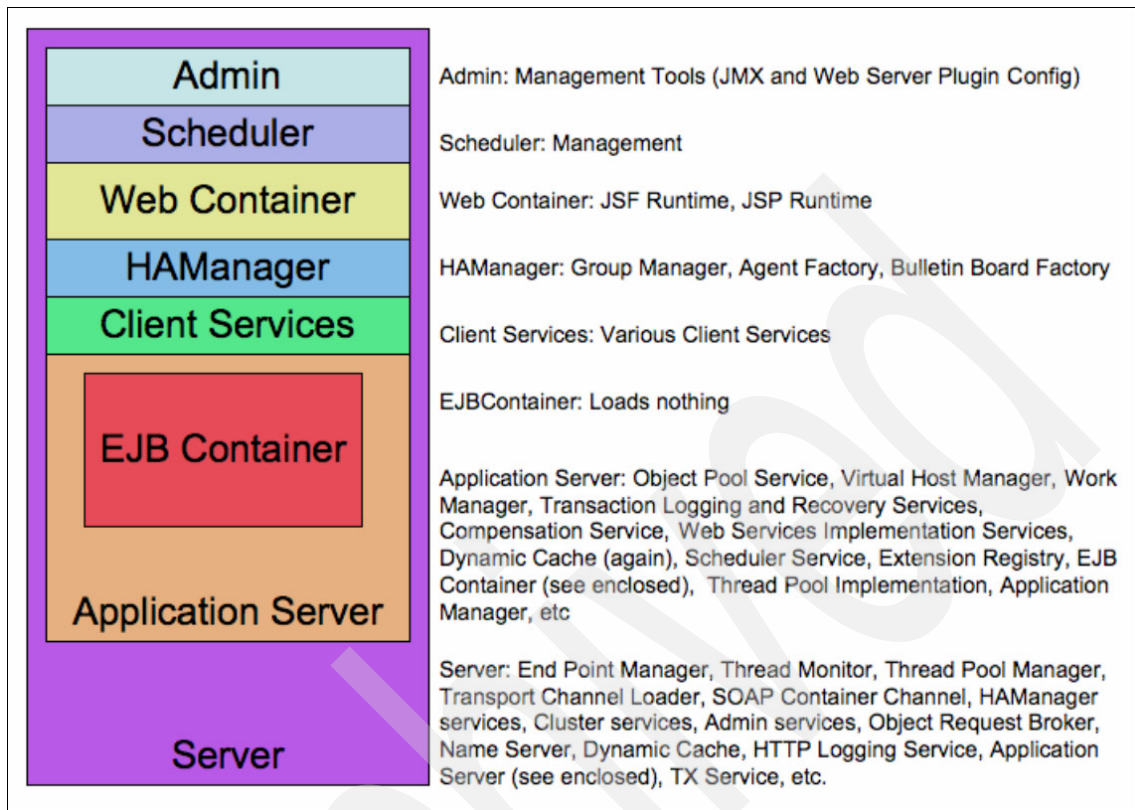


Figure 5-16 WebSphere Application Server containers

First, the base runtime services identified by the extension point `com.ibm.wsspi.extension.server-startup` are started by enumerating the components within that grouping. Although most platforms use a single JVM process to run WebSphere Application Server, z/OS uses multiple processes called servants, controls, and adjuncts to emulate the standard J2EE behavior. So, the `processType` attribute is used to tell the environment how the different components should be started on the z/OS platform (although this is irrelevant to distributed platforms such as AIX).

The `platform` attribute identifies components that should only be run on specific platforms such as z/OS, and for AIX, the distributed value or absence of the attribute identifies what is loaded. The `startup` attribute identifies the order in which the components load, with the lowest value loading first.

The `recoveryType` attribute is another z/OS-specific artifact. Note that there are also iSeries-specific components, showing that although the zSeries and iSeries

core WebSphere Application Server shipped binaries are the same, the code that is run is *not* the same as the other platforms; see Example 5-60.

Example 5-60 Base runtime services startup: lower level components

```
<extension point="com.ibm.wsspi.extension.server-startup">
  <components>
    <component startup="1000"
class="com.ibm.ws.runtime.component.VirtualHostMgrImpl" platform="zos"
processType="Control"/>
    <component startup="1999"
class="com.ibm.ws.runtime.component.ThreadMonitorImpl"
recoveryType="Control,Servant"/>
    <component startup="2000" class="com.ibm.ws.runtime.component.EndpointMgrImpl"
recoveryType="Control,Servant"/>
    <component startup="2000"
class="com.ibm.ws.runtime.component.ThreadPoolMgrImpl"
type="services/threadpoolmanager.ThreadPoolManager" configurationDataRequired="false"
recoveryType="Control,Servant"/>
    <component startup="2000"
class="com.ibm.ws.runtime.component.TransportChannelLoaderImpl"
platform="distributed,os400"/>

...

    <component startup="5000" type="services/orb.ObjectRequestBroker"
recoveryType="Control,Servant"/>
    <component startup="5400" class="com.ibm.ws.hamanager.proxy.ProxyImpl"
platform="zos" processType="Adjunct,Servant"/>
    <component startup="5400"
class="com.ibm.ws.hamanager.runtime.CoordinatorComponentImpl" processType="Control"/>
    <component startup="5500"
class="com.ibm.ws.management.component.MBeanActivationImpl" platform="zos"
recoveryType="Control"/>

...

    <component startup="6600" type="services/applicationserver.DynamicCache"
processType="Control"/>

...
<component startup="13000" class="com.ibm.ws.odc.bb.BBMgr"
processType="Control,Servant"/>
    <component startup="13010" class="com.ibm.ws.odc.ws390.proxy.ODCProxyManager"
processType="Control,Servant"/>
```

```

        <component startup="13500" class="com.ibm.ws390.jfap.JFAPProxyComponentImpl"
platform="zos" processType="Control"/>
        <component startup="14000" class="com.ibm.ws.dwlmc.client.DWLMClientImpl"
processType="Control,Servant"/>
        <component startup="14500"
class="com.ibm.ws.cache.drs.ws390.ControllerCacheServiceImpl" platform="zos"
processType="Control"/>
        <component startup="15000"
class="com.ibm.ejs.container.drs.ws390.SfControllerServiceImpl" platform="zos"
processType="Control"/>
        <component startup="15000"
class="com.ibm.ws.messaging.component.JmsAdminService"
processType="Adjunct,Servant"/>

...
        <component startup="25000" class="com.ibm.ws.runtime.component.TxServiceImpl"
platform="zos" processType="Control" recoveryType="Control"/>
        <component startup="100000"
class="com.ibm.ws.taskmanagement.mapper.TaskManagementServiceMapper"
processType="Control"/>
        <component startup="100100"
class="com.ibm.ws.taskmanagement.task.TaskManagementMBean" processType="Control"/>
        <component startup="999999"
class="com.ibm.ws.wim.management.UserManagementProcess"/>
        <component startup="2147483647" class="com.ibm.ws.runtime.component.WLCImpl"
platform="distributed,os400"/>
        <component startup="2147483647"
class="com.ibm.ws.soapchannel.monitor.component.SOAPMonitorChannelComponentImpl"
platform="zos" processType="Control"/>
    </components>
</extension>

```

After the base runtime lower level support components have been loaded, the application server container itself is loaded. This provides the generic container level services that the J2EE containers (such as the Web and EJB containers) use and sit within.

This is identified by the `com.ibm.wsspi.extension.applicationserver-startup` section of the `plugin.xml` file. The `processType`, `platform`, and `startup` attributes have the same meaning as before; see Example 5-61.

Example 5-61 Base runtime services startup: application server container

```

<extension point="com.ibm.wsspi.extension.applicationserver-startup">

```

```

    <components>
      <component startup="1000" type="objectpoolservice.ObjectPoolService"
processType="Adjunct,Servant"/>
      <component startup="1000" class="com.ibm.ws.runtime.component.MetadataMgrImpl"
processType="Adjunct,Servant"/>
      <component startup="1000"
class="com.ibm.ws.runtime.component.VirtualHostMgrImpl"/>
      <component startup="1001"
class="com.ibm.ws.wccm.services.pme.metadata.impl.MetadataHelperServiceImpl"
processType="Adjunct,Servant"/>

    ...

      <component startup="5000"
class="com.ibm.ws.webservices.component.WSServerImpl"/>
    ...

      <component startup="6600" type="services/applicationserver.DynamicCache"
processType="Servant"/>
      <component startup="6800"
class="com.ibm.wkplc.extensionregistry.wasservice.ExtensionRegistryService"/>
      <component startup="7000"
type="components/applicationserver.ejbcontainer.EJBContainer"
processType="Adjunct,Servant"/>

    ...

      <component startup="8700" class="com.ibm.ws.runtime.component.ThreadPoolImpl"
processType="Adjunct,Servant"/>
      <component startup="9000" type="components" processType="Adjunct,Servant"/>
      <component startup="10000"
class="com.ibm.ws.performance.tuning.serverAlert.ServerRuleDriverMBean"
processType="Servant"/>
      <component startup="10000"
class="com.ibm.ws.runtime.component.ApplicationMgrImpl"
processType="Adjunct,Servant"/>
      <component startup="10000"
class="com.ibm.ws.runtime.component.LibraryMgrImpl"/>
      <component startup="10001"
class="com.ibm.ws.cluster.runtime.ApplicationServerRuntimeImpl"
processType="Adjunct,Servant"/>

    ...

      <component startup="99999"
class="com.ibm.ws.wccm.services.pme.PMECheckServerConfigService"
processType="Adjunct,Servant"/>
    </components>

```

</extension>

As illustrated in Example 5-62, the EJB container startup configuration shows that the EJB container code does not have any additional extensions to run beyond the code in com.ibm.ws.runtime_6.1.0.jar file and the separate com.ibm.ws.ejbportable_6.1.0.jar file that extends this.

Example 5-62 EJB container startup

```
<extension point="com.ibm.wsspi.extension.ejbcontainer-startup">
  <components>
    <component startup="1000" type="services" />
  </components>
</extension>
```

The com.ibm.ws.ejbportable_6.1.0.jar file does not have its own plugin.xml file, but it does not need one. As shown in Example 5-63, its MANIFEST.MF file identifies itself as a singleton, as would be expected. However, it uses the Fragment-Host attribute to identify itself as an extension of the base WebSphere Application Server com.ibm.ws.runtime_6.1.0.jar bundle, so the plugin.xml file from that applies.

Example 5-63 EJB container manifest

```
Bundle-Localization: fragment
Bundle-ManifestVersion: 2
Bundle-Name: ejbportable
Bundle-SymbolicName: com.ibm.ws.ejbportable; singleton:=true
Bundle-Vendor: IBM
Bundle-Version: 6.1.0
Fragment-Host: com.ibm.ws.runtime; bundle-version=6.1.0
```

The Web container can support additional components for JSF and JSP, but currently this configuration is commented out in the plugin.xml file for the WebSphere runtime. So again, it would appear that only the base code in com.ibm.ws.runtime_6.1.0.jar and the com.ibm.ws.webcontainer_2.0.0.jar file that extends this are used.

However, examination of the plugin.xml file for the com.ibm.ws.webcontainer_2.0.0.jar file shows, in Example 5-64 on page 273, that these components and their startup have been copied to this file and uncommented. With future Web services implementations, this is likely to be a heavily used configuration item in future WebSphere Application Server releases.

com.ibm.ws.runtime_6.1.0.jar...

```
<!-- extension point="com.ibm.wsspi.extension.webcontainer-startup">
  <components>
    <component startup="2000" type="services" />
    <component startup="2147483647"
class="com.ibm.ws.jsf.extprocessor.JSFComponentImpl" />
    <component startup="2147483647" class="com.ibm.ws.jsp.runtime.JspComponentImpl"
/>
  </components>
</extension -->
```

com.ibm.ws.webcontainer_2.0.0.jar...

```
<extension point="com.ibm.wsspi.extension.webcontainer-startup">
  <components>
    <component startup="2000" type="services" />
    <component startup="2147483647"
class="com.ibm.ws.jsf.extprocessor.JSFComponentImpl"/>
    <component startup="2147483647"
class="com.ibm.ws.jsp.runtime.JspComponentImpl" />
  </components>
</extension>
```

Other extensions for extension clients and the scheduler service are similarly configured to these containers. Most importantly, from the perspective of WebSphere Application Server, Network Deployment is the HAManager configuration. Without HAManager and the Distributed Replication Services, WebSphere Application Server ND has the same functionality as the standalone WebSphere Application Server base (although the management is still different.)

The Group Manager, Agent Factory and Bulletin Board Factory components are key to keeping the cluster functioning as a single entity with appropriate failover of singleton components and resilience; see Example 5-65.

```
<extension point="com.ibm.wsspi.extension.hamanager-startup">
  <components>
    <component startup="5401"
class="com.ibm.ws.hamanager.runtime.GroupManagerComponentImpl" />
    <component startup="5402"
class="com.ibm.ws.hamanager.runtime.AgentFactoryComponentImpl" />
```

```

        <component startup="5403"
class="com.ibm.ws.hamanager.runtime.BulletinBoardFactoryComponentImpl" />
    </components>
</extension>

```

The com.ibm.wsspi.extensionXXXXX startups bring the thread pools and object pools into life, as well as the TCP transport channel loader. When this is loaded, the extension class framework configuration in the plugin.xml file is used to relate the configuration, interface, and runtime classes mentioned, thus distinguishing between the protocol handlers and the connectors, as shown in Example 5-66.

Example 5-66 Extension class framework configuration in plugin.xml

```

<extension
    point="com.ibm.wsspi.extension.channel-framework-channel-type">
    <channel
        kind="ACCEPTOR"

configurationClass="com.ibm.websphere.models.config.channelservice.channels.DCSInboundChannel"
        runtimeClass="com.ibm.ws.hamanager.channel.DCSChannelFactory"
        deviceInterface="com.ibm.wsspi.tcp.channel.TCPConnectionContext"
        id="DCSInboundChannel"
        validatorClass="com.ibm.ws.hamanager.channel.DCSChannelValidator"/>
    <channel
        kind="PROTOCOL"

configurationClass="com.ibm.websphere.models.config.channelservice.channels.HTTPInboundChannel"
        runtimeClass="com.ibm.ws.http.channel.inbound.impl.WSHttpInboundChannelFactory"
        deviceInterface="com.ibm.wsspi.tcp.channel.TCPConnectionContext"

applicationInterface="com.ibm.wsspi.http.channel.inbound.HttpInboundServiceContext"
        id="HTTPInboundChannel"

validatorClass="com.ibm.ws.http.channel.validation.HttpInboundChannelValidator"/>
    <channel
        kind="PROTOCOL"
        deviceInterface="com.ibm.wsspi.tcp.channel.TCPConnectionContext"

configurationClass="com.ibm.websphere.models.config.channelservice.channels.HTTPOutboundChannel"

runtimeClass="com.ibm.ws.http.channel.outbound.impl.WSHttpOutboundChannelFactory"

```

```

applicationInterface="com.ibm.wsspi.http.channel.outbound.HttpOutboundServiceContext"
    id="HTTPOutboundChannel"

validatorClass="com.ibm.ws.http.channel.validation.HttpOutboundChannelValidator"/>

...
    <channel
        kind="PROTOCOL"
        deviceInterface="com.ibm.wsspi.tcp.channel.TCPConnectionContext"

configurationClass="com.ibm.websphere.models.config.channelservice.channels.SSLInboundChannel"
    runtimeClass="com.ibm.ws.ssl.channel.impl.WSSSLChannelFactory"
    applicationInterface="com.ibm.wsspi.tcp.channel.TCPConnectionContext"
    id="SSLInboundChannel"
    validatorClass="com.ibm.ws.ssl.channel.impl.SSLChannelValidator"/>
    <channel
        kind="PROTOCOL"
        deviceInterface="com.ibm.wsspi.tcp.channel.TCPConnectionContext"

configurationClass="com.ibm.websphere.models.config.channelservice.channels.SSLOutboundChannel"
    runtimeClass="com.ibm.ws.ssl.channel.impl.WSSSLChannelFactory"
    applicationInterface="com.ibm.wsspi.tcp.channel.TCPConnectionContext"
    id="SSLOutboundChannel"
    validatorClass="com.ibm.ws.ssl.channel.impl.SSLChannelValidator"/>
    <channel
        kind="ACCEPTOR"
        deviceInterface="com.ibm.wsspi.tcp.channel.TCPConnectionContext"

configurationClass="com.ibm.websphere.models.config.channelservice.channels.ORBinboundChannel"
    runtimeClass="com.ibm.ws.iiop.channel.impl.IIOPInboundChannelFactory"
    id="ORBinboundChannel"
    validatorClass="com.ibm.ws.orb.channel.impl.ORBChannelValidator"/>
    <channel
        kind="CONNECTOR"

configurationClass="com.ibm.websphere.models.config.channelservice.channels.TCPInboundChannel"
    runtimeClass="com.ibm.ws.tcp.channel.impl.WSTCPChannelFactory"
    applicationInterface="com.ibm.wsspi.tcp.channel.TCPConnectionContext"
    id="TCPInboundChannel"
    validatorClass="com.ibm.ws.tcp.channel.impl.TCPChannelValidator"

```

```

        platform="distributed,os400"/>

...
<channel
    kind="CONNECTOR"

    configurationClass="com.ibm.websphere.models.config.channelservice.channels.UDPOutboundChannel"

    runtimeClass="com.ibm.ws.udp.channel.outbound.impl.WSUDPOutboundChannelFactory"
        applicationInterface="com.ibm.wsspi.udp.channel.UDPContext"
        id="UDPOutboundChannel"

    validatorClass="com.ibm.ws.udp.channel.validation.UDPOutboundChannelValidator"/>
</extension>

```

Similar configuration and startup behavior is used for the EJB container, although the communications are more complex because some communications use a JNI configuration and shared object for IIOP. The plugin.xml file shows this and similar usage for the Service Integration Bus and other WebSphere Application Server subsystems.

5.5 WebSphere Application Server high availability deployments

The subject of deploying WebSphere Application Server in high availability configurations is covered in multiple places in this book. This section explains how to make WebSphere Application Server resilient and available by using the mechanism for high availability, consisting of Java components that are managed inside the WebSphere Application Server JVM process and those that are external to the WebSphere Application Server JVM process.

Within the WebSphere Application Server JVM process, high availability is based on components that are active-active within a cluster, and stateless or made consistent across the cluster using the Data Replication Service (DRS) fast message queuing mechanism between cluster instances. Those that have definite state that must exist only once in the cluster (are singletons) managed by the WebSphere Application Server HAManager.

The fast cluster communications underlying DRS make use of the Reliable Multicast Messaging (RMM) transport, which speeds the notifications between Web containers of session information and EJB containers of transactional state.

Outside the WebSphere Application Server JVM process, standard platform-specific tools such as High Availability Cluster Multi-Processing (HACMP) are used to manage the AIX processes and subsystems.

It is important to keep these two different mechanisms as independent as possible, because they will fail over at different rates and with different heartbeats and criteria. Thus, for WebSphere Application Server environments where an external WebSphere MQ implementation is used, HACMP should be used for failover. For environments using the Service Integration Bus default messaging provider, the Java implementation should use the HAManager built into WebSphere Application Server.

This is best understood with an enterprise-level example of a high availability, active-active, high volume, online transaction processing application and its infrastructure. Starting with Figure 5-17 on page 278, we consider the high level layout of the components, and then look at how each of them works.

This environment consists of three data centers. Two are mirror images of each other and contain a Web tier, application server tier, a database tier and NAS devices. The third data center contains simply a NAS device to provide quorum facilities to avoid a “split-brain scenario” if communications are lost between the two data centers. It is assumed that the entire environment consists of active-active components, to maximize performance and resilience (although this requires effort to achieve, in practice).

Requests from the user community come into an external load balancer that load balances across BladeCenter® environments in the two main centers. These requests are forwarded by the Web server tier to the application server tier for handling.

The application server tier logs any state to the NAS device in transaction logs (in a similar manner to that for DBMS's), and shares information between application server cluster members. Data is provided for the system from the database tier, which also makes use of the NAS devices for logs and quorum maintenance although a traditional IBM DS/8000 SAN environment provides core data storage.

This is simplified from real world environments because it ignores any use of proxies and edge components that many environments would find desirable, ignores any layering of the application server tier into Web and EJB layers, and ignores any earlier system integration. However, for most e-Commerce environments, this would provide a resilient and high performance infrastructure.

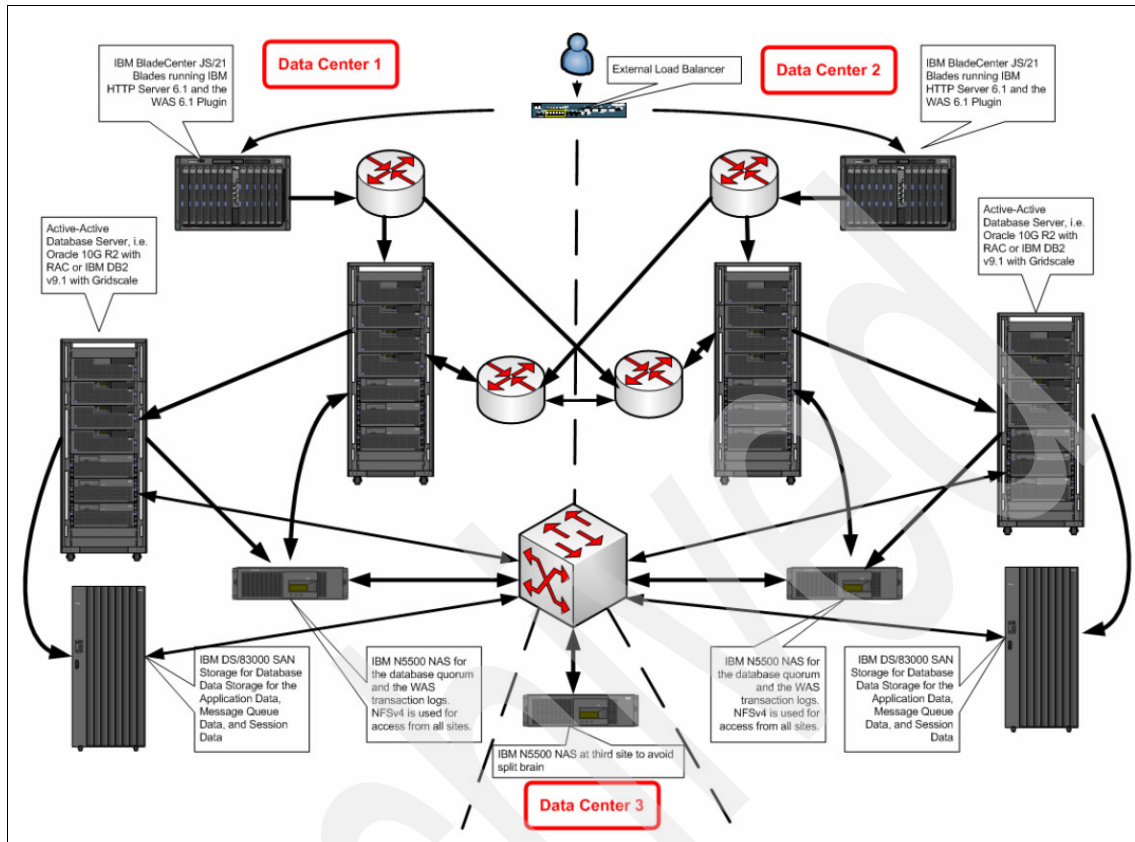


Figure 5-17 Sample WebSphere Application Server high availability e-Commerce architecture

To set up the clustering and manage the environment, a systems management and WebSphere Deployment Manager environment is also required. This can be configured in a number of ways, with mirrors within a data center and mirrors across data centers, but one technique is to use a System p 520 or 550 type machine and split it into partitions for all of the management software.

Management software tends to be passive rather than active from the perspective of the online transactions, and so does not need to support large loads. Centralizing the management software onto a single machine also lends itself to supporting scheduled upgrades where a failover of the entire environment is forced to the passive DR machine, and the formerly active machine is upgraded for all management software.

Thus, there is a passive machine sitting in one data center that is failed over to from the primary management partitions using HACMP in each of the management partitions. This active-passive arrangement for the management

environments does not affect the active-active nature of the transactional environment.

As shown in Figure 5-18, note the use of a spare partition on this environment to support upgrades of the operating system and any one of the management packages by cloning the original and upgrading it before setting it up to take over the original function. This minimizes risk in upgrades because the original is left in place until the new environment is proven, with only physical resources moved between the partitions to recover.

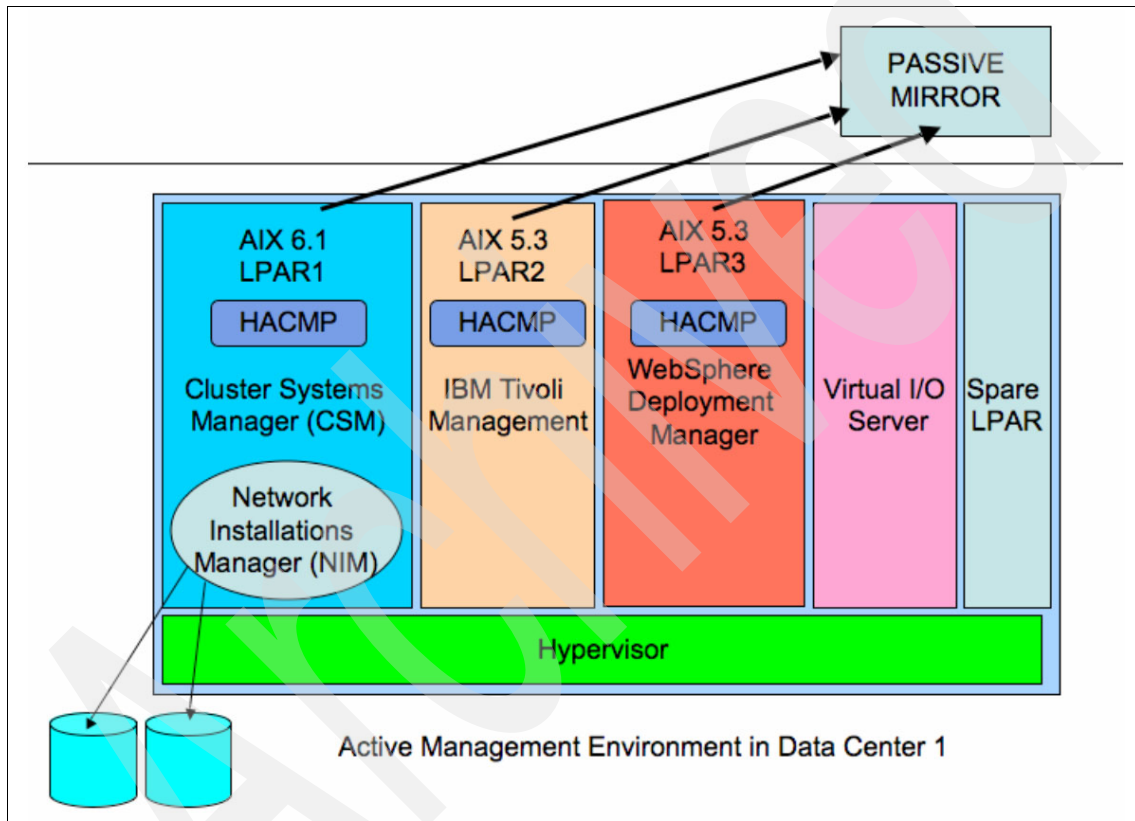


Figure 5-18 Active Management Environment

5.5.1 WebSphere cluster and cell configuration

To configure the WebSphere Application Server environment for high availability, a cell is set up (or two cells, if internal and external user traffic to the application set are to be separated for security reasons), which is the unit of management. A unit of failover with WebSphere Application Server - Network Deployment is a cell, and there can be multiple clusters per cell.

Within the cell two separate clusters are set up and instances created across partitions in both data centers. The number of instances (formerly known as clones) should be calculated by determining how long an average transaction will take and using this with the peak-of-peak number of transactions required multiplied by two or the number of identical data centers in use (assuming the infrastructures are the same in each).

Why multiply by two, you may wonder? Because this is a load balanced environment and at peak, each data center should carry half (or $1 / \text{number of data centers}$) the load. However, if a data center or its communications are lost, then the remaining data center (or data centers) must carry the whole load.

This is best explained with an example, as shown in Figure 5-19 on page 281. Assume you have two data centers, and each data center has a single System p570 with two WebSphere Application Server partitions (to support easy WebSphere Application Server upgrades). Your application needs to handle a peak load of 360,000 transactions per hour, and each transaction takes 2 seconds, of which the whole time is spent actively processing (that is, a calculations engine). So, 360,000 transactions per hour is 100 transactions per second, but because each transaction takes 2 seconds, there will be 200 transactions in flight at any one time.

Each WebSphere Application Server Web container is limited to 50 threads, so each can handle a maximum of 50 in-flight transactions. Therefore, you need to have 4 instances to support your load.

To account for loss of a cluster and loss of a complete data center where the remaining cluster or data center will have to take over the complete load, you need to have 4 instances per cluster split across two data centers and two clusters; that is, 8 instances. This is configured by connecting to the active WebSphere Deployment Manager and using wsadmin scripts or the administration System Console to set up the two cells and 4 instances in each.

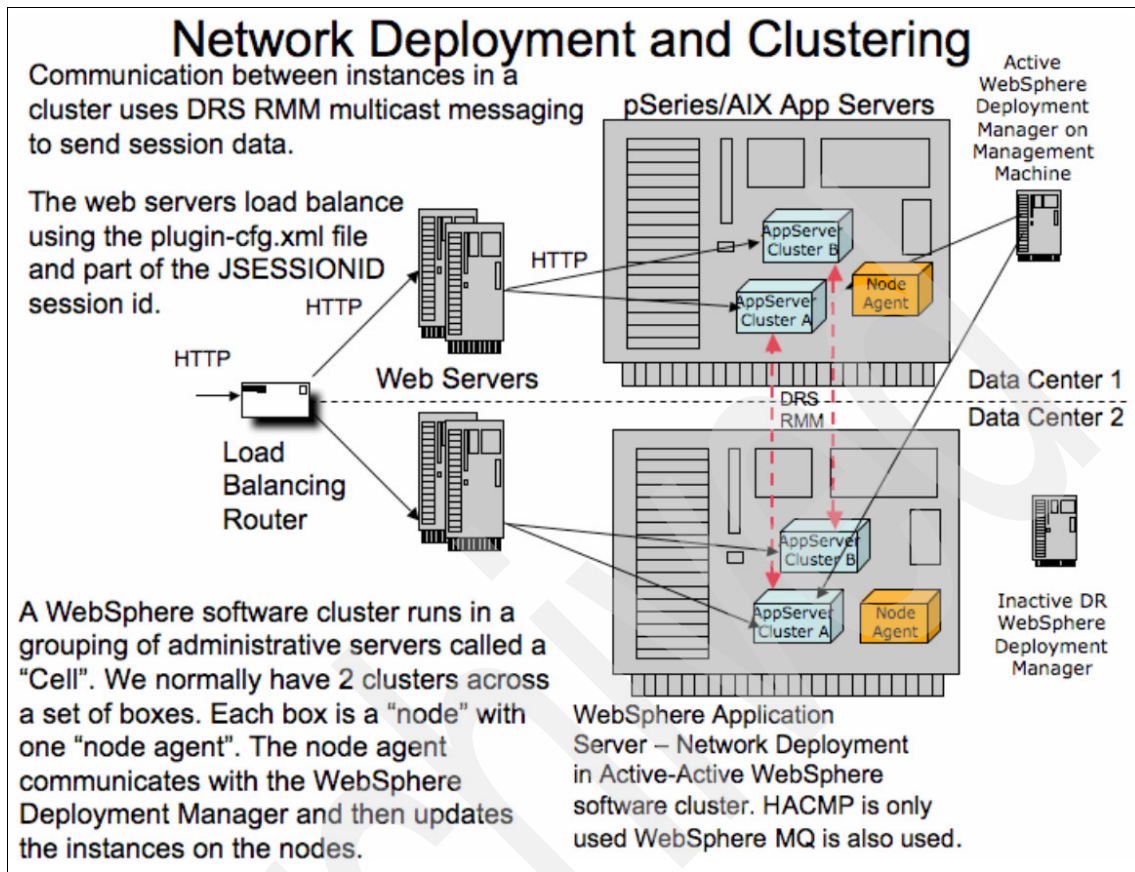


Figure 5-19 Network deployment and clustering

The Deployment Manager holds all of the configuration information for the cell and cluster in XML files as outlined previously. It synchronizes the configuration of the cell and clusters with each node agent, of which there is one node agent (usually) per operating system image unless multiple cells are to be configured. In this case, assume one cell and one node agent per operating system image/partition for your configuration.

The instances are created and the node agent gives each instance its configuration, but the master configuration for the whole cell and all of the clusters it contains is in the Deployment Manager, which should be backed up accordingly. When the application is deployed, it is rolled out to each of the instances, and your middle tier configuration is complete.

5.5.2 IBM HTTP Server and the IBM WebSphere Application Server plug-in

A useful standard Web server tier could be supported by the highly performant and scalable IBM Blade Center environment, with up to 14 IBM Blade Center Servers per blade center chassis. As we are considering AIX, assume the IBM JS-21 BladeCenter Servers are in use with AIX 5.3 or AIX 6.1 as their operating system. With these machines, partitioning is now supported by a HyperVisor, and for AIX 6.1, Workload Partitions are also supported. Thus, multiple layers of isolation and protection for the Web servers is possible for the purposes of security.

In many ways, the ideal configuration is to install IBM HTTP Server (IHS) 6.1 in the global environment, but not make that accessible to the outside world. Then then create multiple WPARS that act as hosts to the outside world to give a very secure environment that can be monitored and controlled from the global environment. For our purposes, assume that AIX 5.3 is being used.

When the IBM HTTP Servers are installed, the Deployment Manager can be used to push the configuration out. A plug-in called `mod_was_ap20_http.so` is installed and configured for loading by the Web server. The configuration of routing requests to the application server clusters is handled by entries in a file usually called `plugin-cfg.xml`. The plug-in is loaded inside IHS by configuring the IHS `httpd.conf` file, as shown in Example 5-67.

Example 5-67 IHS httpd.conf: routing request configuration

```
LoadModule
    was_ap20_module /usr/IBMIHS/Plugins/bin/mod_was_ap20_http.so

WebSpherePluginConfig
    /usr/IBMIHS/Plugins/config/webserver1/plugin-cfg.xml
```

The `plugin-cfg.xml` file is generated using the administration console from the Deployment Manager, or by using the `GenPluginCfg.sh` script in the directory `/usr/IBM/WebSphere/AppServer/bin`. This generates a file with weightings and load balancing algorithms including backoff and retry (that is, Nagle) to ensure that the cluster instances are used appropriately and that resilience is available. As is often the case, manual editing of this file to simplify the configuration can be helpful.

Example 5-68 plugin-cfg.xml example

```
<?xml version="1.0"?>
<Config ASDisableNagle="false" IISDisableNagle="false"
    IgnoreDNSFailures="false" RefreshInterval="60"
```



```

        </Transport>
    </Server>
    <Server Name="server2_lpar2" ConnectTimeout="0" ExtendedHandshake="false"
        LoadBalanceWeight="1" MaxConnections="-1" WaitForContinue="false">
        <Transport Hostname="aix_lpar2" Port="9081" Protocol="http"/>
        <Transport Hostname="aix_lpar2" Port="9444" Protocol="https">
            <Property name="keyring"
value="/usr/IBMIHS/plugins/etc/plugin-key.kdb"/>
            <Property name="stashfile"
value="/usr/IBMIHS/plugins/etc/plugin-key.sth"/>
        </Transport>
    </Server>
    <PrimaryServers>
        <Server Name="server1_lpar1"/>
        <Server Name="server1_lpar2"/>
        <Server Name="server2_lpar1"/>
        <Server Name="server2_lpar2"/>
    </PrimaryServers>
</ServerCluster>
<UriGroup Name="server1_Cluster_URIs">
    <Uri Name="/TestManagerWeb/*" />
    <Uri Name="/TestManagerWeb" />
    <Uri Name="/TMAdmin/*" />
    <Uri Name="/TMAdmin" />
    <Uri Name="*.do" />
    <Uri Name="*.action" />
    <Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid"
Name="/TestManagerWeb/*" />
    <Uri Name="/snoop/*"/>
    <Uri Name="/snoop"/>
    <Uri Name="/hello"/>
    <Uri Name="/hitcount"/>
    <Uri Name="*.jsp"/>
    <Uri Name="*.jsv"/>
    <Uri Name="*.jsw"/>
    <Uri Name="/j_security_check"/>
    <Uri Name="/ibm_security_logout"/>
    <Uri Name="/servlet/*"/>
    <Uri Name="/ivt"/>
    <Uri Name="/ivt/*"/>
    <Uri Name="/_DynaCacheEsi"/>
    <Uri Name="/_DynaCacheEsi/*"/>
    <Uri Name="/wasPerfTool"/>
    <Uri Name="/wasPerfTool/*"/>
    <Uri Name="/wasPerfToolservlet"/>

```



```

    <Uri Name="/wasPerfToolServlet/*"/>
  </UriGroup>
  <Route ServerCluster="server1_Cluster"
    UriGroup="server1_Cluster_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 plugin-cfg.xml file shown in Example 5-68 on page 282 will run in a round robin manner among the application server instances server1_lpar1 and server1_lpar2 in LPARs 1 and 2, the server instances server2_lpar2 and server2_lpar2 for URLs /TestManagerWeb and /TMAdmin, and anything with the extensions .do and .action.

IHS is a derivative of the Apache 2 Web server, but it has an IBM internal security engine with appropriate licenses for some subsystems that Open Source developments cannot obtain. It also has support for Fast Response Cache Architecture (FRCA) to accelerate responses by caching commonly used static data lower down the TCP/IP stack using a kernel extension.

Essentially, the static content cached is kept on the kernel side of the TCP/IP stack, which avoids the expensive copying of large amounts of data between kernel and user space. This improves performance, and is in addition to the content itself being cached rather than reread from disk buffers.

The FRCA kernel extension is loaded using **frctr1 load**, usually as part of the AIX startup initiated in /etc/inittab. It is used by loading the mod_afpa_cache.so shared object by adding the entries shown in Example 5-69 to the end of the httpd.conf file.

Example 5-69 FRCA startup modifications to httpd.conf

```

AfpaEnable On
AfpaCache On
AfpaLogFile "/usr/IBMIHS/logs/afpalog" V-ECLF
AfpaPluginHost aix_lpar1:9080
AfpaPluginHost aix_lpar1:9081

```

```
AfpaPluginHost aix_lpar1:9443
AfpaPluginHost aix_lpar1:9444
AfpaPluginHost aix_lpar2:9080
AfpaPluginHost aix_lpar2:9081
AfpaPluginHost aix_lpar2:9443
AfpaPluginHost aix_lpar2:9444
```

You must also remember to load the FRCA module by adding the appropriate entries to the LoadModules section of httpd.conf, as shown in Example 5-70.

Example 5-70 httpd.conf: loading FRCA modules

```
#
# Dynamic Shared Object (DSO) Support
#
# To be able to use the functionality of a module which was built as a DSO you
# have to place corresponding 'LoadModule' lines at this location so the
# directives contained in it are actually available before they are used.
# Statically compiled modules (those listed by 'httpd -l') do not need
# to be loaded here.
#
# Example:
# LoadModule foo_module modules/mod_foo.so
...
LoadModule deflate_module modules/mod_deflate.so
LoadModule ibm_afpa_module modules/mod_afpa_cache.so
```

In Example 5-70, note that the mod_deflate.so shared object is also loaded to enable compressed HTTP pages. This deflate compression is supported by most browsers and response times can be greatly improved, so this should be used.

Now when a request comes into the Web channel, as shown in Figure 5-20 on page 287, it will be picked up by the load balancing router (something like a Cisco LocalDirector) or IPSprayer and sent to one of the Web servers in one of the data centers, depending on loading and which received the last request.

This will be handled by one of the JS/21 blades IP stack. And, if it is for a static page that it has cached in the Network Buffer Cache/Fast Response Cache Architecture cache, it will return the response immediately from the cache. This is likely to be the case for commonly used static content.

The request is then passed to the Apache2-based IBM HTTP Server process listener for handling. If the request is for content that it can serve, or that it knows is in error, it will either handle it itself or pass it on to another Web server process to handle.

Otherwise, if the request is for a URL configured in the plugin-cfg.xml file, the mod_was_ap2_http.so module comes into play. This compares the request URL to the list configured in plugin-cfg.xml file. The plug-in will check authorizations and look at the HTTP cookies or URL query string to get access to the JSESSIONID value.

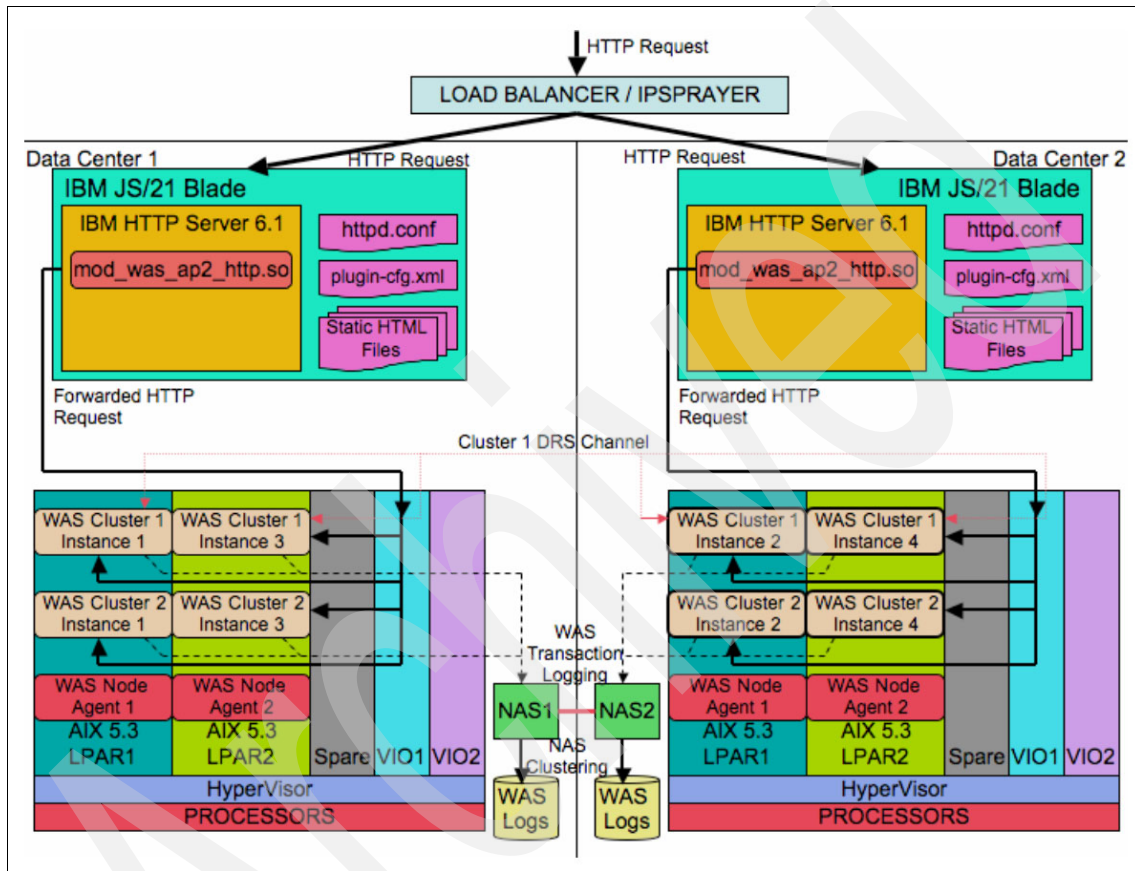


Figure 5-20 Load balanced clustered data center

J2EE specifies that when a Web application sets up a session, usually after a user has logged on, it should return a session cookie or a value in the query string of JSESSIONID that should be a pseudorandom value used to identify the session. The WebSphere Application Server clustering mechanism modifies this value in order to append a clone or instance ID to it, which it can then use in the plug-in to identify the server and cluster that initiated the session.

It then forwards the request as an HTTP or HTTPS request to the WebSphere Application Server instance identified by the instance ID. If that instance has

failed or does not respond within a timeout value, it will choose another instance in that configured cluster, using the information in the plugin-cfg.xml file to route the request to.

This all assumes that the Data Replication Service (DRS) has been configured to replicate all of the session information between the nodes in the cluster using the Reliable Multicast Messaging protocol, or that the session information has been stored in a session database accessible by all nodes in the cluster. Thus, the cluster nodes can take over the session of a failed node. If the user has yet to login and set up a session, then any server can handle the request.

To understand how the plug-in works, the book *Apache: The Definitive Guide* is a useful reference. Essentially, an Apache module is given an opportunity to load its configuration into each instance of the Web server and use the `ap_hook_handler` call to register a handler that gets to peek at requests as the HTTP server examines them; see Figure 5-21 on page 289.

The use of the “strings” call against the `mod_was_as20_http.so` binary shows that the WebSphere plug-in functions in the same way, and suggests that the `websphereHandleRequest` and `websphereExecute` functions handle the reading of requests from IHS and the forwarding to the appropriate WebSphere Application Server cluster instance. To develop a better understanding of this subject, you can download the TomcatConnectors `mod_jk.c` file from the following address and examine the use of the Apache 2 APIs:

<http://www.apache.org>

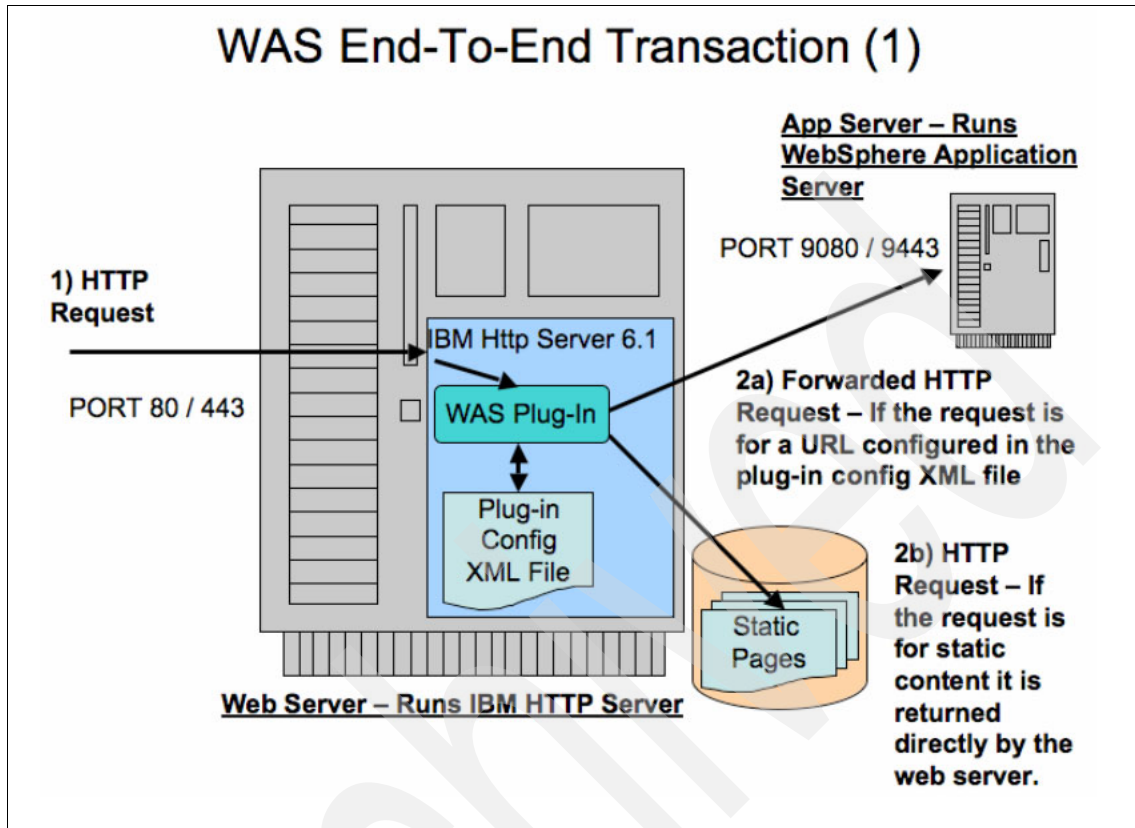


Figure 5-21 WebSphere Application Server end-to-end transaction path

5.5.3 WebSphere Application Server clustering

To improve performance in the Web servers for cached content, we used FRCA. The same FRCA feature can be used by the Web containers on the application server tier, but the benefits are less. The application server tier primarily serves dynamic content from JSPs and servlets, and this dynamic caching is supported with an FRCA implementation on the Windows platform but not on AIX. As before, the FRCA kernel extension is loaded using `fractrl load`, usually as part of the AIX startup initiated in `/etc/inittab`.

WebSphere Application Server has its own Web caching mechanism for dynamic content called *dynacache*. This is called the “Dynamic Cache Service” in the WebSphere Deployment Manager System Console. It is enabled at the Application Server Container level for each application server instance. It is also worth enabling Servlet Caching for the Web Container for the application server instance.

What is cached by the Dynamic Cache Service is controlled by the `cachespec.xml` file that is created for each application module and put in the `WEB-INF` and `META-INF` directories for the given application module. The contents of the cache can overflow to disk if configured at the application server container level by enabling disk offload for the Dynamic Cache Service, but a different directory should be used for each application server instance in a partition to avoid conflicts.

Control over what is cached is handled by `<cache-entry>` entries in the `cachespec.xml` file. Because this can consist of WebSphere commands, Web services output and servlet output that are mostly application-level content, the configuration and use of this is normally something configured by the application team along with the WebSphere administrators, and is not a task for the AIX administrator.

Edge Side Include Caching is another option for caching if the infrastructure is made slightly more complex. If this is the case, then the `DynacacheESI` application must be installed inside the application server. Refer to WebSphere Application Server documentation for more information about this subject.

When a request is received by the application server instance, it is first examined by the Web container using an inbound channel that is routed to the appropriate application module using the configured URL mappings for cluster (see Figure 5-21 on page 289). The request would have come into the given application server instance using the instance (formerly clone) ID that was appended to the `JSESSIONID` cookie value or query string when the session was created.

Before the Web module gets the request, a servlet filter (which is Java code sitting outside the application) is given access to the chain of the HTTP request to allow it to modify the request, and it gets access to the response that is generated further down the chain. While the request is being handled, a thread is tied up in the container, and any objects that are attached to the Java session object contain the state that represents the session for which that request is a part.

To enable that state to be retained when an instance fails, the session can be configured for storage in a session state database, as identified previously in `server.xml`, or it can be replicated entirely by a fast reliable multicast messaging (RMM) protocol to other instances in the cluster, something known as the Data Replication Service or DRS.

The DRS Session Buffer is shared with the HAManager Distribution and Consistency Services, so it should be made larger for high loads. The use of an active-active session database rather than replicating large session objects

using DRS is preferred unless the use of only small session objects by applications can be guaranteed.

Performance is likely to be higher for DRS at the cost of buffer usage, but the session database does reduce intracluster communications overheads.

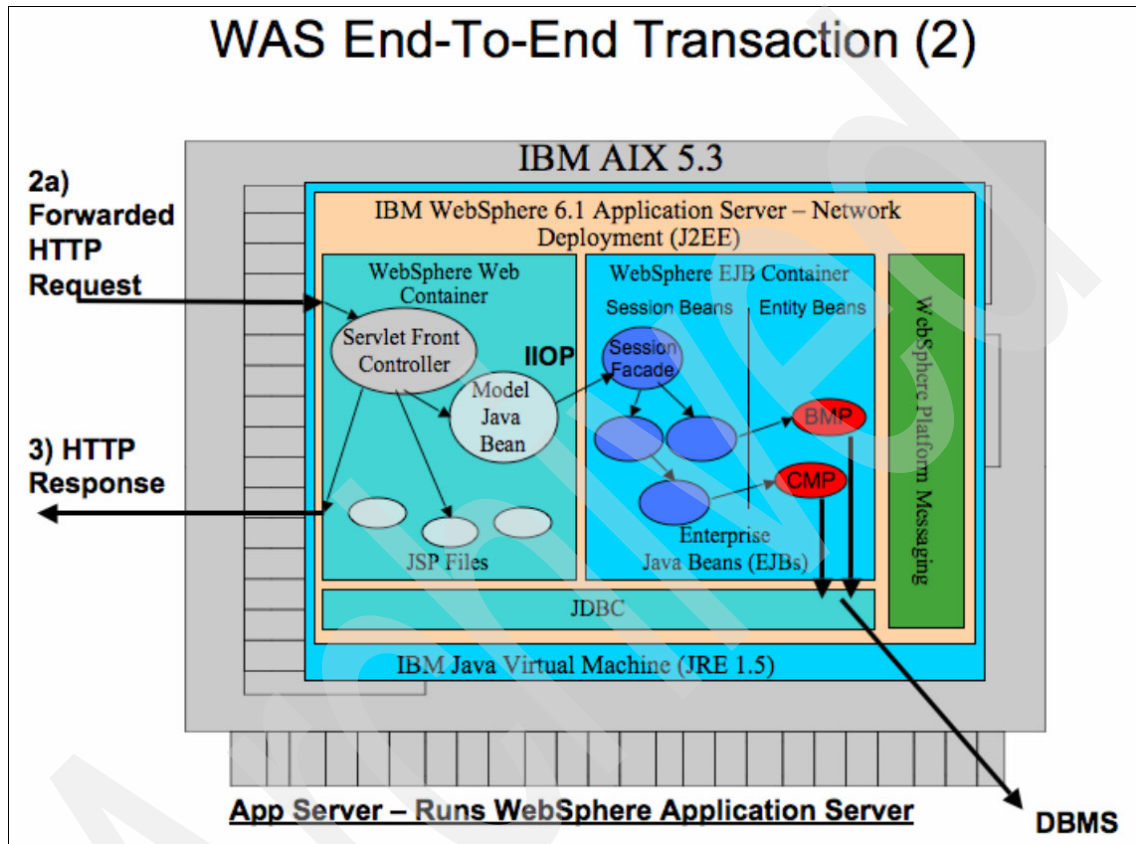


Figure 5-22 WebSphere Application Server end-to-end transaction path, part 2

The Web request is eventually passed on, in an application that fits a standard architecture, to a set of Enterprise Java Beans hosted in the EJB container. Usually a session façade is implemented as a session bean to “front” the application logic and entity beans or some other persistence mechanism.

If the application server environment is configured simply to run within the same Java virtual machine, then the request is merely passed between the two containers as a CORBA/IIOP request. However, if an extra tier has been introduced and the Web container and EJB container are configured on different partitions, then the EJB request is added to a queue of CORBA/IIOP requests.

The application server is designed to handle distributed transactions as an XAResource Coordinator with the use of the Java Transaction Service and the use of transactions within EJBs. Many things will result in an XA distributed transaction, such as EJBs with different configured transaction isolation levels and requirement, or a the use of a transactional write to a database with an entity bean followed by a JMS/MQ message controlled within the same transaction.

To support this, the WebSphere Application Server uses transaction logs for commit and rollback, where transactions are only committed to the real environments rather than the logs after a two-phase commit. These logs are in the tranlog directory and there is one per server instance that is managed by the Transaction Manager.

In our diagrams we have shown the use of a NAS device, but have yet to explain why. If the WebSphere Application Server instances in a cluster all place their logs on the same shared NAS device using NFSv4 where the other instances can see them, then after a failure, another node can apply those transaction log entries to ensure consistency. This is handled by the Transaction Manager and it is the job of the HAManager and Distributed Consistency Services (DCS) (usually using ports 9352 and 9353) to manage the consequences of the failover.

The clusters use the concept of a core group to handle the groupings of instances for HAManager and failover, usually with the JVM ID with the lowest ID acting as the coordinator. The default is for the first process in the core group ordered lexically by name to act as coordinator. This should, however, generally be explicitly configured to prefer a process with low load.

To ensure that the transaction logs are always available for recovery, the availability of the NAS device is now important. NFSv4 is required because this version of the NFS standard forces locks to be removed on timeout, which is required for recovery. The IBM N5200 and N5500 NAS devices, derived from the NetApp® FAS3XXX range, support this with the OnTap operating system and metro-cluster clustering. Versioning of changes is performed on these devices when a write occurs to allow rollback and online backup.

Note that a third device is placed in a third data center to allow quorum to be maintained if there is a loss of communications between the two main data centers, to avoid split-brain scenarios and having both sites update shared data independently, resulting in inconsistency. However, this would only be used if disk cross-connectivity between sites *and* network connectivity fails.

5.5.4 WebSphere HAManager

In our previous examples, the HAManager has been configured as default with a single DefaultCoreGroup. This is the scenario where the lowest number JVM ID

will act as coordinator to manage recovery of in-flight transactions and take over management of coordination of the core group membership (that is, the views). More complex configurations can be configured, as explained in WebSphere Application Server documentation.

HAManager is in many ways like a Java version of HACMP, in that it maintains its own heartbeats between cluster instances. One of its main roles, beyond the interface to the WebSphere Application Server Transaction Manager, is supporting failover of other Java singletons. For example, message ordering should be retained for Java messages so that a single instance should, unless explicitly configured otherwise for load balancing, exist in a cluster or more exactly, a core group. Startup of JVMs in the core group require the node agent to be available to ensure quorum; otherwise, some services will wait.

So, by default the messaging engine will exist as active on one member in a cluster, with persistent messages persisted to a JDBC compliant database such as DB2 or Oracle, or a shared filesystem, and failover is handled by HAManager to bring up the messaging engine on another node. This is similar to the behavior that would be seen if HACMP was used with WebSphere MQ Server, but it is all controlled within the JVM with Distribution and Consistency Services managing the intracore group communications. This all occurs at the node level, rather than at the instance level.

More detailed information about HAManager is available at the following address:

http://www.ibm.com/developerworks/websphere/techjournal/0509_lee/0509_lee.html

Clustering - HAManager for WAS and Messaging Engine

- Nodes all log to a shared NAS (Network Attached Storage) with NFSv4
- Only one JMS Messaging Engine for cluster, but each node has its own Transaction Manager log manager managing the transaction logs for that node.
- HAManager fails over the management of the transaction log for the failed node so another node in the cluster can process it in addition to its own logs.
- NAS allows all nodes to see all logs, but only one node can write to each log
- NFSv4 is required so all locks are released on timeout after a node failure.

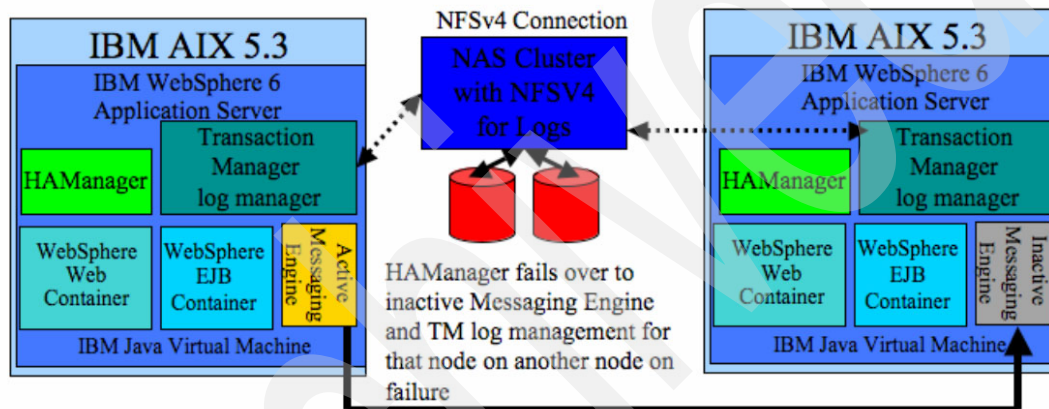


Figure 5-23 HAManager

You may be wondering how HAManager works. It has a number of key components, but you need to understand a few facts about its intent and configuration before delving into its components.

Every instance in a WebSphere Application Server - Network Deployment environment has an HAManager. The HAManager is responsible for managing the singletons primarily, that is, the Transaction Manager of which there is one per instance and the Messaging Engine of which there is one per cluster (to maintain message ordering).

The management of the Transaction Manager requires notifying another instance to take over the transaction management logs, used for XA resource coordination, so they can be processed and apply or roll back any in-doubt transactions.

The management of the Messaging Engine requires another node to take over processing the messages and any persistent storage or database that is associated with it.

To do this all of this, HAManager uses the concept of a core group, of which there is at least one per cell (the DefaultCoreGroup). It is possible to have more than one per cell, and even to bridge between them using the CoreGroupBridge (for example, to cross firewalls isolating cell members). Management of the group membership is handled by the Group Manager component inside HAManager.

Communications between group members must be fast to allow synchronization of data, and for this the Distribution and Consistency Services (DCS) are used. They make use of Reliable Multicast Messaging (RMM) transport and unicast communications. Two inbound DCS TCP transport chains are used; DCS and DCS-Secure, the latter of which supports SSL, and both of which terminate at the DCS_UNICAST_ADDRESS which can be found in serverindex.xml.

This is all part of the DCS services, and essentially consists of control of messages for communication of state updates from group members and heartbeat information, and is more heavily used when memory-to-memory replication of state is enabled.

To receive and hold state information from other members of the core group, a bulletin board function is required, to which messages are sent and interest parties subscribe. This is all handled by the Bulletin Board component inside HAManager.

Overall control of running HAManager is handled by the Agent component. The Agent component maintains a picture of the core group and its state, so it can take remedial action in the event of a failure.

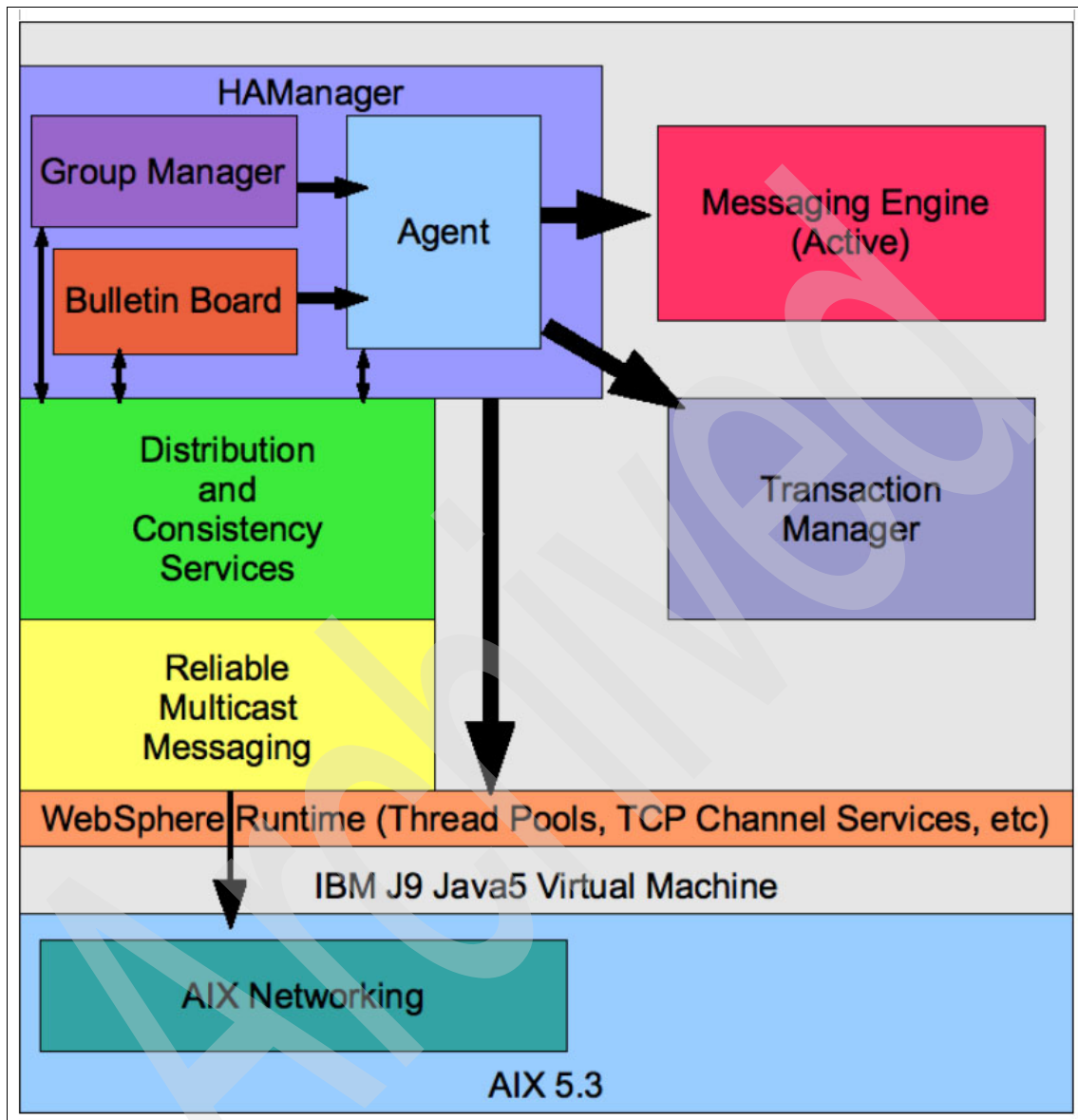


Figure 5-24 HAManager runtime control

When HAManager initializes in a JVM process, it tries to contact the other members of the core group. When it finds another JVM process in the core group, it starts a join process to join that core group. If it is accepted, it becomes one of the instances that receives messages to its bulletin board from other group members, and subscribes to their bulletin board services.

All HAManager instances will then log a message to say the new node has joined. This starts the View Synchrony protocol to ensure that all group members have a synchronized view of the state of the other members, and a similar View Synchrony action is undertaken when an instance stops or its connection drops. Both of these are cases of a view change. In large groups, the HAManager may start to consume significant resources and generate significant network traffic, so breaking the group into smaller groups is advised.

On failure, the JVM with the lowest ID (usually the Deployment Manager, followed by the Node Agent) will take over as coordinator for the group. This behavior can be configured. Overall behavior of the HAManager core group is configured as a policy that is controlled by the Group Manager and applied by the Agent.

So, how should this be managed on AIX? Beware of port conflicts on the DCS_UNICAST_ADDRESS, because it can prevent the instance from starting. Therefore, ensure the nodes are all synchronized from the Deployment Manager if anything changes.

Also be aware that there is a link between the AIX tcp_sendspace, tcp_recvspace, and sb_max settings and the behavior of the DCS unicast and multicast messaging, and a custom setting inside WebSphere Application Server called IBM_CS_SOCKET_BUFFER size may be required for tuning.

The IBM_CS_FD_PERIOD_SECS can be used to change the heartbeat period to additionally change the loading. There are differences in the underlying protocols used between WebSphere Application Server 6.0.0, 6.0.9, and 6.1, so ensure that the appropriate version is configured.

Performance can also be improved by enabling IP address-to-name caching, so set the IBM_CS_IP_REFRESH_MINUTES setting to something appropriate for the given environment. The heap memory used by the replication process can be significant, so tuning for the appropriate size using the IBM_CS_DATASTACK_MEG setting is recommended.

For some WebSphere Application Server versions (6.0.X), the default thread pool was used for HAManager, and overloading caused by errant applications could affect HAManager and its response to the requests on the TCP inbound chain, so configuration to use its own thread pool is recommended.

In general, if the messaging engine is not used and the transactional integrity XA resource coordination is not required, then HAManager can be disabled to increase overall performance of WebSphere Application Server. In general, however, less than 20 core group members should not present significant impact.

5.5.5 WebSphere Application Server, WebSphere MQ Server and HACMP

Although the WebSphere Application Server - Network Deployment environment fully supports messaging from within the JVM instance, the use of a full WebSphere MQ Server 6 configuration is still supported and may be preferable for some environments. There are many reasons why this choice might be made, and you can refer to the WebSphere documentation to obtain further details.

However, in this environment HAManager cannot be used because there are multiple WebSphere MQ processes that run outside of the JVM process at the operating system image level. In this situation, HACMP must be used to fail over the active WebSphere MQ Server instance to a standby copy, probably using PPRC to mirror any data and remount any storage.

HACMP can also be used to failover the WebSphere Application Server instance itself for some environments, such as for the Node Agent and Deployment Manager that manage the clusters.

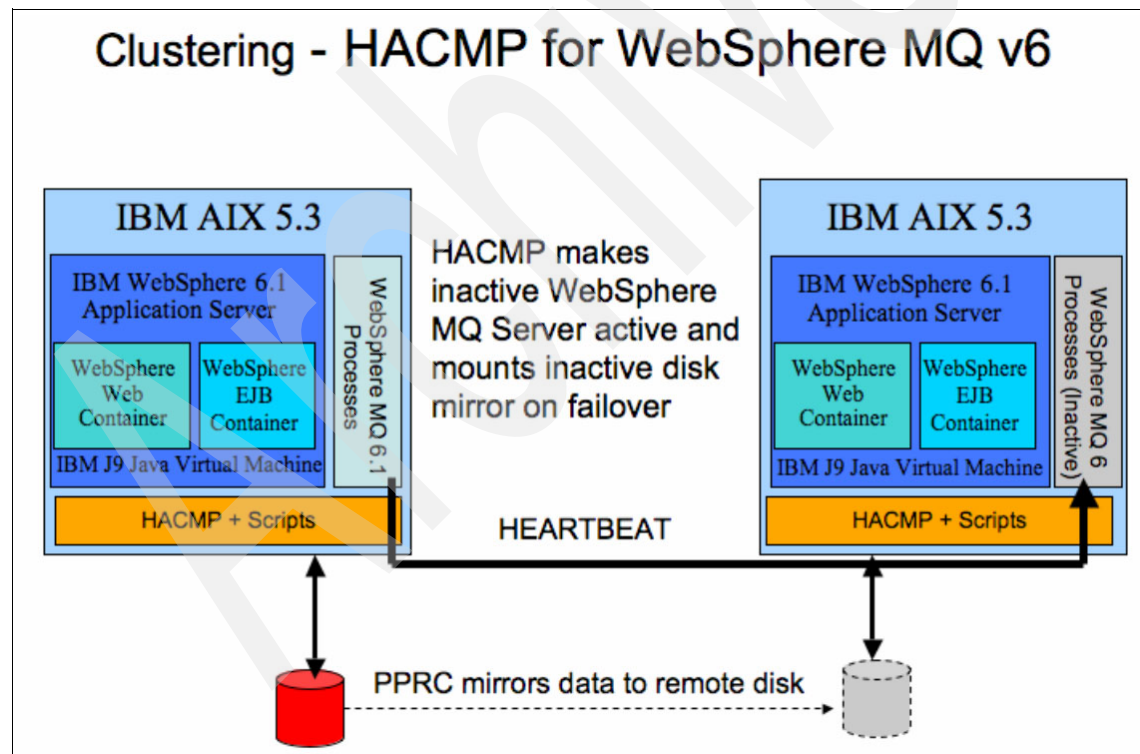


Figure 5-25 Clustering - HACMP for MQ 6

5.5.6 The WebSphere Application Server database tier

Although it is outside of WebSphere Application Server, the database tier affects the behavior of WebSphere Application Server from an active-active performance and failover perspective, as discussed here.

If you have an active-passive configuration, then the data center with the active database has a shorter network path length so it responds quicker, which skews load balancing. In this case, a DB2 on AIX HADR configuration for the session database, application database, and persistent messaging database is not ideal.

Oracle supports clustering in a shared everything database environment with the cache fusion fast interconnect with Real Application Clustering (RAC) and Oracle 10GR2, which has been proven to fully support a fully active-active and shared load environment using the NAS configuration to avoid split-brain scenarios. It is believed a similar active-active and fully load balanced configuration can be achieved using the Gridscale product for DB2, although this was not tested for this document.

5.5.7 WebSphere Application Server ND versus WebSphere Application Server XD on System p

Although the core of an enterprise level OLTP environment is produced by WebSphere Application Server ND, there are benefits to using WebSphere Application Server XD and its intelligent router feature.

WebSphere Application Server XD can identify a requirement for more resources to support a configured application load according to a set of predefined rules. WebSphere Application Server XD will try to find an environment that can host a newly created WebSphere Application Server instance to help support the load, which may require shutting down lower priority applications or instances.

Note that there is a difference in the paradigms underlying these products. WebSphere Application Server XD aims to move the load to the available resources. In contrast, the System p HyperVisor aims to move the available resources to the load, so if a partition is overloaded running a WebSphere Application Server instance and another partition has less load and resources available within the same physical machine, the HyperVisor can attempt to take physical CPU cycles away from the given partition and get it to cede those cycles to the WebSphere Application Server instance if capping does not come into play.

In the meantime, WebSphere Application Server XD may be trying to create another WebSphere Application Server instance, possibly in the less-utilized partition, to reduce the load.

Both approaches are valid, but it is important to configure rules and thresholds to ensure that conflicts do not occur. Future WebSphere Application Server XD implementations may make use of more HyperVisor statistics using the appropriate APIs; see Figure 5-26.

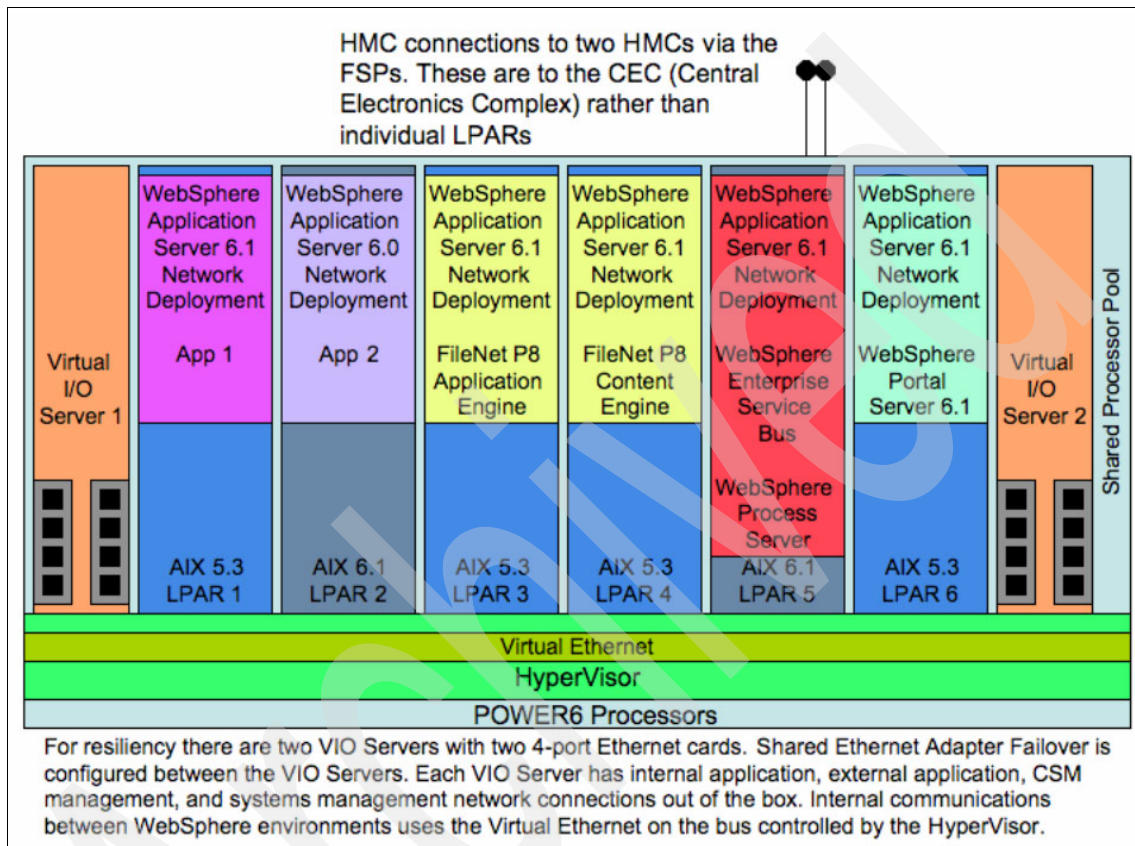


Figure 5-26 Future WebSphere Application Server platform configuration scenario

WebSphere Application Server 6.1 uses an Eclipse 3.1.2 platform as its base to provide the OSGI platform functionality for dynamic component management.

It is likely that WebSphere Application Server 7 will use Eclipse 3.3 as its base, or possibly Eclipse 3.4, and continue the move to a more modular component-based runtime where components that provide WebSphere Application Server features are loaded dynamically when required.

Eclipse 3.3 has its own HTTP listener and JMX implementation, so it is likely that WebSphere Application Server will move to using these rather than its own implementation and the open source tmx4j implementation, respectively.

To summarize, the key to an understanding the direction of WebSphere Application Server and many IBM products is to understand the OSGI architecture and how it improves dynamic extensibility and reduces memory consumption by only loading required components.

Archived

Tuning the IBM Java Virtual Machine

This chapter describes the use of tuning parameters that are available on the IBM Java Virtual Machine (JVM) on AIX. Because WebSphere Application Server as well as the applications deployed on it execute within the JVM, tuning the Java runtime is a crucial step in optimizing the performance of your applications.

Because each application is unique and may be well-designed, poorly designed, or something in between, achieving performance improvements from following the ideas and recommendations presented in this chapter cannot be guaranteed. However, the chapter introduces the concept of load testing and discusses the use of load testing tools.

6.1 The importance of JVM tuning

The inclusion of the IBM Java 5 runtime environment into WebSphere Application Server V6.1 has introduced many improvements intended to enhance Java's performance on AIX. These enhancements include an entirely new runtime environment with new garbage collection algorithms; an improved Just-in-Time (JIT) compiler; the availability of class sharing among running JVMs; and better utilization of the AIX advanced performance and virtualization features. These enhancements produce greatly improved runtime performance, a significant decrease in the memory footprint of the JVM, and the availability of new options for *tuning* the JVM for specific purposes.

It is surprisingly common for an application to be deployed on WebSphere Application Server with incorrect runtime parameters set. There are many reasons for this:

- ▶ Lack of communication between application designers and server administrators
- ▶ Lack of understanding of runtime parameters
- ▶ Lack of understanding of the importance of proper runtime parameter settings
- ▶ Considering runtime options as an afterthought
- ▶ Lack of load testing prior to deployment

Deploying an application with incorrect settings can lead to confusion and frustration when serious errors “mysteriously” appear. These errors are often fatal and may include out-of-memory errors, crashes, hangs, and so on. This scenario can result in frustration, system downtime, and lost time spent in problem determination and in unnecessary calls to technical support.

To achieve optimal application performance and to avoid being surprised by serious errors, the Java runtime environment™ must be tuned according to the specific application's needs. The material in this chapter instructs and guides administrators in optimizing the Java runtime for WebSphere Application Server V6.1. For instructions about tuning AIX and utilizing AIX's advanced performance and virtualization features, see Chapter 4, “AIX configuration” on page 127.

6.1.1 Overview of JVM tuning capabilities

The following JVM tuning capabilities are available with the JVM that is included in WebSphere Application Server V6.1 for AIX.

- ▶ Choosing a garbage collection policy
- ▶ Tuning garbage collection
- ▶ Heap sizing
- ▶ Using large page sizes

- ▶ Using shared classes
- ▶ Micropartitioning and DLPAR considerations
- ▶ JIT tuning options

Important: This list is presented in a roughly descending order of impact. The two most important items that help with performance are choosing the correct garbage collection policy and heap sizing.

At a minimum, garbage collection and heap sizing should be tuned properly. When they are set properly, you will have contributed significantly to optimizing performance of your application (assuming that the application is properly designed and coded in the first place). If an application is poorly designed, or if poor Java coding practices are followed, then tuning may be of little help in optimizing performance.

Although choosing the correct garbage collection policy and heap sizing are very important factors in tuning the IBM JVM, significant improvements can be achieved by utilizing other tuning factors. For instance, a small percentage gain in application runtime performance can be achieved through leveraging the System p large page capability. And if multiple JVMs are run on the same server, then certain performance improvements can be achieved through the use of a shared class cache.

6.2 Choosing a garbage collection policy

This section focuses on choosing garbage collection (GC) policies and describing how to approach tuning the IBM JVM garbage collector especially for WebSphere V6.1 applications. There are four garbage collection policies available in the Java 5 IBM JVM. Choosing the correct one will have major impact on improving application performance.

The four GC policies available in the IBM JVM are:

- ▶ Optimal Throughput
- ▶ Optimal Average Pause
- ▶ Generational Concurrent
- ▶ Subpool

The default GC policy is Optimal Throughput. For WebSphere Application Server applications, however, this is most likely not the best choice. Because throughput is the goal for this policy, longer pauses during garbage collection can be expected. The two policies better suited for WebSphere Application Server applications are Generational Concurrent and Subpool. Generational Concurrent

was designed with the typical characteristics of WebSphere applications in mind, and Subpool takes advantage of AIX and System p hardware features.

The four GC policies are explained in greater detail in other sections of this chapter. Some policies require additional manual tuning. Because the tuning parameters require knowledge of GC internals, a brief explanation of Java memory management is provided next.

6.2.1 Java memory management: garbage collection and allocation

Memory management, in its simplest terms, is nothing more than allocating and freeing memory. But of course it is much more than that. How allocation occurs and when and how collection of free objects occur can matter significantly when it comes to runtime performance.

The memory management component of the Java runtime environment is composed of the garbage collector and the allocator. The allocator allocates space for objects in a contiguous section of memory called the heap. The garbage collector frees up space in the heap when an object is no longer needed.

Heap

The heap is a contiguous section of virtual memory where all the Java objects in a running JVM are stored. The maximum size of the Java heap is preallocated when the JVM starts. There is another memory area that Java uses called the system (or native) heap where allocations from application JNI code, compiled JIT code, and threads to map to Java threads, among other things are stored. The Java heap is commonly referred to as “the heap.”

Allocator

All running threads have a local allocation buffer (cache). The cache block for all threads is sometimes called the thread local heap (TLH). Objects are allocated from the TLH if possible; if this is not possible, then a locked heap allocation is performed. If many allocations occur at once, then heap lock can become an issue. GC policies try to avoid heap lock through unique allocation algorithms.

Garbage collector

The garbage collector keeps track of which objects are still referenced (and thus, still needed) and removes all other objects, which are considered “garbage”. Garbage collection only occurs when an allocation failure (AF) occurs due to lack of room in the heap, or possibly when a call to `System.gc()` occurs.

When an allocation failure occurs, the running thread that had the failure takes exclusive control of the VM. All application threads stop and garbage collection commences. This is referred to as a stop-the-world (STW) event.

The process of garbage collection consists of three phases:

- | | |
|-------------------------|---|
| Mark phase | This is when all reachable objects are identified and all else is considered garbage. |
| Sweep phase | This is when space used by all unmarked objects is reclaimed. |
| Compaction phase | This is when objects are moved into as contiguous a space as possible and references are updated. |

The compaction phase is the most costly because objects are moved and therefore require all references to them to be updated as well. Thus, compaction is always avoided if possible.

The four GC policies available to the IBM JVM each take different approaches to the handling of memory management and have been given descriptive names to help identify the purpose for each.

Additional references for the Java 5 garbage collector

For a more complete description of IBM Java memory management, including the heap and garbage collection, refer to Chapter 2 of *IBM Java 5 Diagnostics Guide*.

For additional information about the four GC policies available in the IBM JVM, refer to the following developerWorks article “Java Technology; IBM Style: Garbage collection policies, Part 1”

<http://www-128.ibm.com/developerworks/java/library/j-ibmjv2/index.html>

For further discussion about how to choose a GC policy, refer to the article “Java Technology; IBM Style; Garbage Collection policies, Part 2”. This article focuses on Java applications in general and is not specific to WebSphere Application Server.

<http://www-128.ibm.com/developerworks/java/library/j-ibmjv3/index.html>

6.2.2 Optimal Throughput GC policy

Optimal Throughput is a single generation Mark Sweep Compact (MSC) Stop-The-World collector. It is the default GC policy and is designed with speed in mind, so lengthy pauses during GC are to be expected.

Use Optimal Throughput when you want an application to run to completion as quickly as possible. This goal is usually at odds with typical WebSphere applications, but may be appropriate for very short-running applications.

Because Optimal Throughput is the default GC policy, no runtime option is required, although it can be set explicitly by entering `-Xgcpolicy:optthruput` in the Generic JVM arguments field in the Java Virtual Machine Process Definition page in the WebSphere Application Server administration console.

To open the Java Virtual Machine settings page:

1. Open the WebSphere integrated solutions console.
2. In the navigation tree, expand **Servers** and click **Application servers**.
3. Select the desired application server.
4. Under the Server Infrastructure heading, expand **Java and Process Management** and click **Process Definition**.
5. Under Additional Properties, select **Java Virtual Machine**.

Figure 6-1 on page 309 shows the console page used to access these settings.

Application servers

[Application servers](#) > [server1](#) > [Process Definition](#) > [Java Virtual Machine](#)

Use this page to configure advanced Java(TM) virtual machine settings.

Configuration **Runtime**

General Properties **Additional Properties**

Classpath

Boot Classpath

☐ Verbose class loading

☐ Verbose garbage collection

☐ Verbose JNI

Initial Heap Size
1024

Maximum Heap Size
1024

☐ Run HProf

HProf Arguments

☐ Debug Mode

Debug arguments
-Djava.compiler=NONE -Xdeb

Generic JVM arguments
-Xgcpolicy:optthruput

Executable JAR file name

☐ Disable JIT

Operating system name
aix

Apply OK Reset Cancel

Figure 6-1 Setting garbage collection policy in the Java Virtual Machine Process Definition page

6.2.3 Optimal Average Pause GC policy

The Optimal Average Pause GC policy reduces pause times to a more practical level. However, the pauses will occur more frequently. Optimal Average Pause is similar to Optimal Throughput, but with the mark and sweep phases running partially concurrent. In addition, some of the work for Mark and Sweep phases have been moved to allocation time. It is designed to reduce Stop-The-World pause time. Use this policy when the application requires good response times even when processor-intensive events occur during application lifetime.

To set the GC policy to use Optimal Average Pause, enter `-Xgcpolicy:optavgpause` into the Generic JVM arguments field in the Java Virtual Machine Process Definition page in the WebSphere Application Server administration console.

6.2.4 Generational Concurrent GC policy

The Generational Concurrent GC policy is designed to provide fast collection for short-lived objects. Use this policy when the application has high object allocation and death rates. This policy is most likely the best policy to choose for WebSphere applications because it produces a smaller footprint and reduces fragmentation of the heap. However, it is possible that this policy may degrade overall performance a small amount, but with the reduced pause times this may still be a desirable choice.

To set the GC policy to use Generational Concurrent, enter `-Xgcpolicy:gencon` into the Generic JVM arguments field in the Java Virtual Machine Process Definition page in the WebSphere Application Server administration console.

Note: Manual tuning is required when using Generational Concurrent policy.

Because manual tuning of the Generational Concurrent policy is required, a brief description of it is provided in “Tuning the Generational Concurrent GC policy” on page 311.

Generational Concurrent GC policy

The heap is divided into two generational areas: nursery and tenured. The nursery space is where all new objects are allocated. The tenured space contains older objects that persist beyond a few collections. The nursery is collected often. The tenured space is collected less frequently using the Optimal Average Pause policy. Objects that survive a certain number of collections (called the tenure age) are promoted to the tenured space.

For a more detailed description of Generation Concurrent, refer to *IBM Java 5 Diagnostics Guide*.

Tuning the Generational Concurrent GC policy

The goal when tuning the Generational Concurrent policy is to create a tenure space large enough to hold all persistent application objects, while also trying to prevent a compaction from occurring in the tenure space. The Generational Concurrent GC policy is tuned by finding the appropriate nursery and tenure heap spaces for the running application. Running with the defaults, the IBM JVM will dynamically size the nursery and tenured spaces, but this may not give the most optimal performance.

To tune the Generation Concurrent GC policy, follow these steps:

1. Determine a rough estimate of the size of the persistent object store.
2. Make the tenure space size large enough so that a collection never occurs, or occurs very infrequently.
3. Tune the nursery space size by balancing throughput against pause times.

Execute test run under normal load

Using available performance and load test applications, such as Rational Performance Tester, schedule a test run of the application under normal load on a production-like environment. For information about how to use Rational Performance Tester including an example of a performance run, see 10.5.3, “IBM Rational Performance Tester” on page 407.

Determine tenure space size

Because the tenured space is where long-term objects are held, it follows that the space should be large enough to hold all persistent application objects. Typically, for WebSphere applications, this will be a few hundred megabytes. To determine the tenure space, execute a load test of the application using `verbosegc` and `Optimal Thruput GC policy` (the default policy).

For an explanation about how to enable `verbosegc` and interpret the output, refer to 6.4.1, “Analyzing verbose output” on page 316. Investigate the `verbosegc` output to determine the size of the persistent object store within the heap.

Size the tenure space

Now execute test runs of the application with the Generational Concurrent policy (instead of the default policy) using the tenure space determined from the previous procedure. Analyze the `verbosegc` output to determine how frequently the tenured space is collected. The goal is to never have a collection in the tenured space.

Size the nursery space

The next step is to determine the size of the nursery. In general, a larger nursery is better for application throughput and a smaller nursery will have lower pause times. Start by setting the nursery size to be relatively large. Execute test runs under normal load while measuring response times and throughput. Increase the nursery size if better throughput is desired, or decrease the nursery size if lower pause times are desired.

Generational Concurrent runtime options

Table 6-1 lists the runtime options that you use to tune the Generation Concurrent GC policy.

Table 6-1 Generational concurrent runtime options

Runtime option	User Guide description
-Xmn<value>	Sets the initial and maximum size of the new (nursery) heap to the specified value. Equivalent to setting both -Xmns and -Xmx. If you set either -Xmns or -Xmx, you cannot set -Xmn. If you attempt to set -Xmn with either -Xmns or -Xmx, the VM will not start, returning an error. By default, -Xmn is selected internally according to your system's capability. You can use the <code>-verbose:sizes</code> option to find out the values that the VM is currently using. If the scavenger is disabled, this option is ignored.
-Xmns<value>	Sets the initial size of the new (nursery) heap to the specified value. By default, this option is selected internally according to your system's capability. This option will return an error if you try to use it with -Xmn. If the scavenger is disabled, this option is ignored.
-Xmx<value>	Sets the maximum size of the new (nursery) heap to the specified value. By default, this option is selected internally according to your system's capability. This option will return an error if you try to use it with -Xmn. If the scavenger is disabled, this option is ignored.
-Xmo<value>	Sets the initial and maximum size of the old (tenured) heap to the specified value. Equivalent to setting both -Xmos and -Xmox. If you set either -Xmos or -Xmox, you cannot set -Xmo. If you attempt to set -Xmo with either -Xmos or -Xmox, the VM will not start, returning an error. By default, -Xmo is selected internally according to your system's capability. You can use the <code>-verbose:sizes</code> option to find out the values that the VM is currently using.

Runtime option	User Guide description
-Xmos<value>	Sets the initial size of the old (tenure) heap to the specified value. By default, this option is selected internally according to your system's capability. This option will return an error if you try to use it with -Xmo.
-Xmox<value>	Sets the maximum size of the old (tenure) heap to the specified value. By default, this option is selected internally according to your system's capability. This option will return an error if you try to use it with -Xmo.
-Xmr<value>	Sets the size of the Garbage Collection "remembered set". This is a list of objects in the old (tenured) heap that have references to objects in the new (nursery) heap. By default, this option is set to 16 kilobytes.
-Xmr<value>	Sets the remembered maximum size setting

6.2.5 Subpool GC policy

The Subpool GC policy is designed to take advantage of IBM System p hardware features for multi-CPU servers. Subpool uses the Optimal Throughput collector, but has a different allocation scheme. The goal the subpool policy strives for is reduced contention on allocation lock. This is accomplished by dividing the heap into sections based on the size of allocation requests. Subpool is best suited for applications with high allocation rates on many threads, and is best suited for 16-way or higher servers.

6.2.6 Additional runtime options

Additional runtime options exist for the garbage collector that may be useful in particular circumstances. A complete list of runtime options is provided in *Java 5 User's Guide*. A few of them are explained in the following sections.

Additional garbage collection runtime options

The GC runtime options in Table 6-2 on page 314 are available to all four GC policies. They may be used to overcome special problems specific to a particular application.

Table 6-2 Additional garbage collection runtime options

Runtime option	User Guide description	Use when
-Xconcurrentbackground <number>	Specifies the number of low priority background threads attached to assist the mutator threads in concurrent mark. The default is 1.	Increasing this value does not provide better performance. Changing this value is not recommended.
-Xconcurrentlevel <number>	Specifies the allocation “tax” rate. It indicates the ratio between the amount of heap allocated and the amount of heap marked. The default is 8.	Changing this value from the default is not recommended.
-Xdisableexcessivegc	Disables the throwing of an OutOfMemoryError if excessive time is spent in the GC. By default, this option is off.	Use this option only when an OutOfMemoryError is not desired during excessive GC.
-Xdisableexplicitgc	Signals to the VM that calls to System.gc() should have no effect. By default, calls to System.gc() trigger a garbage collection.	Use when application contains excessive calls to System.gc(). Normally, applications should never call System.gc().
-Xgcthreads<number of threads>	Sets the number of helper threads that are used for parallel operations during garbage collection. By default, the number of threads is set to the number of physical CPUs present -1, with a minimum of 1.	The default is the best option to use. For multi-processor installations, the number may be reduced to help application performance, if GC performance is adequate.
-Xnopartialcompactgc	Disables incremental compaction.	Use only as a last resort on a fragmented heap since this option forces full compaction.
-Xpartialcompactgc	Causes incremental compaction to occur on every GC cycle. Partial compaction happens by default, but will occur only when necessary - such as when the heap is fragmented.	Normally not necessary to set. Use only as a last resort if heap fragmentation is a problem. Compacting on every GC cycle will significantly increase pause times.

6.3 The large object area

In order to help reduce heap fragmentation, the IBM GC policies have segmented the tenure space into two additional spaces. The large object area (LOA) is an area of the tenure spaced used solely to satisfy allocations for large

objects that cannot be satisfied in the main area of the heap, or the small object area (SOA). A large object is considered to be any object 64 KB or greater in size. Tuning the large object area will not necessarily yield better performance outright; however, it may help reduce heap fragmentation, which can affect garbage collection execution time.

How JVM uses the LOA

All object allocations are first attempted in the SOA. If this is not possible and the object size is greater than or equal to 64 KB, then allocation is retried in the LOA. The LOA expands or shrinks according to the algorithm specified in the Java Diagnostics Guide. If the LOA is not used after a few collections, it is shrunk down to nothing. For the Generational Concurrent GC policy, the default initial size of the LOA is 5% of the current tenure area. The default maximum size is 50% of the current tenure area.

Because the LOA is sized dynamically, it is usually not necessary to change the default settings. But an application that utilizes many large object may require sizing of the large object area.

Note: The LOA is allocated for all GC policies except subpool.

Sizing the large object area

To determine the values for initial and maximum LOA sizes, test runs of typical loads in a production-like configuration are needed. After the load tests, examine the verbosegc to determine the usage characteristics of the large object area. If the LOA is not utilized much, then the default settings will provide good results. However, if the LOA is utilized, then setting the LOA runtime options may help.

It is unlikely that an application will require a large object area that is more than 50% of the tenure space. However, if the diagnostics show that the LOA is consistently full at this default size, then adjust the maximum LOA to a higher percentage. Additionally, the initial LOA percentage may be increased as well. The initial and maximum percentages should never be the same amount. You should always allow room for the LOA to expand and shrink.

Table 6-3 Large object area runtime options

Runtime option	User Guide description
-Xloainitial <number>	<number> is between 0 and 0.95, which specifies the initial percentage of the current tenure space allocated to the large object area (LOA). The default is 0.05 or 5%.

Runtime option	User Guide description
-Xloamaximum <number>	<number> is between 0 and 0.95, which specifies the maximum percentage of the current tenure space allocated to the large object area (LOA). The default is 0.5 or 50%.
-Xconmeter:<soa loa dynamic>	Determines which area's usage, the large object area (LOA) or the small object area (SOA), is metered and hence which allocations are taxed during concurrent mark. The allocation tax is applied to the selected area. If -Xconmeter:dynamic is specified, the collector dynamically determines the area to meter based on which area is exhausted first. By default, the option is set to -Xconmeter:soa.

6.4 Heap sizing

Heap sizing is a very important factor in improving an application's performance. Determining the proper initial and maximum heap sizes can greatly improve an application's performance. Heap must always be kept within the physical memory space. If the heap is larger than what is available in physical memory and pages to disk, performance will be severely degraded.

Note: Always keep the heap within physical memory.

The following steps provide an overview on heap sizing. Detailed descriptions of each step are provided in the sections that follow.

1. Run the application with verbose:gc turned on.
2. Run the application with no load to determine the heap size at steady state. This will be a rough estimate of the initial heap size.
3. Run the application under stress to determine the maximum heap size.
4. Run the application under normal usage to determine if the heap expands or contracts, or if a steady state is achieved.

6.4.1 Analyzing verbose output

Verbose garbage collection is enabled by the **-verbosegc** (or alternatively, **-verbose:gc**) command line option.

To enable verbosegc using the WebSphere Application Server console, check the Verbose garbage collection box in the Java Virtual Machine settings page.

To get to the Java Virtual Machine settings page:

1. Open the WebSphere integrated solutions console.
2. In the navigation tree, expand **Servers** and click **Application servers**.
3. Select the desired application server.
4. Under the Server Infrastructure heading, expand **Java and Process Management** and click **Process Definition**.
5. Under Additional Properties, select **Java Virtual Machine**.

Reading verbosegc output

The verbosegc output from WebSphere Application Server V6.1 is stored in the native_stderr.log file. This file is found in the server logs directory. Verbosegc content has changed formats from Java 1.4.2 to Java 5. Example 6-1 shows sample verbosegc output.

Verbosegc in Java 5 is outputted as xml. The xml format is more difficult for a person to read, but it provides a better format for monitoring and analysis tools to read.

Example 6-1 Sample verbosegc output in native_stderr.log

```
<af type="tenured" id="7" timestamp="Fri Sep 29 12:27:16 2006"
intervalms="6384.
924">
  <minimum requested_bytes="4194328" />
  <time exclusiveaccessms="0.823" />
  <tenured freebytes="20754960" totalbytes="104857600" percent="19" >
    <soa freebytes="17609232" totalbytes="101711872" percent="17" />
    <loa freebytes="3145728" totalbytes="3145728" percent="100" />
  </tenured>
  <gc type="global" id="7" totalid="7" intervalms="6385.221">
    <compaction movecount="454417" movebytes="36596648" reason="compact to meet
allocation" />
    <classloadersunloaded count="0" timetakenms="3.112" />
    <refs_cleared soft="133" weak="77" phantom="1" />
    <finalization objectsqueued="192" />
    <timesms mark="169.897" sweep="4.660" compact="335.373" total="513.365" />
    <tenured freebytes="68253888" totalbytes="104857600" percent="65" >
      <soa freebytes="64059584" totalbytes="100663296" percent="63" />
      <loa freebytes="4194304" totalbytes="4194304" percent="100" />
    </tenured>
  </gc>
  <tenured freebytes="64059560" totalbytes="104857600" percent="61" >
    <soa freebytes="59865256" totalbytes="100663296" percent="59" />
    <loa freebytes="4194304" totalbytes="4194304" percent="100" />
  </tenured>
```

```
<time totalms="514.485" />
</af>
```

In Example 6-1 on page 317, the event described is a collection caused by an allocation failure. The cause of each GC event can be discerned by the first tag; in this case, the tag for an allocation failure is `<af>`. Other causes of GC events are from either a concurrent collection or from a call to `System.gc()`. Concurrent GC events begin with `<con>`, while system GC events begin with the `<sys>` tag.

Table 6-4 Common tags found in verbosegc output

Tag	Description
<code><af></code>	An allocation failure has occurred
<code><sys></code>	A call to <code>System.gc()</code> has occurred
<code><con></code>	A concurrent event has taken place; can be contained within an <code><af></code>
<code><time></code>	Total amount of processor time for a particular event
<code><minimum></code>	Number of bytes requested that triggered the allocation failure
<code><nursery></code>	Status of the nursery area
<code><tenured></code>	Status of the tenured area
<code><gc></code>	A garbage collection occurred; can be either global or scavenger
<code><compaction></code>	A compaction occurred
<code><expansion></code>	Heap expansion occurred

This list of tags is only an overview of the most important ones. For a more detailed description of how to read xml verbosegc, see the Garbage Collector diagnostics in *IBM Java 5 Diagnostics Guide*.

Garbage collection analysis tools

Tools are available to help administrators read and understand verbosegc output. The IBM Pattern Modeling and Analysis Tool for Java Garbage Collector, which is available from the IBM alphaWorks® Web site, is an excellent choice. It is available for download at the following URL:

<http://alphaworks.ibm.com/tech/pmat>

IBM Pattern Modeling and Analysis Tool for Java Garbage Collector

The IBM Pattern Modeling and Analysis Tool for Java Garbage Collector (PMAT) parses the verbosegc output, provides analysis and graphs, and recommends configuration changes. See Figure 6-2 for a view of this tool.

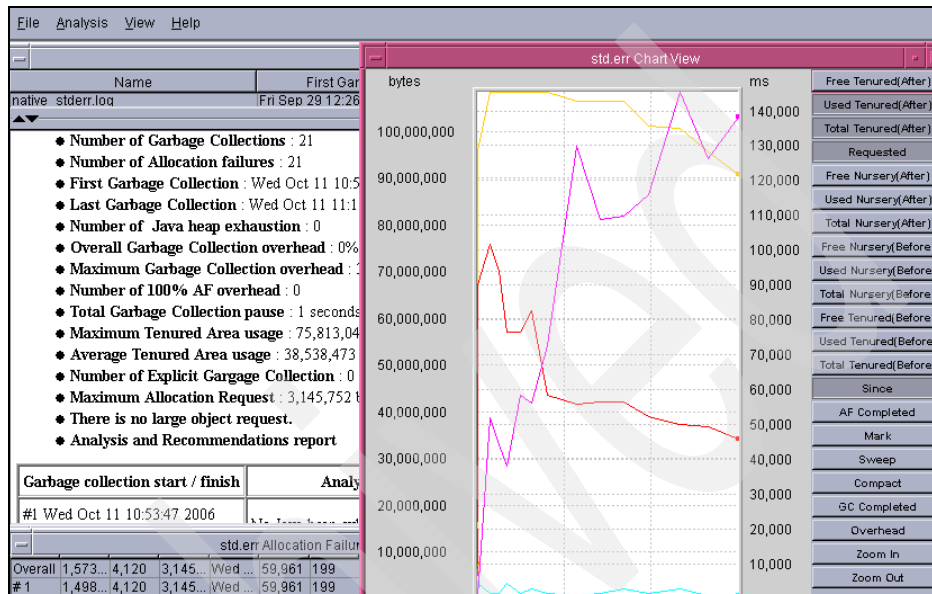


Figure 6-2 Pattern Modeling and Analysis Tool from IBM alphaWorks

6.4.2 Determining initial heap size

Starting a WebSphere application with an appropriate heap size can be an important factor in optimizing performance. Running an application on WebSphere Application Server using the default heap size settings is not usually recommended. Processing time is wasted expanding the heap up to the normal operating size. Although the heap size may change greatly during an application's lifetime, it is generally a good idea to set the initial heap size close to the actual size that the application will operate at.

An optimal application would have an initial and maximum heap of the same size; in fact, for performance testing, the initial and maximum heap sizes should be the same. But most applications are dynamic in nature, so a heap that fluctuates in size is expected.

Determining the proper initial heap size for a particular application is subject to interpretation. Starting too small will result in too much time spent in heap expansion. Starting with too large a value may result in a fragmented heap

because a collection may not occur until after a long period of time. Remember that a collection only occurs after an allocation failure happens.

To determine an appropriate initial heap size for an application, an administrator will need to know the approximate runtime values of the heap. A test run under normal load in a production-like environment will provide these numbers. The heap size may vary quite a bit, but a good sense of the average range will become apparent. A useful starting point is to set the initial heap size to be at the low end of the range discovered during the test run. This method balances the cost of heap expansion against the possibility of a fragmented heap due to too large a setting.

Because the heap size is so important to WebSphere Application Server, separate fields have been created for the initial and maximum heap size settings for the Process Definition in the Java Virtual Machine settings page. To set the initial heap size using the WebSphere Application Server console, enter the size (in megabytes) into the Initial Heap Size field on the Java Virtual Machine settings page.

Figure 6-3 on page 321 shows the JVM settings page for the Process Definition of the server known as server1.

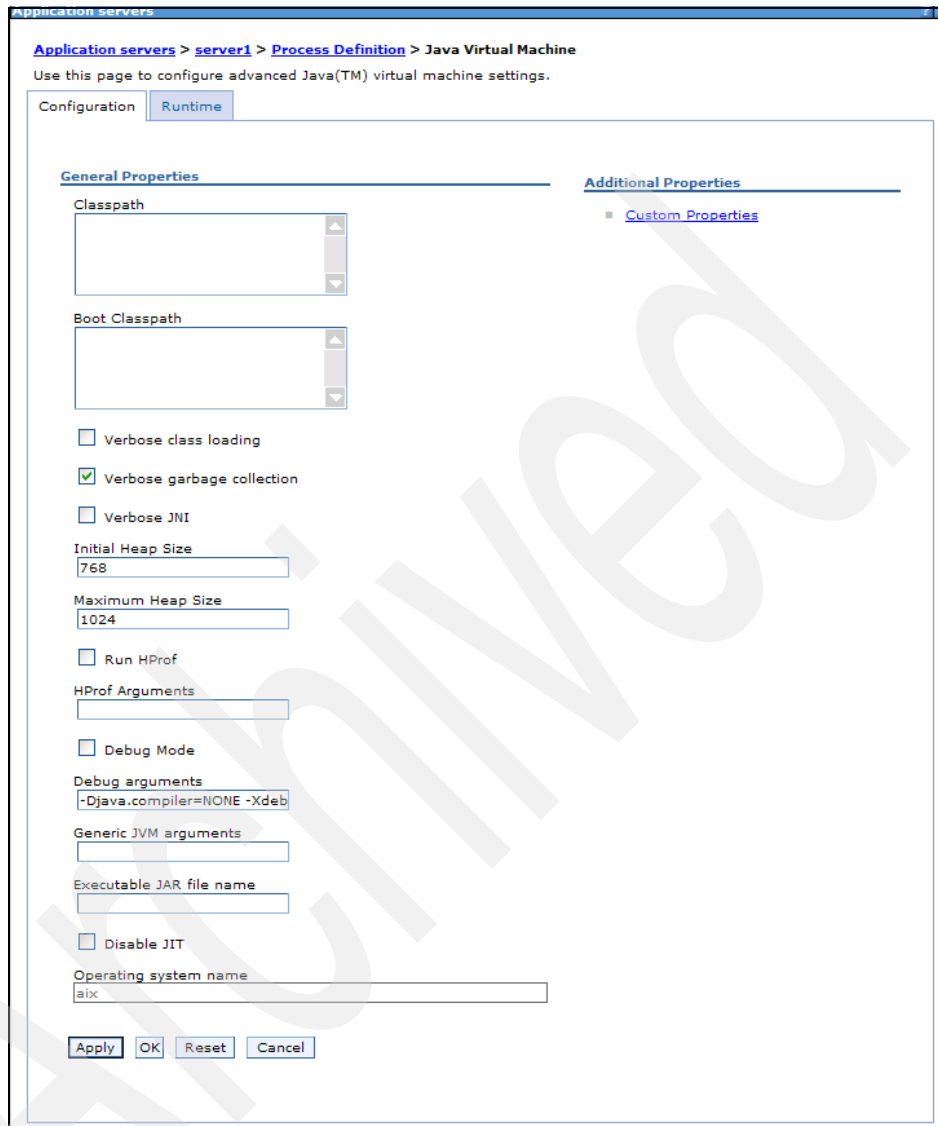


Figure 6-3 Java Virtual Machine settings pane for Process Definition

6.4.3 Determining maximum heap size

The conventional wisdom for determining the maximum heap size has been to base it on percentage utilization, usually about 50%. For example, a running application would show the percent freebytes as 50%. Example 6-2 on page 322 shows this scenario.

```
...  
<tenured freebytes="53276072" totalbytes="104857600" percent="50" >  
...
```

However, this approach does not take into account the purpose in sizing the heap. The goal is to have a heap large enough to avoid too many allocation failures, while also not being so large that collections become too costly a procedure, because collections take longer on larger heaps. Also, the possibility of the heap becoming fragmented exists if the heap is so large that no collection occurs for a long period of time. When an allocation failure occurs when the heap is heavily fragmented, the collection will require a compaction that could take an unacceptably long time.

Note: Keep in mind the goal when determining the maximum heap size. It should be large enough so that the number of allocation failures is minimal, but yet small enough so that collections do not become too costly. Larger heaps run the risk of becoming fragmented which can in turn lead to costly compactons.

The correct approach to take is to analyze verbosegc output to see what the collector is doing, and then size to optimize the GC operations.

To determine an optimal maximum heap size, perform the following steps:

1. Start with a size that is at least 30% larger than the maximum heap used, or 50% larger than the average heap utilized, if the heap tends to fluctuate in size.
2. Monitor the application for performance. Use these numbers as a baseline.
3. At the same time, determine the allocation failure rate from analyzing verbosegc output.
4. Increase the maximum heap size if allocation failures are quite frequent.
5. Decrease the maximum heap size if allocation failures are infrequent, but cause pause times to be unacceptably too long. A smaller maximum heap size also will be necessary if, during an extended run, the heap becomes fragmented.

It must be said that it is impossible perform test runs that perfectly mimic application usage. So how can you size the heap on a production system? The answer is to monitor both application performance and the allocation failure rate on the production server. Try to keep the allocation failure rate as low as possible (for example, once every ten minutes or more).

6.4.4 Expansion and contraction

There is a cost associated with expanding and shrinking the heap because processor time is spent in the overhead involved with paging. It is good policy to reduce the amount of times the heap changes size by monitoring the heap size and tuning accordingly. The ideal situation is to have a heap that never changes size. But if the heap does change size significantly and often during an application's lifetime, then it is worth the effort to tune heap expansion and shrinkage policies. Additionally, it is prudent at this time to investigate whether increasing the page size will help application performance. To determine whether increasing the page size will help you application, see 6.6, "Using large page sizes" on page 329 in this chapter.

Tuning the heap expansion and shrinkage policies will improve an application's performance by reducing the amount of time spent paging. Table 6-5 on page 324 describes heap expansion and contraction runtime options. The design specifics of heap expansion and shrinkage are explained fully in *IBM Java 5 Diagnostics Guide*.

The following runtime options are used to set the heap expansion and shrinkage policies:

- ▶ **-Xminf** and **-Xmaxf** set the free space after collection.
- ▶ **-Xmine** and **-Xmaxe** set the expansion amount.
- ▶ **-Xmint** and **-Xmaxt** set the GC collection time threshold.

Example 6-3 shows what an expansion of the heap looks like in `verbosegc`. Note that the expansion will be contained within an allocation failure or possibly some other GC event.

Example 6-3 Verbosegc output showing heap expansion

```
...
<expansion type="tenured" amount="3146752" newsize="7341056"
timetaken="0.139" reason="satisfy allocation request" />
...
```

Example 6-4 shows what a contraction of the heap looks like in `verbosegc`.

Example 6-4 Verbosegc output showing heap contraction

```
...
<contraction type="tenured" amount="5327872" newsize="101239808"
timetaken="0.020" reason="excess free space following gc" />
...
```

Example 6-5 shows a default heap expansion which results in 30% free heap.

Example 6-5 Verbosegc output of default heap expansion event

```
<af type="tenured" id="5" timestamp="Wed Oct 11 10:53:49 2006"
intervalms="409.840">
  <minimum requested_bytes="4120" />
  <time exclusiveaccessms="0.485" />
  <tenured freebytes="0" totalbytes="23528448" percent="0" >
    <soa freebytes="0" totalbytes="21411840" percent="0" />
    <loa freebytes="0" totalbytes="2116608" percent="0" />
  </tenured>
  <gc type="global" id="5" totalid="5" intervalms="409.937">
    <expansion type="tenured" amount="9783296" newsize="33311744"
timetaken="8.881" reason="insufficient free space following gc" />
    <refs_cleared soft="0" weak="3" phantom="0" />
    <finalization objectsqueued="2" />
    <timesms mark="19.728" sweep="0.265" compact="0.000" total="28.949" />
    <tenured freebytes="9999056" totalbytes="33311744" percent="30" >
      <soa freebytes="9999056" totalbytes="33311744" percent="30" />
      <loa freebytes="0" totalbytes="0" percent="0" />
    </tenured>
  </gc>
  <tenured freebytes="9994936" totalbytes="33311744" percent="30" >
    <soa freebytes="9994936" totalbytes="33311744" percent="30" />
    <loa freebytes="0" totalbytes="0" percent="0" />
  </tenured>
  <time totalms="29.517" />
</af>
```

Table 6-5 lists and describes the expansion and contraction runtime options for heap.

Table 6-5 Heap expansion and contraction runtime options

Runtime option	Description
-Xmine<size>	Sets the minimum amount by which the garbage collector expands the heap. Typically, the garbage collector expands the heap by the amount required to restore the free space to 30% (or the amount specified using -Xminf). The -Xmine option sets the expansion to be at least the specified value. For example, -Xmine50M sets the expansion size to a minimum of 50 MB. By default, the minimum expansion size is 1 MB.

Runtime option	Description
-Xminf<size>	Specifies the minimum percentage of heap that should be free after a garbage collection. If the free space falls below this amount, the JVM attempts to expand the heap. Specify the size as a decimal value in the range 0-1; for example, a value of -Xminf0.3 requests the minimum free space to be 30% of the heap. By default, the minimum value is 0.3.
-Xmaxe<size>	Sets the maximum amount by which the garbage collector expands the heap. Typically, the garbage collector expands the heap when the amount of free space falls below 30% (or the amount specified using -Xminf), by the amount required to restore the free space to 30%. The -Xmaxe option limits the expansion to the specified value. For example, -Xmaxe10M limits the expansion to 10 MB. By default, there is no maximum expansion size.
-Xmaxf<size>	Specifies the maximum percentage of heap that must be free after a garbage collection. If the free space exceeds this amount, the JVM attempts to shrink the heap. Specify the size as a decimal value in the range 0-1. For example, -Xmaxf0.5 sets the maximum free space to 50%. The default value is 0.6 (60%).
-Xmint<value>	Specifies the minimum percentage of time which should be spent in Garbage Collection. If the percentage of time drops below this value, the JVM attempts to shrink the heap. Specify the percentage as a decimal in the range 0-1. For example, -Xmint0.1 sets the minimum time spent in Garbage Collection to 10%. The default value is 5%.
-Xmaxt<value>	Specifies the maximum percentage of time which should be spent in Garbage Collection. If the percentage of time rises above this value, the JVM attempts to expand the heap. Specify the percentage as a decimal in the range 0-1. For example, -Xmaxt0.2 sets the maximum time spent in Garbage Collection to 20%. The default value is 13%.

6.5 Using shared classes

WebSphere Application Server V6.1 offers the ability to share classes among VMs. The IBM JVM allows the sharing of system and application classes between VMs by storing and dynamically updating them in a cache in shared memory. Class sharing reduces virtual memory footprint and startup time for subsequent VMs started after the cache has been created.

Startup time is decreased because classes are loaded from memory rather than from disk. The shared class cache is independent of any active VM and persists beyond the lifetime of the VM. The usage of shared classes is completely transparent because the cache is dynamically updated and no restrictions are placed on the VMs that utilize shared classes.

Important: For a detailed discussion about using shared classes, including usage examples, refer to “Shared Classes” on page 201.

Shared classes are particularly useful on servers that use more than one VM that are running similar code and where VMs frequently start up and shut down. The maximum theoretical cache size is 2 GB, but the actual limit is dependent upon the amount of available virtual address space. This space is shared with the Java heap, so increasing the size of the heap will reduce the size of shared class cache that can be made.

Use shared classes only if the following criteria are met:

- ▶ Multiple instances of a JVM will be run on the same machine.
- ▶ The JVMs are of the same release.
- ▶ The JVMs sharing classes are all either 32-bit or 64-bit. There is no sharing between 32-bit and 64-bit VMs.
- ▶ The user IDs running the JVMs must be in the same group.

Note: Use care if runtime byte code modification is used. Refer to *Java User's Guide* for more information about shared classes and runtime byte code modification.

6.5.1 Creating multiple caches

It is possible to have multiple shared class caches on the same partition. To prevent a JVM from attaching to a cache, we recommend always specifying a name, rather than letting the default name be used. This way, when a JVM starts up, it will create its own cache rather than attaching itself to an undesired cache. Access can be further controlled and administered via group usage or through Java Security.

IBM SDK User Guide contains information about creating, populating, monitoring and using shared classes, and is available at the following address:

<http://www-128.ibm.com/developerworks/java/jdk/aix/>

For a closer look at class sharing, refer to the developerWorks article: “Java Technology; IBM Style: Class Sharing”, which is available at the following address:

<http://www-128.ibm.com/developerworks/java/library/j-ibmjava4/>

6.5.2 Tuning the shared class cache size

If shared classes are utilized, the cache size will need to be tuned to the specific application. This is a multistep process that involves stopping and starting the server, so it is best to perform this operation in a production-like test environment.

To determine optimum cache size, you must perform the following steps:

1. Stop all running Java applications.
2. Destroy the existing shared classes cache.
3. Perform a test load of the application using a large cache.
4. Use `printStats` to determine how much class data has been stored.
5. Use the `-Xscmx` runtime option to set the cache size to be slightly larger than the amount shown via `printStats`.

Here we present a simple demonstration showing how to tune the shared class cache size. This scenario assumes that multiple instances of the same application are installed and run on the same partition. It is possible to execute test loads of more than one application; simply start them all up in addition to the application started first in Step three.

Keep in mind that only the first Java application started with the cache size set will have its setting take effect. The shared class cache size setting only works when the cache is created. It is not possible to change the cache size after the cache has been created.

In the following sections, we explain the steps in greater detail.

Step 1 - Stop all running Java applications

Stop the server node and all running applications on the server. It is not possible to destroy a cache while a JVM is still attached to it. To ensure that no Java application, execute `ps -ef | grep java`. No output will be returned if the application server and all its applications have been stopped.

Step 2 - Destroy the existing shared classes cache

Execute the `java -Xshareclasses:destroyAll` command.

Step 3 - Perform a test load using large cache

Set up and execute a test load of the application using the **-Xscmx** option in the Generic JVM arguments field in the Process Definition page. Use a large value to start with, say 200 MB.

The default cache size is platform-dependent. On AIX it is 50 MB. Determine the shared cache name by executing **java -Xshareclasses:listAllcaches**. See Example 6-6 for sample output from the **listAllcaches** command.

Example 6-6 Sample shared classes listAllcaches output

```
/usr/IBM/WebSphere/AppServer/java/bin/java -Xshareclasses:listAllcaches
Shared Cache      OS shmid      in use      Last detach
time
webspherev61_system  4            3          Fri Sep 29
12:26:30 2006
```

Step 4 - Use printStats to determine how much class data has been stored

Stop the running application and view the statistics by executing the command **java -Xshareclasses:printStats**. Note that it is not necessary to stop the application, but it is recommended. Example 6-7 displays sample output from the **printStats** command.

Determine the stored size by multiplying the cache size by the percentage full. In the example shown, we set the cache to be 200 MB. After running a test load and printing the shared cache statistics, we saw that the cache was 35% full. So the stored size of the shared class cache will be 35% of 200 MB, or 70 MB.

Example 6-7 Sample shared classes printStats output

```
/usr/IBM/WebSphere/AppServer/java/bin/java
-Xshareclasses:name=webspherev61_system,printStats
Current statistics for cache "webspherev61_system":
```

```
base address      = 0x0700000010000090
end address       = 0x070000001C7FFFF0
allocation pointer = 0x0700000014525B88
```

```
cache size        = 209715056
free bytes        = 135686036
ROMClass bytes    = 72506104
Metadata bytes    = 1522916
Metadata % used   = 2%
```

```
# ROMClasses      = 15769
# Classpaths      = 10
# URLs            = 25
# Tokens          = 0
# Stale classes   = 0
% Stale classes   = 0%
```

Cache is 35% full

Step 5 - Set the cache size to tuned value

A tuned cache size has now been determined for use in the production environment. Use the value determined in Step 4 plus an additional 10 to 20 percent.

6.6 Using large page sizes

AIX provides multiple page sizes to help increase performance of memory intensive applications. Because all applications running on WebSphere Application Server with a large heap will realize a benefit from using larger page sizes, their use is recommended.

This section explains how to use the large page sizes available on AIX. For an in-depth explanation of the multiple page sizes offered in AIX and how to use them, refer to the white paper “Guide to Multiple Page Size Support on AIX 5L Version 5.3”, which is available at the following address:

http://www-03.ibm.com/servers/aix/whitepapers/multiple_page.pdf

6.6.1 Large page support in AIX

System p5 processors support 4 KB and 16 MB page sizes. The introduction of System p5+ and later processors has added two new page sizes: 64 KB and 16 GB.

AIX Versions 5.2 and later support the 16 MB page sizes. AIX 5.3 ML 5300-04 and later support all four page sizes: 4 KB, 16 MB, 64 KB, and 16 GB. Table 6-6 on page 330 shows the support matrix for medium and large page sizes.

The default page size is always 4 KB, which processes will use unless specified otherwise. 16 MB and 16 GB page sizes are restricted and thus require system configuration changes.

Table 6-6 Large page size support matrix

Machine type	Available page sizes	AIX support
POWER4 or later	4 KB, 16 MB	64-bit and 32-bit AIX 5.2 and later
POWER5+ or later	4 KB, 64 KB, 16 MB, 16 MB	64-bit AIX 5.3 ML 5300-04 and later

6.6.2 64 KB size for POWER5+ and later systems

POWER5+ and later systems support a new 64 KB page size. No system configuration changes are necessary to enable 64 KB page size; the AIX 5L kernel will automatically configure and manage the pool of 64 KB page frames.

6.6.3 Java large page support

The IBM SDK on AIX supports the use of large pages through the -Xlp option. To set Java to utilize the larger page sizes, set the -Xlp option in the Generic JVM arguments field in the Java Virtual Machine settings page in the Process Definition. If no option is specified when using -Xlp, it defaults to 16 MB.

The large page sizes are only available to Java if AIX has been configured for large pages. The next section describes how to set up a WebSphere Application Server to utilize large pages.

6.6.4 Changing page sizes in WebSphere Application Server

Enabling an installation to utilize large pages is a multistep process. The following steps will guide you through enabling large page sizes on a WebSphere Application Server installation:

1. Determine how much large page space is required.
2. Enable large page use in AIX.
3. Enable large page use in WebSphere Application Server.
4. Verify actual large page usage.

In the following sections, we explain each step in more detail.

Step 1 - Determine how much large page space is required on the server

If 16 MB page sizes are used, AIX must be configured to utilize this feature. This will require an estimate of the amount of space that the application will need.

Make the size larger than the maximum heap size, while accounting for future application memory needs.

Step 2 - Enable large page sizes in AIX

Refer to Table 6-6 on page 330 to determine the page sizes supported on your particular server. Alternatively, you can issue the `pagesize -af` command.

The following example shows how to allocate 4 GB of RAM for use for 16 MB large pages. All commands must be run as root user. To enable the use of large pages in AIX, follow these steps:

- ▶ Use the Virtual Memory Manager command `vmo` in conjunction with the `-o` and `-r` flags to set the new value permanently. The `-o` option sets the new size and the `-r` flag tells the VMM to set this value during boot. You must set both `lpgg_regions` and `lpgg_size`. Issue the command:

```
vmo -r -o lpgg_regions=256 -o lpgg_size=16777216
```

This command sets 256 regions of 16 MB pages, which will result in 4 GB of RAM to be allocated, and only applications that are enabled to utilize 16 MB pages will be able to use this section of memory.

- ▶ These settings do not require a reboot, so they must be added to the boot image in order to take effect on every boot. To do this, issue the command:

```
bosboot -ad /dev/ipldevice
```

- ▶ Reboot the server.
- ▶ The user that the WebSphere process executes under will need the capability to utilize the 16 MB large pages. This is accomplished through the command:

```
chuser capabilities=CAP_BYPASS_RAC_VMM,CAP_PROPAGATE $USER
```

Where `$USER` is the user ID that WebSphere runs under.

AIX is now ready for WebSphere to utilize 16 MB large pages.

Step 3 - Enable large page use in WebSphere Application Server

To allow the WebSphere Application Server process to utilize the available large page size, set the `LDR_CNTRL` environment variable from within the console.

- ▶ Open the WebSphere integrated solutions console.
- ▶ In the navigation tree, expand **Servers**, click **Application servers**, and select the appropriate server.

- ▶ Expand **Java and Process Management** and click **Process Definition**.
- ▶ Click **Environment Entries**, then click **New**.
- ▶ Set LDR_CNTRL to the value LARGE_PAGE_DATA=Y. Refer to Figure 6-4 to see an example.

Application servers > **server1** > **Process Definition** > **Environment Entries** > **New**

Specifies an arbitrary name-value pair. The value is a string that can set internal system configuration properties.

Configuration

General Properties

* Name: LDR_CNTRL

* Value: LARGE_PAGE_DATA=Y

Description: Use large pages

Apply OK Reset Cancel

Figure 6-4 Setting LDR_CNTRL in the Process Definition

Step 4 - Verify actual large page usage

You can use the **vmstat** command to see page size usage statistics. To see statistics displayed on a per-page basis, enter the **vmstat -P all** command. See Example 6-8 for sample output that includes 16 MB large pages in use. The actual pages used appear in column **avm**. In the example, twenty-one 16 MB pages are in use.

Example 6-8 View page size statistics per page

```
# vmstat -P all
```

```
System configuration: mem=8192MB
```

pgsz	memory		page						
	siz	avm	fre	re	pi	po	fr	sr	cy

4K	561424	382048	130451	0	0	0	0	0	0
64K	30447	2093	28354	0	0	0	0	0	0
16M	256	21	233	0	0	0	0	0	0

An alternative way to view the actual page usage, but on a per-process basis, is to use the `-Z` option. Example 6-8 on page 332 shows that a data page size of 16 MB is being used for the server process, which in this case has a PID of 364600.

Example 6-9 View page size statistics per process

```
# ps -Z -p 364600
      PID   TTY  TIME DPGSZ SPGSZ TPGSZ CMD
364600 pts/1 0:58  16M   4K   4K java
```

6.7 Dynamic logical partitions

Using dynamic logical partitions (DLPARs) involves the ability to move system resources (for instance, memory and processors) among partitions without requiring a reboot. These changes in resources are transparent to Java, but you should use care if resources are being removed.

Attention: If WebSphere is running on a partition with a single processor allocated and it is known that additional processor capacity will never be dynamically added, performance may be improved by exporting the following environment variable: `export NO_LPAR_RECONFIGURATION=1`.

Adding resources

Dynamically adding processors (or part of a processor) to a partition in which Java is running is seamless. If the Java application is utilizing almost all processing capacity, then the new additional capacity will be utilized.

Dynamically adding memory to a partition running a Java application will help application performance in certain circumstances.

- ▶ If the running application is currently paging a portion of the heap to disk because the maxheap was set higher than the amount of available physical memory
- ▶ If the application needs a larger heap size, and the soft maximum heap size has been set and the application is capable of being made aware of the increase in memory

Removing resources

Dynamically removing processing capacity from a partition in which Java is running should not affect the running application, other than possibly degrading performance if too much processing capacity is removed.

Note: Care should be taken to not remove too much processing capacity from a partition in which a JVM is running.

Dynamically removing memory is not advisable because the memory that is moved may contain part of the heap. If a portion of memory is removed from a partition and that portion of memory contains a section of the heap, then that section of the heap will have to be paged to disk, which will severely impact performance. .

6.8 Just-in-Time compiler

The IBM JVM includes the Just-in-Time compiler (JIT), which is enabled by default. The JIT compiler dynamically generates machine code for frequently used byte code sequences in Java applications. The JIT is tuned for long-running applications, such as WebSphere applications.

Note: The JIT should always be used in production environments.

Modifying JIT optimization levels is normally not recommended. Tuning the JIT should only be necessary for short-running applications.

Use **-Xquickstart** for short-running applications in which execution time is not concentrated into a small number of methods. **-Xnoquickstart** to be used for applications that demand that application methods be compiled at a high level of optimization very early in the execution period.

Refer to *IBM Java Diagnostics Guide* for more details about tuning the JIT:

<http://publib.boulder.ibm.com/infocenter/javasdk/v5r0/index.jsp>

High availability, clustering and WebSphere

Clustering WebSphere Application Server can serve two purposes: high availability (HA) and performance. This chapter describes methods you can use to implement high availability and related concepts using various WebSphere and AIX capabilities together.

7.1 Clustering WebSphere for availability

“Clustering” is a fundamental approach for accomplishing high availability. IBM WebSphere Application Server Network Deployment V6 offers a built-in application server clustering function and the HAManager for protecting WebSphere singleton services. Clustering application servers provides workload management and failover for applications that reside on the application server cluster. It is also possible to use external clustering software such as HACMP to provide availability clustering for WebSphere Application Server.

These two kinds of cluster failovers can be categorized as follows:

- ▶ IP-based cluster failover, using the IBM High Availability Cluster Multi-Processing for AIX 5L (HACMP)
- ▶ Non-IP cluster failover, such as WebSphere WLM and WebSphere HAManager.

Usually, IP-based cluster failover is slower (taking one to five minutes), and non-IP cluster failover is very fast (instantaneous). WebSphere V6 HAManager does not require extra software. However, IP cluster failover still relies on cluster software such as HACMP to provide the cluster information.

7.1.1 Availability considerations

An IBM WebSphere Application Server Network Deployment V6.1 environment will likely include one or more of each of the following processes as part of the WebSphere specific infrastructure:

- ▶ WebSphere Deployment Manager process
- ▶ WebSphere Node Agent processes
- ▶ WebSphere Application Server processes
- ▶ HTTP server processes
- ▶ Load Balancer processes
- ▶ Database server processes
- ▶ LDAP server processes

Making an entire end-to-end system highly available will include all of these and more, including any back-end computers the system relies on, any networking components such as routers, switches and firewalls, any distributed storage components such as SAN, NAS, or Networked File Systems, any remote security solutions with which the system must integrate, and so on.

This chapter focuses entirely on the WebSphere processes themselves, that is, the Deployment Manager, the Node Agent, and the Application Server instances. Keep in mind that this is just *one* part of the system, and the whole system must

be made highly available and performant in the same way in order to be of any value.

When using a clustered IBM WebSphere Application Server Network Deployment V6 environment, WebSphere process failures usually do not contribute much to the total number of client request failures, because the WebSphere process failover is instantaneous. However, a more serious failure (such as a physical problem with the machine on which an LPAR is running) can be mitigated by using HACMP.

Moreover, the HACMP configuration, when running on AIX/POWER 5, can be configured to automatically request and assign the resources required to the backup LPAR, meaning that these backup resources are not utilized until required. In a traditional non-LPAR environment, these resources would typically be machines acquired to stand idle, in case of failure, which is far less cost effective than using AIX/System p LPARs.

7.1.2 Clustering options

This section presents and evaluates two options for performing clustering with WebSphere Application Server on AIX/System p. To cluster for availability, we will investigate:

- ▶ WebSphere Clustering on micropartitions
- ▶ HACMP and LPARs

7.2 WebSphere clustering

A WebSphere cluster is a set of application servers that are managed together and participate in workload management. Application servers participating in a cluster can be on the same node, or on different nodes. A Network Deployment cell can contain no clusters, or it can have many clusters, depending on the need of the administration of the cell. The cluster is a logical representation of the application servers. It is not necessarily associated with any node, and does not correspond to any real server process running on any node. A cluster contains only application servers, and the weighted workload capacity associated with those servers.

When creating a cluster, note that it is possible to select an existing application server as the template for the cluster without adding that application server *into* the new cluster (the chosen application server is used only as a template, and is not affected in any way by the cluster creation). All other cluster members are then created based on the configuration of the first cluster member.

Cluster members can be added to a cluster in various ways: during cluster creation and afterwards. During cluster creation, one existing application server can be added to the cluster, or one or more new application servers can be created and added to the cluster. There is also the possibility of adding additional members to an existing cluster later on. Depending on the capacity of your systems, you can define different weights for the various cluster members.

Cluster members are required to have identical application components, but they can be sized differently in terms of weight, heap size, and other environmental factors. You must be careful, however, not to change anything that might result in different application behavior on each cluster member. This concept allows large enterprise machines to belong to a cluster that also contains smaller machines such as Intel-based Windows servers.

Starting or stopping the cluster starts or stops all cluster members automatically, and changes to the application are propagated to all application servers in the cluster.

In horizontal scaling, cluster members are created on multiple physical machines (or LPARs). This allows a single WebSphere application to run on several machines while still presenting a single system image, thereby 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. Client requests that overwhelm a single machine can be distributed over several machines in the system.

Failover is another important benefit of horizontal scaling. If a machine becomes unavailable, its workload can be routed to other machines containing cluster members. Horizontal scaling can handle application server process failures and hardware failures (or maintenance) without significant interruption to client service.

Using a WebSphere Application Server multiple machine or LPAR configuration eliminates a given application server process as a single point of failure. In IBM WebSphere Application Server Network Deployment V6, there are basically no dependencies on the administrative server process for security, naming, and transactions. Thus, a single process failure normally does not disrupt application processing.

WebSphere's workload management also takes care of failing over existing client requests to other, still available application servers, and of directing new requests only to available processes if an application server in the cluster should fail. In addition, WLM enables servers to be transparently maintained and upgraded while applications remain available for users. You can add additional cluster members to a cluster at any point, thus providing scalability and performance if an existing environment is not able to handle the workload any more.

The basic steps for clustering a WebSphere Application are described in *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392.

7.3 High availability on logical partitions

A System p logical partition (LPAR) is a natural home for a WebSphere node. A single physical server can host many nodes, with the added advantage that each node is a separate installation of the operating system and is isolated from the others. If your requirements include running separate applications on the same WebSphere environment, then running them in separate LPARs provides a far greater level of isolation. The following sections discuss the concepts of isolation, ease of use, and redundancy in relation to the use of LPARs.

7.3.1 Isolation

To start the discussion of isolation, imagine two applications (alpha and beta), which have different non-functional requirements. Application alpha needs to be available 24x7, and absolutely *cannot* become unavailable, on pain of severe financial loss. Application beta, on the other hand, is only used between 7am and 9pm GMT, and for the rest of the time is closed to external connections.

Application alpha's developers have spent a long time perfecting and tuning their code; they are aware that they are writing a business-critical application, and have gone through every single line of code many times in a complicated and thorough peer review process. Application beta was written by a number of contractors, who have since left the company, and is maintained on a "best efforts" basis by the system administrator (who has gained some invaluable basic Java skills in the process).

There is clearly a desire to separate these applications; the differing NFRs and the risk factor of the code practically demand separation. In a traditional WebSphere clustered environment, both of these applications might well be run on different WebSphere nodes on the same machine, or at an absolute minimum they would be run on separate application server instances. However, when they are still sharing the same operating system installation, what happens if the "loosely coded" application beta suddenly hits a bug and starts using up all of the system resources? It can make the entire machine run slow, and affect all applications running on the same machine, including the business-critical application alpha.

Running both of these applications on separate LPARs gives a far greater degree of isolation. Even if application beta were to begin to use up all of the memory or

CPU allocated to its own LPAR, it would be unable to affect application alpha. In addition, each LPAR could be sized so it is able to handle the predicted load for the application, and thus fewer resources would be consumed by the less-critical application beta.

Figure 7-1 shows applications alpha and beta deployed in different LPARs on the same server. The LPAR for application alpha is larger than that of application beta, and the two are separated; application beta cannot use up resources assigned to application alpha.

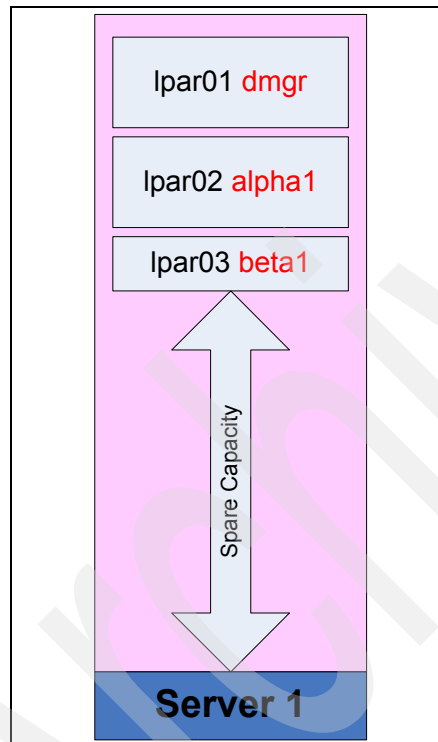


Figure 7-1 Applications alpha and beta deployed on separate LPARs

7.3.2 Ease of use

Often a limitation is placed on a WebSphere infrastructure architect such that the infrastructure must fit within an available set of machines. Even when a system is up and running, it can be difficult to expand upon it; limitations exist so it is difficult to purchase more machines, or data center floor space may become an issue. At such times, compromises are often made that result in applications with different non-functional requirements being co-located.

With logical partitioning and shared CPU LPARs, however, the ability to simply add a new LPAR and assign it resources, possibly taking unused resources from existing applications, is extremely useful in situations such as these. If you need to add a second “machine” to the infrastructure for application alpha, all you need to do is create a new LPAR and configure it into the WebSphere cell, as depicted in Figure 7-2.

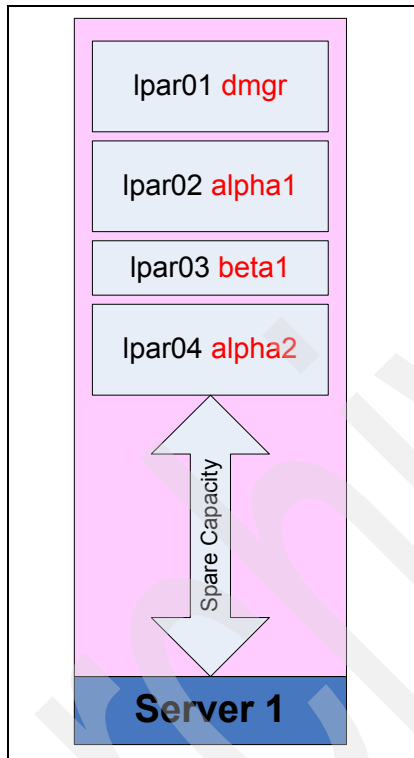


Figure 7-2 A second LPAR is configured for application alpha

7.3.3 Redundancy

In an LPAR-based environment, even the simplest application can use WebSphere clustering and have its own redundant LPARs to ensure continuity of service in the event of a serious error. This ability raises the standards of application deployment.

In order to implement physical redundancy (continuity of service in the event of physical machine failure), LPARs should be split across separate physical System p servers. Figure 7-3 on page 342 depicts the deployment of our earlier applications alpha and beta on a pair of System p servers.

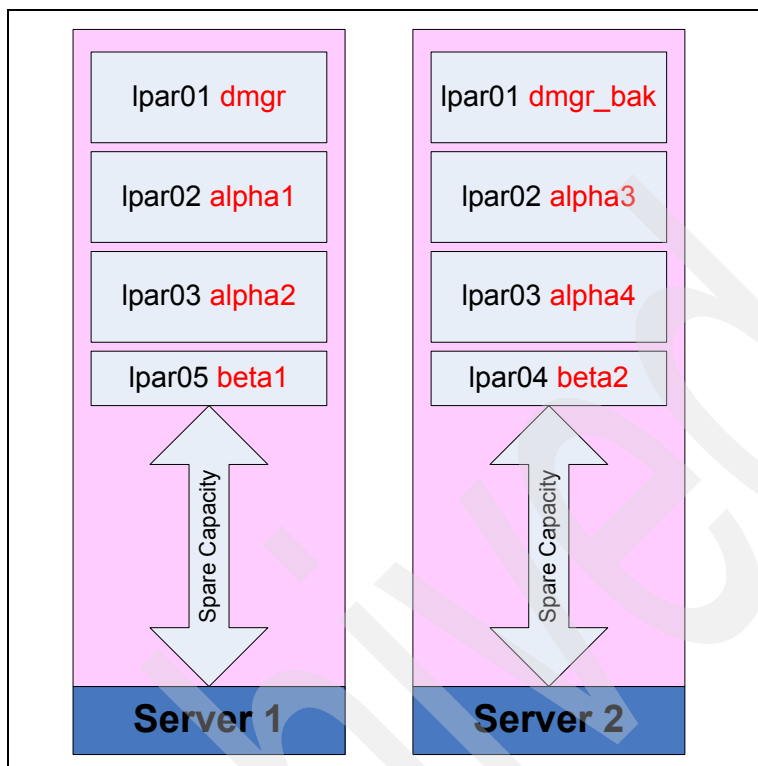


Figure 7-3 Deployment of applications alpha and beta in a redundant configuration

Each application has its own logical partitions, and the partitions are split across the two servers. WebSphere clustering is used to ensure that if one application server instance is unavailable, requests will not be routed to that LPAR.

7.4 Disaster recovery

High availability and disaster recovery are two related but very distinct topics, with different goals and different requirements. High availability (HA) involves providing redundancy and allowing automatic failover to ensure that the service being provided to a user is always available.

Disaster recovery (often referred to as DR) deals with true disasters, for example when an entire data center is hit by a power outage, or any other form of total, catastrophic system failure. This section discusses disaster recovery.

WebSphere clustering provides high availability. However, WebSphere clustering alone should not be used to provide disaster recovery. WebSphere cells should not span data centers, and a disaster recovery environment must be located physically distant to the main production site, to avoid being taken down by the localized effects of the same disaster that took down the production environment.

An advantage of running WebSphere on micropartitions is that the cost of having a “standby” partition is minimal. You can completely automate the addition of LPARs into a WebSphere cluster by preparing the LPAR previously; a WebSphere node would be already installed and configured into the cell, so thereafter you would only need to turn on the LPAR and assign resources to it. This ability allows customers to be very flexible with their WebSphere infrastructures.

However, each customer environment is different and has varying requirements, so no single design covers all possible configurations. But for the purpose of illustration, in the following section we describe a simple environment with two System p servers. One server acts as the production environment, and the other server performs the functions of preproduction and testing.

7.4.1 Environment configuration

Figure 7-4 on page 344 shows a high level representation of a hypothetical customer configuration that contains two System p servers (Server 1 and Server 2) in separate data centers.

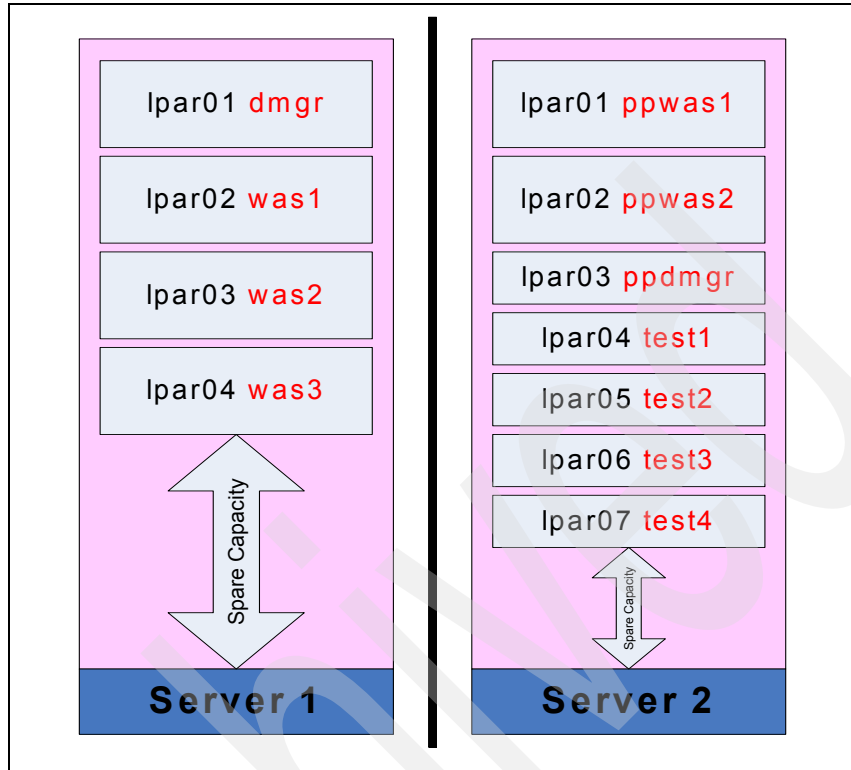


Figure 7-4 LPAR distribution on two servers (each in a separate data center)

Server 1 contains the production environment, with four large LPARs. Server 2, located in a remote data center, contains a preproduction environment with LPARs for the application servers as large as in production, but with a smaller LPAR for the deployment manager. In addition, Server 2 hosts four test environments, each on its own LPAR, which are used by developers to unit test their code.

This customer has decided that it cannot afford to have a “full DR environment” standing idle in its second data center. In the event of a true disaster, where the first data center is wiped out, the customer would be left frantically reconfiguring hardware and software in order to somehow get back up and running.

For simplicity, the applications running on this particular environment are assumed to be stateless. The configuration described here is a small subset used only to demonstrate our example.

Note: A real customer configuration would probably have more environments, more physical servers for redundancy purposes, and so on. A real disaster recovery configuration would also have to take in to account more than just the WebSphere infrastructure; the entire wider environment would need to be replicated correctly. This could also be achieved by using System p LPARs and techniques similar to those described here.

7.4.2 Planning for disaster

Because Server 1 and Server 2 are in separate data centers, Server 2 would be a good candidate for hosting a disaster recovery replica of the production environment served on Server 1. However, this DR environment does not necessarily need to be constantly running and consuming resources. The customer could carry out the following steps:

- ▶ Create extra LPARs with definitions identical to those of the production environment.
- ▶ Install and configure the WebSphere Application Server software on the extra LPARs.
- ▶ Install and configure their applications with correct settings for the Disaster Recovery environment (for instance, JDBC connection strings should point at the DR version of databases).
- ▶ Shut down the newly created LPARs, but leave them available to be restarted.

At this point, the situation would be as depicted in Figure 7-5 on page 346.

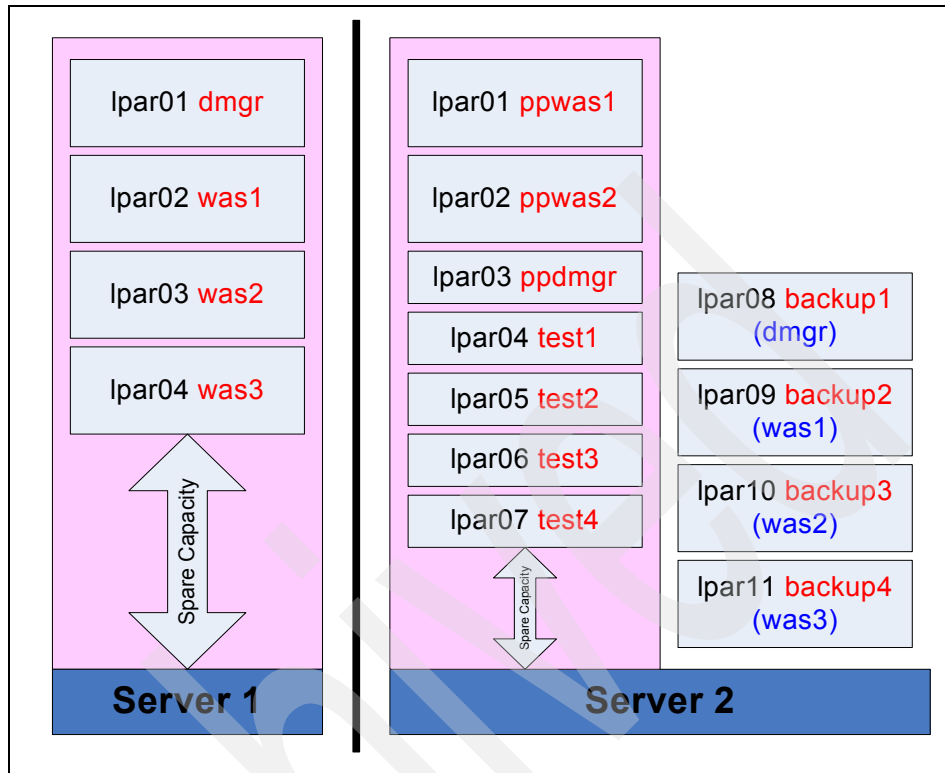


Figure 7-5 Backup LPARs are created for the production environment but not running

The only resource being consumed by the entire DR environment is disk. Server 2 is being actively used as a test and preproduction environment. In the event of a disaster, there is an understanding that these activities will be curtailed or limited.

7.4.3 Disaster event and recovery

In the event of a disaster which makes Server 1 unavailable, the following sequence of events will occur:

- ▶ All other DR tasks will be carried out on relevant required services. For instance, DR databases will be brought up, DR legacy systems will be failed over, and so on.
- ▶ The running LPARs on Server 2 will be shut down in a controlled fashion.
- ▶ The disaster recovery LPARs will be started in a controlled fashion, in the correct order.

After the failover, Server 2 runs all production systems on their respective backup LPARs. After Server 1 is repaired, the systems can be migrated back at a convenient time with a scheduled outage.

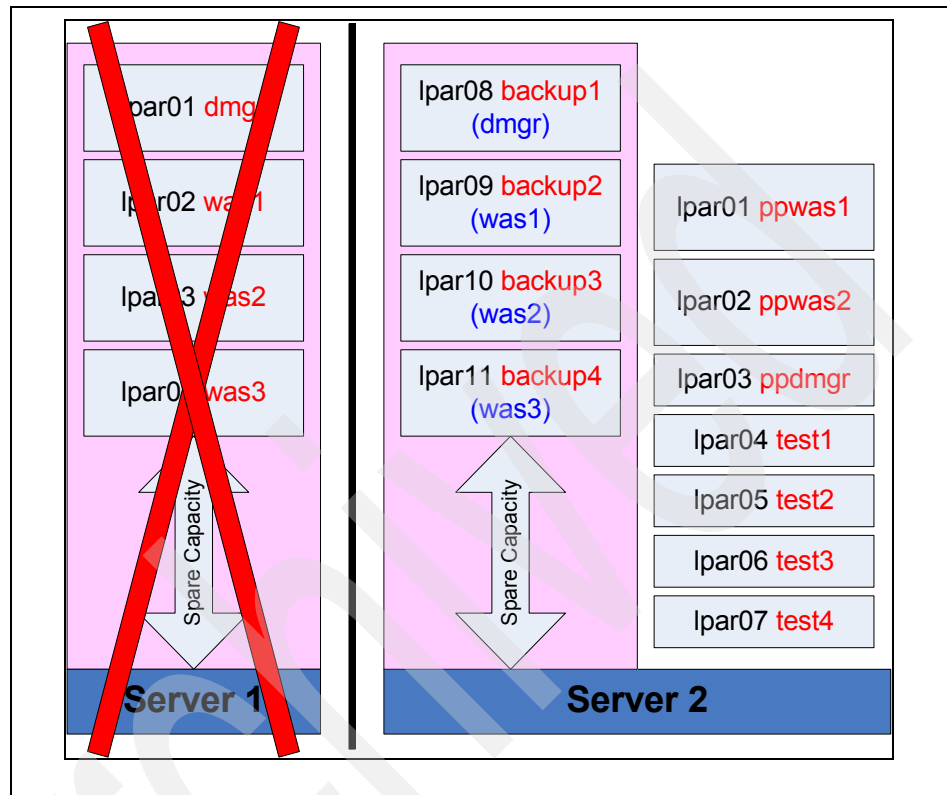


Figure 7-6 The two systems after disaster recovery

7.4.4 Process and procedure

This section only discusses a small part of the considerations required when implementing a disaster recovery strategy, inasmuch as System p can assist in the implementation of that strategy. Everything previously discussed regarding disaster recovery still applies. For a useful reference about making WebSphere Application Server highly available and implementing an overall disaster recovery plan, refer to *WebSphere Application Server Network Deployment V6: High Availability Solutions*, SG24-6688.

A key element of a successful DR plan is to implement known processes and procedures, and test them regularly; the time to test disaster recovery plans should not be during a disaster. In addition, it should become “business as usual”

to update this disaster recovery environment when the production environment is updated, because there is little value in having a disaster recovery environment that is not identical to the production environment.

As part of this readiness, scheduled failovers to the disaster recovery environment should be run every so often, to confirm function and ensure that any recent changes have been implemented and updated. One strategy to ensure that this is the case in our example would be to treat Server 1 and Server 2 not as “primary” and “disaster recovery” environments, but rather for whichever is currently holding the production environment to be considered the real production environment, until the time comes to fail over to the other.

7.5 HACMP

In general, high availability is achieved by making systems redundant. The more system redundancy, the higher the level of availability that can be achieved. IBM HACMP for AIX provides a highly available computing environment by adding software and redundant hardware components. It automatically switches applications and data from one system to another in an HACMP cluster after a hardware or software failure. WebSphere Application Server, coupled with IBM HACMP for AIX, delivers a proven and reliable software portfolio for mission-critical On Demand Business applications.

7.5.1 HACMP, logical partitions and WebSphere

HACMP can be used with logical partitions (LPARs) to fail over application server instances from one server to another in the event of a failure. It can be used to make the Cell Manager and Node Manager administrative processes highly available, as well as the actual application servers themselves that are running customer Java code and processing requests. HACMP can also be used to allocate and release CPU and memory on the LPARs when application servers are started and stopped on the LPARs. Moreover, it can request extra resources from CPUoD if there are not enough system resources to fulfill a request.

This section provides a suggested process for configuring WebSphere Application Server Network Deployment V6 in an HACMP environment. It does not document the complete HACMP setup, including planning, designing, customization, installation, and configuration.

For detailed information about HACMP setup, refer to HACMP documentation *WebSphere Application Server Network Deployment V6: High Availability Solutions*, SG24-6688, and *Implementing High Availability Cluster*

Multi-Processing (HACMP) Cookbook, SG24-6769. This HACMP documentation is available at:

http://www.ibm.com/servers/eServer/pseries/library/hacmp_docs.html)

7.5.2 How HACMP works

There are two main components to the cluster configuration: cluster topology and cluster resources.

The cluster topology describes the underlying framework, that is, the nodes, networks and storage. HACMP uses this framework to keep the other main component, the resources, highly available.

Cluster resources are those components that HACMP can move from node to node (for example, IP addresses, file systems and applications). An “application” in this context is any application that can run on a standalone AIX server; be started and stopped by scripts; and be recovered by a script after an unexpected shutdown. The application and its associated resources is the unit of failover in HACMP.

When a cluster is configured, the cluster topology and resource information is entered on one node. A verification process is then run, and the data is synchronized out to the other nodes defined in the cluster. HACMP keeps this data in its own Object Data Manager (ODM) classes on each node in the cluster.

Although HACMP can be configured or modified from any node in the cluster, it is good practice to perform administrative operations from one node to ensure that HACMP definitions are kept consistent across the cluster, thus preventing a cluster configuration update from multiple nodes, which may result in inconsistent data.

7.5.3 WebSphere HACMP configuration

To configure a pair of LPARs to failover WebSphere processes with HACMP, we need to set up a resource group that knows about the two LPARs, the resources involved (an IP address and shared file system), and the WebSphere Application Server processes themselves. This resource group can then be used to move all resources in the group from a primary machine to a standby machine when the primary machine fails.

The WebSphere Application Server software is installed on a volume group available through the VIO Server to the primary and standby LPARs. When the primary LPAR fails, the WebSphere Application Server configuration on the volume will be mounted to the standby LPAR. The standby LPAR will take over

the IP address of the primary LPAR. The HACMP start script on the standby LPAR will then run each command in the script to start all necessary server processes. During this failover process, the WebSphere Application Server processes on the LPAR are not available to service clients.

During a failure (such as a network or hardware failure), HACMP on the primary machine notifies its peer services on the standby machine through the heartbeat communication. HACMP on the standby machine recognizes the failure event. It takes over the service IP address of the primary machine, mounts the shared file system, and starts all registered servers, such as WebSphere Application Server.

As a simple HACMP configuration for illustration, imagine a cluster configuration consisting of two AIX logical partitions with shared disk storage provided by the VIO Server. The two partitions are connected via an Ethernet-based IP network, over which a heartbeat is performed. WebSphere product binaries are to be installed on the shared file system, and will be used by whichever partition is active at a given moment.

Deployment Manager

The Deployment Manager is commonly regarded as a single point of failure from a systems management point of view in a WebSphere Application Server environment. Note that it is not considered a single point of failure when considering the runtime functionality of the application servers in the cell. Still, making the Deployment Manager available using HACMP is a sensible idea.

Installation

To install the Deployment Manager and create the profile, do the following:

1. Start the HACMP services on the first LPAR and mount the shared file system.
2. Install IBM WebSphere Application Server Network Deployment V6 on the first LPAR. The installation path must be on the shared file system.
3. If necessary, install any needed WebSphere Refresh Packs or Fix Packs. (WebSphere Install Factory can be used to streamline this process.)
4. Create a Deployment Manager profile on the first LPAR. The profile directory should be a subdirectory of the WebSphere <install_root>, which is on the shared file system. If you select a different profile directory, then ensure that it is on the shared file system. When creating the profile, use the virtual host name as the Host name. Do not use the physical host name or physical IP address.
5. Start the Deployment Manager and add nodes to the cell as needed. Use the virtual host name when you specify the Host name or IP address of the Deployment Manager.

Configuration

To configure HACMP to run the Deployment Manager, do the following:

1. Add the Deployment Manager **stop** and **start** commands to the respective HACMP stop (ha.stop) and start (ha.start) scripts on both LPARs. The Deployment Manager stop and start scripts can be quite simple, calling existing scripts to stop and start. For example, the stop script could call as follows:

```
/usr/WebSphere6/AppServer/profiles/Dmgr01/bin/stopManager.sh
```

The start script could call as follows:

```
/usr/WebSphere6/AppServer/profiles/Dmgr01/bin/startManager.sh
```

2. Ensure that the HACMP services are active on the second LPAR.

Testing

Because the Deployment Manager is not required for client access to the application, failover should be completely transparent to clients. Administrative clients may not be able to administer the system while the failure is in place.

It would be prudent, at a minimum, to test the following failover scenarios:

- ▶ Conduct a graceful failover test using the HACMP takeover option. Watch the SystemOut.log of the Deployment Manager for a successful start. Check all node agents to make sure they can synchronize before and after failover. There might be failed synchronization attempts during the failover.
- ▶ Fall back to the first LPAR and conduct a shutdown failover test using **reboot -q** on this LPAR. Watch the SystemOut.log on the Deployment Manager and the node agents for success.
- ▶ End the Deployment Manager process using the **kill** command.
- ▶ Unplugging the network cable of the first LPAR.

Node agent and application server

A node agent process will run on each node that is part of the cell. The node agent will look after application server processes running on that node.

Installation

To install a node agent and create a profile, do the following:

1. Start HACMP services on the first LPAR to mount the shared file system from the VIO Server.
2. Install WebSphere Application Server Network Deployment V6.0 on the first LPAR. The installation path should be on the shared file system.

3. If necessary, install Refresh Packs or Fix Packs. WebSphere Install Factory could (and should) be used to streamline this process.
4. Create a custom profile on the first LPAR. The profile directory should be a subdirectory of the WebSphere <install_root>, which is on the shared file system. If you select a different profile directory, then ensure that it is on the shared file system. When creating the profile, use the virtual host name as the Host name. Do not use the physical host name or physical IP address.
5. Federate the node to the Deployment Manager while creating the custom profile or later using the **addNode** command.
6. Create application servers on the node as desired.

Configuration

To configure HACMP to run the node agents and application servers, add the node agent stop and start commands as well as the appropriate start and stop commands for each application server in the failover unit to the HACMP stop and start scripts. Follow these steps:

1. Add the node agent and application server **stop** commands to the HACMP stop script (ha.stop) on both LPARs. Be aware that the application server names are case sensitive, and make sure that you specify the correct profile directory.

For example, if your node had two clones called svr01 and svr02, then in the profile called svr, the script might contain the following:

```
/usr/WebSphere/AppServer/profiles/svr/bin/stopServer.sh svr01
/usr/WebSphere/AppServer/profiles/svr/bin/stopServer.sh svr02
/usr/WebSphere/AppServer/profiles/svr/bin/stopNode.sh
```

2. Add the node agent and application server **start** commands to the HACMP start script (ha.start) on both LPARs. It is usually prudent to add a **sleep** command after the node agent **start** command, because the application servers depend on an active node agent to be able to start successfully. The length of the sleep will need to be adjusted for your environment, depending on the time the node agent takes to start.

Using the preceding example with a sleep length of 60 seconds, the start script might contain the following:

```
/usr/WebSphere/AppServer/profiles/svr/bin/startNode.sh
sleep 60
/usr/WebSphere/AppServer/profiles/svr/bin/startServer.sh svr01
/usr/WebSphere/AppServer/profiles/svr/bin/startServer.sh svr02
```

3. Ensure that the HACMP services are active on the second LPAR.

Testing

During the failover process, the application servers hosted on the HACMP node cannot serve any client requests. At a minimum, the following two failover scenarios should be tested.

- ▶ Conduct a graceful failover test using the HACMP takeover option. Watch the SystemOut.log on the node agent and each application server for a successful start. Check the node agent to make sure it can synchronize with the Deployment Manager before and after failover. There might be failed synchronization attempts during the failover.
- ▶ Fall back to the first LPAR and conduct a shutdown failover test using **reboot -q** on this LPAR. Watch the SystemOut.log on the node agent and each application server for a successful start.

7.5.4 HACMP on DLPARs and shared CPU LPARs

HACMP can be used to assign and release CPU and memory on logical partitions, and even to request extra resources from CPUoD if there are not enough system resources to fulfill a request. On a shared CPU LPAR, HACMP can also manage virtual processors as needed.

Capabilities

When you configure an LPAR on the HMC (outside of HACMP), you provide LPAR minimum, desired, and maximum values for the number of CPUs and amount of memory. These values can be obtained by running the **lshwres** command on the HMC. The stated minimum values of the resources must be available when an LPAR node starts. If more resources are available in the free pool on the frame, an LPAR can allocate up to the stated desired values.

During dynamic allocation operations, the system does not allow that the values for CPU and memory go below the minimum or above the maximum amounts specified for the LPAR.

HACMP obtains the LPAR minimums and LPAR maximums amounts and uses them to allocate and release CPU and memory when application servers are started and stopped on the LPAR node.

HACMP requests the DLPAR resource allocation on the HMC before the application servers are started, and releases the resources after the application servers are stopped. The Cluster Manager waits for the completion of these events before continuing the event processing in the cluster.

HACMP handles the resource allocation and release for application servers serially, regardless of whether the resource groups are processed in parallel.

This minimizes conflicts between application servers trying to allocate or release the same CPU or memory resources. Therefore, you must carefully configure the cluster to properly handle all CPU and memory requests on an LPAR.

These considerations are important:

- ▶ After HACMP has acquired additional resources for the application server, when the application server moves again to another node, HACMP releases only those resources that are no longer necessary to support this application on the node.
- ▶ HACMP does not start and stop LPAR nodes.

It is possible to create a custom event or customize application start and stop scripts to stop LPAR nodes if desired.

The granularity of what HACMP can manage is the physical CPU for dedicated processors, and virtual processors for shared processors configuration. With micropartitioning, HACMP works with virtual processors, and not physical processors. In a mixed environment, HACMP does the verification of the possibility to add the resources to respect the maximum value.

For dedicated processor mode, the maximum value unit is the physical processor. For shared processor mode, the maximum value unit is the virtual processor. The free pool resources is calculated by adding the desired capacity entitlement (CE) values of each partition in shared processor mode, and the physical processor in dedicated processor mode.

Configuration

To configure HACMP with DLPAR, the following high level steps need to be carried out:

- ▶ Configure consistent name resolution.
- ▶ Install ssh on the HACMP nodes.
- ▶ Configure the HMC for ssh access.
- ▶ Define HMC and managed systems to HACMP.
- ▶ Define the DLPAR resources to HACMP (that is, application provisioning).

Refer to *Implementing High Availability Cluster Multi-Processing (HACMP) Cookbook*, SG24-6769, for more information about configuration.

7.6 Lifecycle management and upgrades

Maintaining availability while upgrading applications, middleware, and operating systems is one of the greatest challenges of modern distributed computing

systems. When nonfunctional requirements state that applications need to be available 24/7 and unscheduled outages cost thousands of dollars per second, it is absolutely critical that processes and procedures be available that enable systems to remain available and fully functional while being updated.

7.6.1 The problem

Some sites simply cannot afford any downtime. Even among those that can, when an application is running in production, it is often difficult to schedule any downtime for updates, whether these are application updates, application server updates, operating system updates, hardware updates or any other kind of change to the system. Compounding this reality is the fact that emergency fixes (such as a newly discovered and fixed security flaw) must be applied as soon as possible for the sake of system integrity, and these often need to bypass the downtime schedule process entirely.

Even when downtime is scheduled to perform an update, there is the issue of rollback. If, despite all of the extremely thorough testing that happens before production, an issue is discovered on the production environment which means that the update needs to be rolled back, how exactly do you do this? Sometimes customers perform a full backup after taking the system down, and this is used as their rollback solution; if something goes wrong, the system is taken down again and the backup is restored. This may work, but again involves downtime, and also precludes the possibility of investigating the “failing” system.

7.6.2 The solution

The ideal solution is to have unlimited hardware. Traditionally, the recommendation has been to set up an identical copy of the live environment on separate machines, and configure it so that it is ready to function. Then, at the lowest possible usage point (to ensure minimum disruption if something *does* go wrong), use the network load balancer to send all new requests to the new version of the production environment. All old requests should still be served by the old environment, to ensure consistency, although this may not be relevant for all application types (the suggestions here need to be taken in conjunction with a deep understanding of your particular environment).

After a while, no more requests will be sent to the “old” live environment, and the new one will be servicing all of the users. After a suitable period has elapsed and we are confident that the new environment is functioning correctly, the old environment can be turned off and the hardware reappropriated.

7.6.3 Unlimited hardware

Of course, not everyone has unlimited hardware. Many customers resist the prospect of purchasing extra hardware for the purposes of disaster recovery, let alone buying *another* identical copy of their production environment for upgrading purposes.

Fortunately, with System p and logical partitioning, it is very possible to carry out this process using your existing hardware and logical partitioning. The following high level steps should be used as a guide to carrying out an application server upgrade, making it relevant to your particular environment. Similar steps could be used for operating system upgrades and even major application upgrades.

For our example, we use a very simple environment (although this sequence is possible even for complex environments), as shown in Figure 7-7.

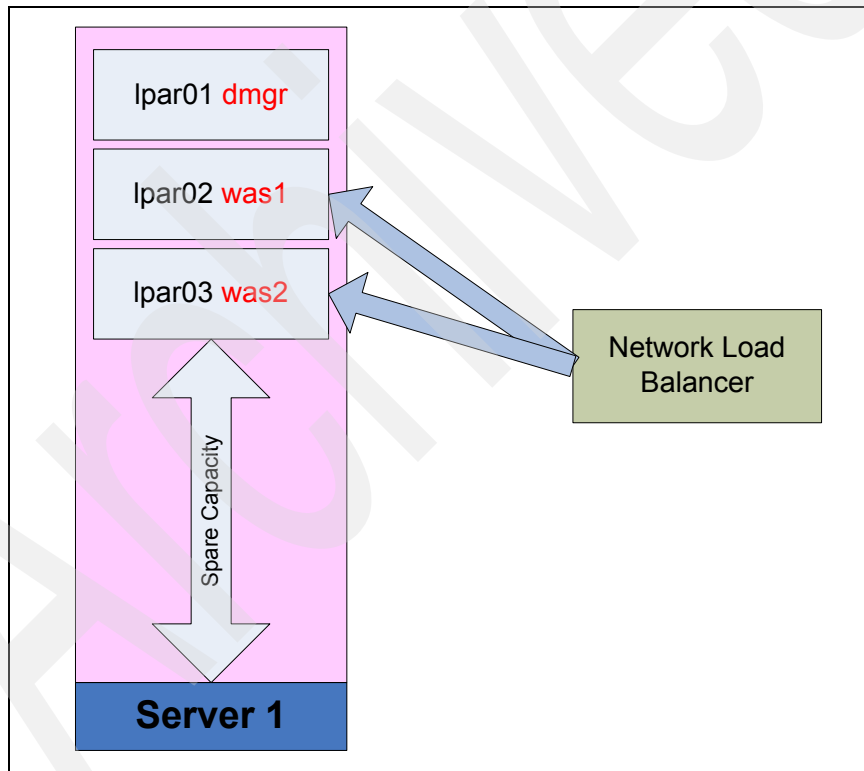


Figure 7-7 A simple production environment which needs upgrading

Consider the following procedure:

1. Create an identical copy of the production environment; see Figure 7-8 on page 357.

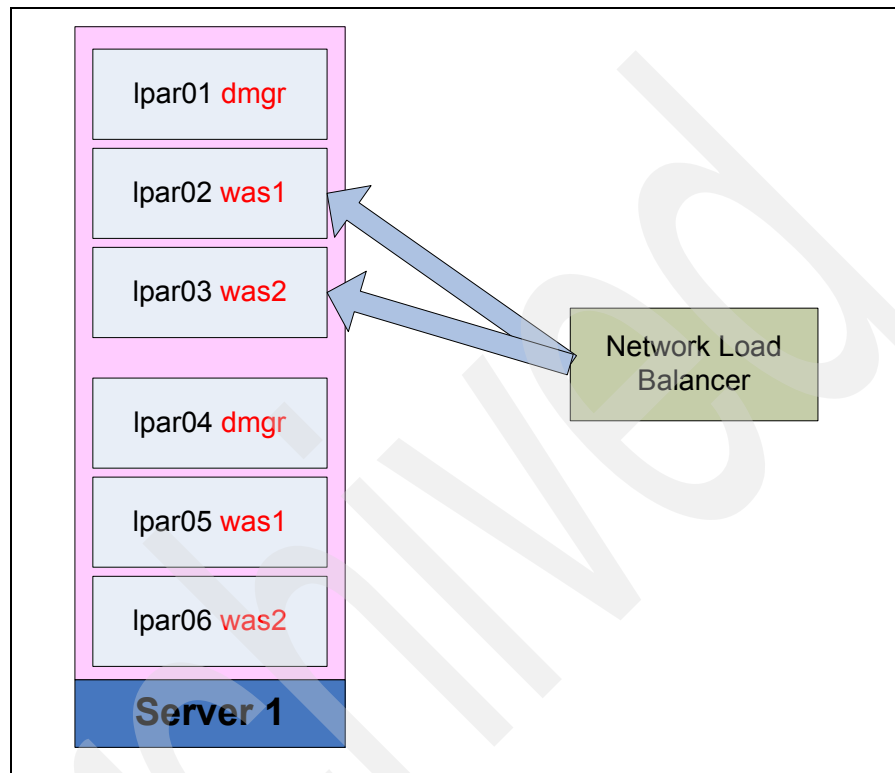


Figure 7-8 Create a new copy of the production environment

2. At a convenient time, when the system is least used, configure the load balancer to route new requests to the new systems; see Figure 7-9 on page 358.

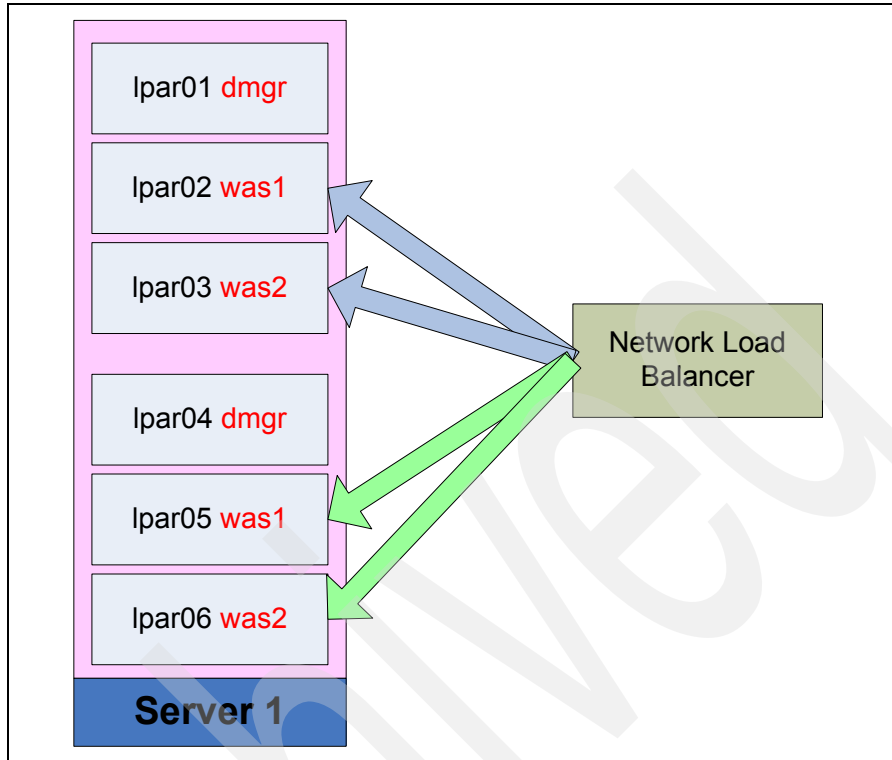


Figure 7-9 Configure new requests to be routed to the new environment

3. After all requests to the old environment have completed, the new environment will be serving all requests and the old environment will be available for rollback as needed; see Figure 7-10 on page 359.

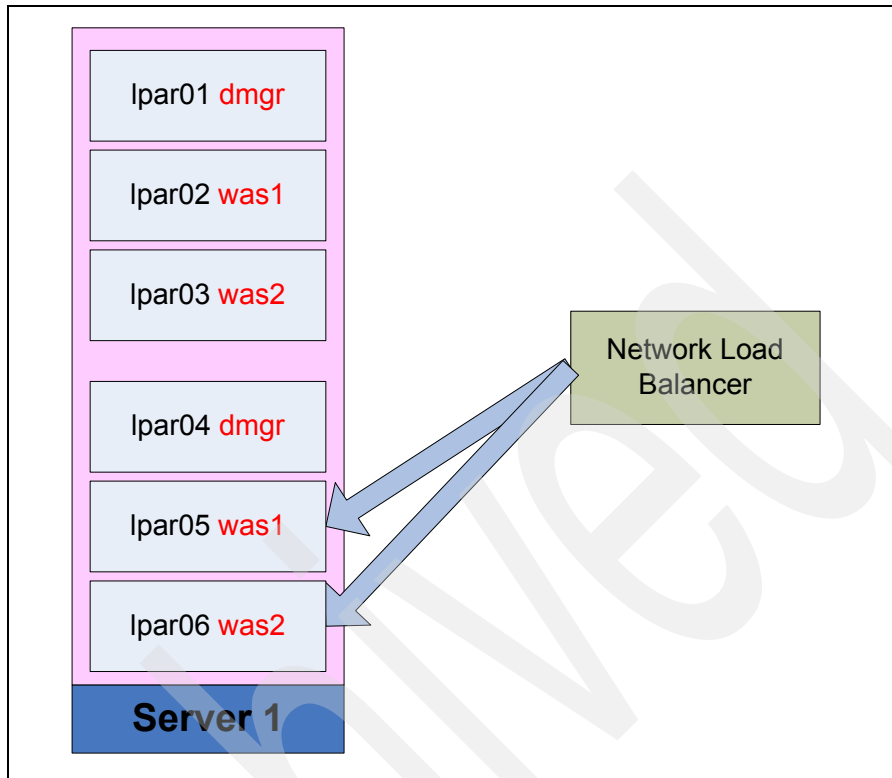


Figure 7-10 The new environment is serving all requests

4. After a suitable period of time has elapsed, the LPARs for the old environment can be archived and taken offline; see Figure 7-11 on page 360.

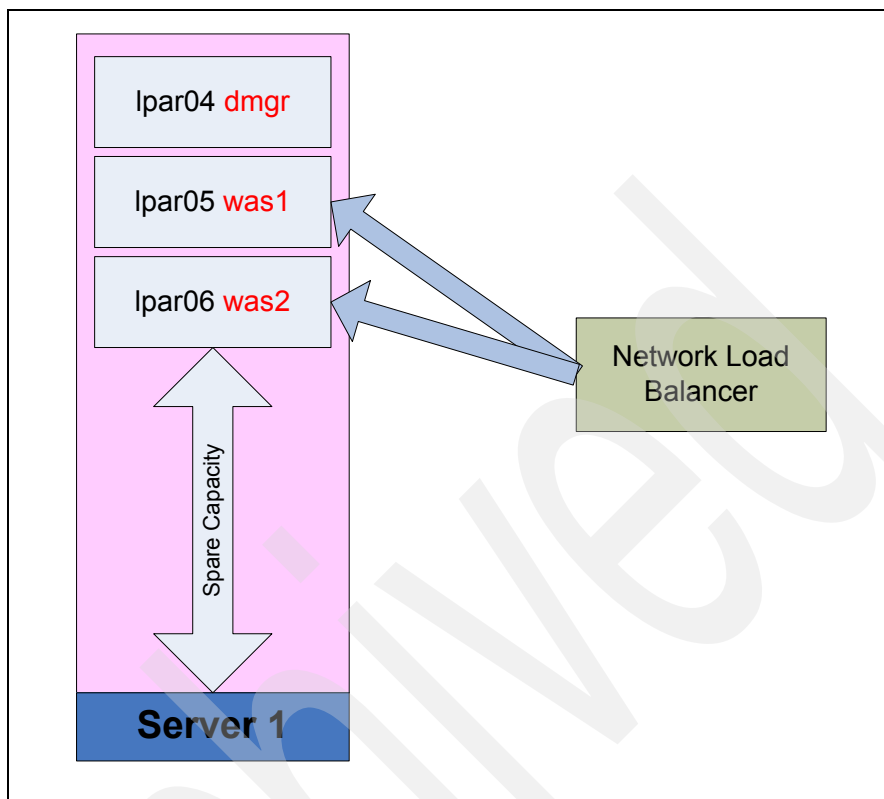


Figure 7-11 The new environment is live

Clustering WebSphere Application Server for performance

Clustering WebSphere Application Server serves two purposes, namely availability and performance. This chapter describes methods for implementing highly scalable and performant systems using various WebSphere and AIX capabilities.

8.1 Clustering for performance overview

The term “clustering” in regard to performance refers to scaling multiple copies of an application so it is able to service more requests than a single copy of the application would be.

Making an entire system performant means tuning, “tweaking” and finding the bottlenecks in all of the supporting infrastructure and back-end connections. WebSphere may often be blamed for performance issues that are in fact caused by completely separate systems.

This chapter focuses on methods of scaling WebSphere Application Server to improve performance by exploiting available resources. It specifically does *not* cover tuning any other part of the infrastructure on which the system relies, and which may cause the system as a whole to underperform.

8.1.1 Clustering options

To cluster for performance, we discuss three different workload management mechanisms, as summarized in Table 8-1.

Table 8-1 Workload management mechanisms for advanced POWER virtualization

Shared processor, uncapped partitions	The POWER Hypervisor™ allocates unused processor cycles to those uncapped partitions that can make use of them.
Workload Manager (WLM)	Prioritizes applications' access to system resources: CPU, memory, and I/O within a partition.
Partition Load Manager (PLM)	Adds and moves CPU and memory resources to and between partitions using dynamic LPAR operations.

8.2 WebSphere clustering on micropartitions

IBM WebSphere Application Server Network Deployment V6 workload management optimizes the distribution of incoming requests between application servers that are able to handle a client request. WebSphere workload management is based on application server clusters containing multiple application servers, so-called cluster members. An application deployed to a cluster runs on all cluster members concurrently. The workload is distributed

based on weights that are assigned to each cluster member. Thus, more powerful machines receive more requests than smaller systems.

The simplest way to increase the throughput of a well-tuned and efficient WebSphere application is to give it more resources. In the traditional WebSphere world, this was accomplished in the following three ways.

Put the application onto a bigger machine

Instead of a dual CPU, this involves moving to a four-way box and instead of using Gb of memory, giving it 2 Gb of memory. Depending on the application, this may or may not be the right approach.

Create a vertical cluster

This involves having another instance of the application running on the same machine. Applications that require large amounts of memory may be limited in the amount of Java heap space they can use; in these instances, it is more efficient to have a second Java virtual machine to provide the extra memory.

Create a horizontal cluster

This involves having another machine running another instance of the application.

The same holds true in the world of logical partitions and virtualization, except that it is much easier to implement. To put the application onto a bigger machine, all that is needed is to assign more CPU or memory to the LPAR. To create a horizontal cluster, you simply need to create another LPAR and assign it some resources. With the clever use of scripting, you can automate much of this work and have extra performance when needed, with little effort.

In the world of the AIX LPARs, it is easy to add new resources to a WebSphere cluster. In our scenario, under “Dynamic testing” on page 420, we demonstrate the effect that dynamically adding resources has on an application under significant load. This is significantly easier than configuring new machines into a cell or migrating to bigger machines, and has the added advantage that the overall environment is easier to manage.

8.2.1 Isolation

As mentioned in 7.1, “Clustering WebSphere for availability” on page 336, a System p LPAR is a natural home for a WebSphere node. A single physical server can host many nodes, with the added advantage that each node is a separate installation of the operating system and is isolated from the others. This has a significant impact on the ability of the system to be able to provide different

levels of service to different applications with differing non-functional requirements.

For example, if one application has a requirement that it must respond to requests within (say) two seconds, and another application does not have such a requirement, it makes sense that in a heavy load situation, the application with no specific requirement is not allowed to take resources away from the one with a two-second requirement. By simply placing them on separate LPARs and setting hard limits on the maximum amount of resources available to the relevant partition, it is possible to arrange it so that the two-second rule cannot be degraded by the consumption of the application without such a requirement.

8.3 AIX Partition Load Manager

The Partition Load Manager (PLM) for AIX 5L is a method of dynamically distributing load between logical partitions. PLM is part of the Advanced POWER Virtualization feature. It is supported on both dedicated and shared-processor partitions running AIX 5L V5.2 (ML4) or AIX 5L V5.3.0 or later on IBM System p5 servers.

The Partition Load Manager is designed to automate the administration of memory and CPU resources across logical partitions. To improve system resource usage, PLM automates the migration of resources between partitions based on partition load and priorities. That is, partitions with a high demand will receive resources donated by or taken from partitions with a lower demand. A user-defined policy governs how resources are moved. PLM will not contradict the partition definitions in the HMC. PLM allows administrators to monitor resource usage across all managed partitions.

8.3.1 PLM capabilities

PLM uses a client/server model to monitor and manage partition resources. The clients act as agents on each of the managed partitions. The PLM server configures each of the agents (clients), setting the thresholds at which the server should be notified. The agents monitor the partition's resource usage and notify the PLM server whenever PLM-set thresholds are passed (underutilized or overutilized). Based on a user-defined resource management policy, the PLM server invokes dynamic reconfiguration (DR) operations through the HMC to move resources from a spare pool to a partition or between partitions.

PLM allows for groups of partitions. Resources within a group are managed independently. Because each partition is monitored locally and the agents only

communicate with the PLM server when an event occurs, PLM consumes a negligible amount of system and network resources.

8.3.2 AIX PLM configuration

In the following sections we describe PLM configuration in more detail.

Resource management policies

The resource management policy for each partition group is specified in a policy file that defines both the managed environment and the parameters of the resource management policy.

For each resource, there is an upper and lower load threshold. Every time a threshold is crossed, PLM receives a Resource Management and Control (RMC) event. When the load on the resource is above the upper threshold, the partition PLM considers the partition in need of additional resources; the partition is said to be a *requestor*. When the load on the resource is below the lower threshold, the partition becomes a potential *donor*. Normally, resources are only removed from donors when another partition enters the requestor state for the same resource. When the load on the resource is between the two thresholds, PLM considers that the resources available are adequate.

The policies are defined by a policy file, which can be created using a wizard or by manually editing a file. If you edit the PLM policy by hand, be advised that the file has a strict, stanza-based structure. If this structure is not respected, then PLM will not be able to use it. Make a copy of a known “good” file and only edit the copy. You can use the `xlplm -C -p policy_file` command, where `policy_file` is the manually edited file to check the syntax of the policy file.

The basic steps to define a PLM policy are:

1. Create a new policy file.
2. Define the global environment and, optionally, the global tunables.
3. Define the partition groups and, optionally, the group tunables.
4. Add partitions to the groups and, optionally, define the partition tunables.

Tunables

The parameters which can be configured in a PLM policy are listed in Table 8-2.

Table 8-2 Tunable parameters for Partition Load Management

Parameter	Description
Memory minimum	The minimum memory capacity of the partition. PLM will never leave a partition with less than this amount.

Parameter	Description
Memory guaranteed	The guaranteed amount of memory.
Memory maximum	The maximum amount of memory resources PLM will allow a partition to have.
Memory shares	A factor used to specify how memory capacity in excess of the memory guaranteed is distributed to partitions in the group. Specifying a value of 0 indicates that a partition will never receive more than the guaranteed memory.
CPU minimum	The minimum CPU capacity of the partition. PLM will never leave a partition with less than this amount.
CPU guaranteed	The guaranteed amount of CPU whenever a partition's load is greater than the CPU load average low threshold.
CPU maximum	The maximum amount of CPU resources PLM will allow a partition to have.
CPU Shares	A factor used to specify how CPU capacity in excess of the CPU guaranteed is distributed to partitions in the group. Specifying a value of 0 indicates that a partition will never receive more than the guaranteed CPU.
CPU notify intervals	The number of contiguous 10-second sample periods that a CPU-related sample must cross a threshold before PLM will initiate any action.
CPU load average high threshold	The processor load average high threshold value. A partition with a load average above this value is considered to need more processor capacity (requestor).
CPU load average low threshold	The CPU load average low threshold value. A partition with a load average below this value is considered to have unneeded CPU capacity (donor).
Immediate release of free CPU	This indicates whether or not unused excess processor capacity will be removed from the partition and placed in the shared processor pool. A value of no indicates unneeded CPU capacity remains in the partition until another partition has a need for it. A value of yes indicates unneeded CPU capacity is removed from the partition when the partition no longer has a need for it.

Parameter	Description
Entitled capacity delta (<i>shared partitions only</i>)	The percentage increase of CPU entitled capacity to add or remove from a shared processor partition. The value specifies the percent of the partition's current entitled capacity to add or remove.
Minimum entitlement per VP (<i>shared partitions only</i>)	The minimum amount of entitled capacity per virtual processor. This attribute prevents a partition from having degraded performance by having too many virtual processors relative to its entitled capacity. When entitled capacity is removed from a partition, virtual processors will also be removed if the amount of entitled capacity for each virtual processor falls below this number.
Maximum entitlement per VP (<i>shared partitions only</i>)	The maximum amount of entitled capacity per virtual processor. This attribute controls the amount of available capacity that may be used by an uncapped shared processor partition. When entitled capacity is added to a partition, virtual processors will be added if the amount of the entitled capacity for each virtual processor goes above this number. Increasing the number of virtual processors in an uncapped partition allows the partition to use more of the available processor capacity.
Memory notify intervals	The number of contiguous 10-second sample periods that a memory-related sample must cross a threshold of before PLM will initiate any action.
Memory utilization low	The memory utilization low threshold below which a partition is considered to have excess memory and will become a memory donor. The units are in percent. The minimum delta between memory_util_low and memory_util_high is 10 percent.
Memory utilization high	The memory utilization high threshold at which a partition is considered to need more memory. Units are in percent. The minimum delta between memory_util_low and memory_util_high is 10 percent.
Memory page steal high	The page steal rate threshold at which a partition is considered to need more memory. Units are page steals per second. The result of checking this threshold is logically AND'd with the result of the memory utilization high threshold when determining if additional memory is needed by a partition.

Parameter	Description
Memory free unused	Indicates whether or not excess memory capacity will be removed from the partition and placed in the spare memory pool. A value of no indicates unneeded memory capacity remains in the partition until another partition has a need for it. A value of yes indicates unneeded memory capacity is removed from the partition when the partition is no longer using it.
Memory delta	The number of megabytes of memory to be removed or added to a partition in any single DR operation. If the value is less than the system's logical memory block (LMB) size, the value is rounded up to the system's LMB size. If the value is greater than the system's LMB size but not a multiple of it, the value is rounded down to the nearest multiple of LMB size.

For more information on configuring Partition Load Management, see *PowerVM Virtualization on IBM System p: Introduction and Configuration Fourth Edition*, SG24-7940.

8.4 AIX Workload Management feature

As described, it is easy to partition work across a System p machine by creating a new LPAR for each application, and using virtualization or PLM to assign appropriate resources to each. However, this coarse granularity may not be desired, especially when running many smaller well-tuned and efficient applications. Some customers may prefer to deploy several application servers into a single LPAR, and might still want to apply different priorities and resources to the different applications.

This section discusses the Workload Management feature in AIX, which is different to the Workload Management features of WebSphere. AIX Workload Management can manage a WebSphere service as a Java process, where different applications execute in separate application servers' JVMs sharing one LPAR.

8.4.1 WLM capabilities

WLM is an integral part of the AIX kernel, always installed but not enabled by default. WLM is a feature of AIX 5.1 Maintenance level 3 or AIX 5.2. There is a restricted form in AIX 4.3.3 maintenance level. It effectively modifies the thread dispatching by using recalculated process priorities instead of using static

priorities assigned by the system. These priorities are calculated from current usage for all processes every second, broken down by WLM classes. WLM configuration assigns different shares of system resources to each class. It calculates the priorities to correct imbalances between desired and achieved resource share.

AIX WLM can control processor, memory and disk I/O bandwidth. It does not control network bandwidth. It is possible, and popular, to restrict WLM to controlling processor use alone.

WLM can run in “passive” mode, where it records resource utilization, but does not control it, that is, run in “active” mode, where it adjusts the priorities to desired resource utilization.

8.4.2 Co-locating WebSphere applications

Each WebSphere application is different, and will have different requirements regarding heap size and overall process memory. If a customer's WebSphere applications after performance tuning and load testing do not require much memory to process efficiently, then performance gains may be seen by co-locating them on the same LPAR.

WebSphere Application Server runs as several processes, all executing the same Java runtime binary and probably under the same user ID (although this can be changed). A small, well-tuned and efficient low volume application might use 256-756 Mb of memory per process that should not be paged, and probably 50-250 threads, maybe more. This means that an efficient low-volume Web application might need less than 25% of a processor, and as little as 128 Mb memory. Managing such requirements are well within the AIX workload management scope.

These processes may run independent applications funded by different businesses. Some applications may need more than one process (JVM) to achieve the desired capacity, in which case they will be cloned JVMs. Clones form a group, called a *cluster*. All processes in a cluster have the same configuration and run the same application. There is also an administrative process (actually another process just like the others, but running a different application) performing shared services for the other processes under its administration. The SLA and the resource plans are at the application level, so you need to manage the application as the aggregation of these JVM processes.

8.4.3 AIX WLM configuration

In the following sections, we discuss AIX WLM configuration considerations.

Superclasses and subclasses

AIX WLM aggregates processes into superclasses and subclasses of those superclasses. There can be up to 27 user-defined superclasses, and up to 10 subclasses within each. WLM enables a delegated authority to configure the subclasses within each superclass.

WLM can automatically assign processes to superclasses and subclasses using rules examining the owning user ID, binary executable, and other parameters. Alternatively, the **wmlassign** command manually assigns processes to particular classes. Given the appropriate authorization, a startup script can assign its own process to the appropriate class. A process may be assigned to only a superclass, in which case it is automatically assigned to the “default” subclass; or it might be assigned to a specific subclass within the superclass. Example 8-1 illustrates class assignment.

Example 8-1 Assigning classes

```
WebSphere assigned for /usr/WebSphere/AppServer/java/bin/java
WebSphere.Mortgages subclass manually assigned
WebSphere.AccountOpening subclass manually assigned
```

WLM collects processor, memory, and disk I/O bandwidth use by class. It could assign processes running the WebSphere Java binary executable to a WebSphere supergroup. That would allow for up to 10 subgroups, each a different application.

Shares

In active mode, WLM reads a configuration file that assigns each class shares of the system resources. The class gets its resource in proportion of its share to the total. For example, assume there are two classes:

```
WebSphere 300
Tape Backup 150
```

The WebSphere class would get $300/(300+150) = 66.7\%$ of the resource, and Tape Backup would get 33.3% . If later another application, Billing, was also assigned 150 shares, the split becomes $WebSphere\ 300/(300+150+150) = 50\%$, $Tape\ Backup\ and\ Billing = 25\%$.

Shares are not percentages; they are *relative* amounts of the resource. In the preceding example, if the resource was processor, and the LPAR had three processors, the first shares would be WebSphere 200% and Tape Backup 100% of a processor. These correspond, not to dedicated processors, but to the total processor time allocated across all three. If later a fourth processor was added

before the Billing application, the shares increase to WebSphere 266.7%, Tape Backup 133.3%.

Subclasses also have shares, but in this case they refer to the superclass's resource, not the total resource. So assume the first example was as follows:

```
WebSphere.Mortgages 200
WebSphere.Account Opening 300
```

Mortgages would get $200/(200+300) = 40\%$ of 66.7% = 26.7% of the processor resource, and Account Opening would get 40%. With three processors, that would be 80% and 120% of a processor, respectively.

These shares are not limits; processes can use more than their fair share if no other applications want them. But where there is contention, the available resource is given first to processes below their fair share. Further, if a class has no active processes, it is completely ignored in the calculation. Assume an example with:

```
WebSphere 300
Tape Backup 150
Billing 150
```

If Tape Backup were inactive, WebSphere would get 66.7% and Billing would get 33.3% of the available resource. WLM also supports limits, rather than shares (which are discussed in “Shares” on page 370).

Tiers

Subclasses can be organized into numbered tiers, so that the higher-numbered tiers only obtain resources after all lower-numbered tiers are satisfied. Thus system processes and critical services may be in tier 0, with normal applications in tier 1, and background processes in tier 2.

For a WebSphere service there may be some Tier 2 processes, such as log maintenance, but most would be Tier 1.

Limits

Processor resources may have minimums, soft maximums and hard maximums, as well as shares. Limits take precedence over shares. A class with a minimum of 10% processor resource will be permitted to take that much resource, even if the shares calculation produces a lower percentage. If it produces a higher value, that will be honored. Thus, a minimum can be used to ensure that an application obtains a minimum service level even if many other applications start.

The soft maximum can be breached if there is no contention for resource, but the hard maximum cannot be breached (at least, over any significant time period,

because WLM controls allow short-term peaks in the order of a very few seconds).

The effect of the WLM algorithm can be approximately described as follows:

```
if used < minimum, increase priority;
else if used > hard maximum, decrease priority;
else if used > soft maximum during contention, decrease priority;
else if used < share, increase priority;
else if used > share, decrease priority.
```

Memory minimums may reduce the danger of application server JVMs being paged. A process requesting less memory than the minimum does not prevent other processes using the difference, but if the first process then requests more, memory will be taken from those other processes, thereby causing paging.

A processor maximum may prevent one application server from monopolizing resources at the expense of another. It can limit its performance to its budgeted resources, rather than giving unnaturally good performance when the LPAR is undercommitted.

Limits are expressed as a percentage of the LPAR's resource. For example, on a 4-processor LPAR, they still range from 0 to 100% (of those 4 processors in aggregate), not 0-400%.

Minimums at the subclass level must total no more than 100%, and apply to the superclass resource. Minimums at the superclass level must total no more than 100%, and apply to the system. Minimums must not exceed the soft maximum, which in turn must not exceed the hard maximum for the same resource. The defaults are 0%, 100%, 100%, so that WLM allocates resource according to shares alone.

On an LPAR dedicated to WebSphere applications, the WebSphere superclass might have a minimum and soft maximum of 85% processor, the subclasses allocated according to their budget share of that. For example, three applications of equal budget might have a hard maximum of 33%.

Allocating actual measures of LPAR resource, for example to ensure that one application has at least 512 Mb memory, needs a program or shell script to calculate the appropriate percentages of the available resources. With static LPARs, this program would need running only at startup. With dynamic LPARs, it would need running after each reconfiguration, or at regular intervals.

Resource sets

WLM can allocate resources in named resource sets. A *resource set* is one or more processors or an amount of memory. WLM classes can be tied to a

particular resource set. This allows a very simple allocation of resource that sometimes may improve performance by optimizing the hardware cache hits, but it offers very coarse granularity.

Unexpected workload changes

During a disaster recovery situation, workload that was spread evenly across two LPARS may suddenly be transferred to just one of them.

If an entire LPAR fails, all the applications will be transferred to the other. If the WLM configuration is mainly controlled by shares, the resources should continue to be allocated as before. However, if only one application transfers from one LPAR, it is important to reconfigure the other LPAR for the increased workload. Otherwise, WLM may allocate a large enough share of the resources for the doubled workload.

This implies that each application needs different a workload management configuration when running in a single LPAR.

8.4.4 Options for WebSphere service workload management

Workload management offers several interacting sophisticated controls over resource allocation. This allows for subtle and powerful interaction between them. Experience shows that complex rule sets lead to unexpected problems.

When starting a new service, it is important to generate a baseline from realistic tests to determine effects of any proposed rule set. In the absence of a real workload, experiments with benchmarks such as IBM's Trade 6 suite can provide this baseline.

IBM recommendations

IBM recommends starting by classification, and running WLM in passive mode to collect the resource use profile. It is important to run the `wlmcheck` program to validate the configuration.

Having that data, the next steps are assigning small changes to the processor shares, and then monitoring the effects. This stage can be repeated until the desired control is obtained. It allows for rollback if the rules cause service problems.

WLM has a special mode of operation that ignores memory and disk bandwidth, which may suffice for a WebSphere Application Server service with ample memory. However, if the total demand for memory from background tasks and WebSphere Application Server exceeds what is available, then memory

minimums may be worthwhile. The use of memory maximums is not recommended, as they can cause paging even if the LPAR is undercommitted.

As described in 8.4, “AIX Workload Management feature” on page 368, you can define several WebSphere superclasses, and within them up to 10 application subclasses. IBM recommends keeping the number of different classes relatively small; that is, use a few application classes over and above the supplied System and Default classes.

Resources

This section discusses WLM resources in more detail.

Processor

Hard limits on processors ensure that applications do not monopolize resources. Although allocating applications to specific processor resource sets is also effective, this approach is somewhat inflexible and may inhibit performance for multi-threading applications like WebSphere.

Having specified limits for maximal workloads and shares that reflect each application’s expected workload provides fair performance at lower levels, assuming constant incoming workloads.

Memory

Paging WebSphere Application Server JVMs significantly damages performance. You can reduce this by specifying minimum memory percentages or memory resource sets.

You must ensure the allocation allows non-WebSphere Application Server applications sufficient memory. If memory is overcommitted, unmanaged applications or even the system bear the brunt of paging, which can trigger thrashing.

Disk

WebSphere applications rarely monopolize disk I/O bandwidth. Most data access uses enterprise services such as databases and J2C connectors, which generate network I/O. AIX WLM does not control network bandwidth.

If one application monopolizes I/O bandwidth (for example, excessive tracing to System.out), then I/O shares or limits might prove useful, but their effect is to make it look like a slow I/O subsystem.

Automation support

As discussed, several changes would facilitate automatic WLM operation, as described here.

Changes to startServer.sh

WebSphere application servers have a unique server name within a domain or cell. The generic WebSphere architecture specifies that an LPAR is in only one domain or cell, so the server name is a unique identifier within the LPAR. You could modify startServer.sh script to issue the **wlmassign** command to set the Java process WLM class and subclass.

Extensions

As described in 2.1.2, “Workload management” on page 15, WLM configurations are specified in terms of relative resource shares, not absolute values. A script could convert from absolute units such as percentage processor use and Mb of memory to shares.

Failover configurations

As described, fallback operation may require changes to the WLM configuration, when workload is transferred to a single LPAR. This could be handled by generating multiple WLM configurations using WARM, or a separate script to reconfigure WLM for a specific fallback situation.

8.5 Sharing the technologies

WebSphere clustering works well in scaling for performance in conjunction with the advanced power virtualization features of shared CPU partitions, partition load management and work load management, as described.

All of these workload management mechanisms have two things in common:

- ▶ The workload and resource management mechanisms only take effect when there is a shortage of system resources and there is competition for these resources. This means that they will only shift resources when the system resources are insufficient to meet the requirements of the workload and there are two or more active workloads. When the system is normally loaded or when there is only one resource consumer, then the resource managers do not have a significant effect even if they intervene.
- ▶ All workload and resource management mechanisms require an explicit policy describing the (relative) priorities of the managed components, which means identifying those components that will be penalized and those that will be given an advantage whenever there is a resource shortage.

WLM manages resource usage conflicts within a partition. If the partition is known to have sufficient resources for the workload, or there is only one application running in the partition, there will be no competition for resources and

WLM will have no role. WLM is most often used when consolidating several different applications on a single AIX 5L server or partition.

Shared-processor partitions and the POWER Hypervisor can move CPU resources almost instantaneously from one partition to another. Shared processor partitions are appropriate when there is marked, short-term fluctuation in the workload when consolidation in to a single AIX 5L partition is inappropriate.

Shared processor partitions can be used when a partition requires a fractional part of a POWER5 processor (for example, two partitions, each with an entitlement of 1.5 running on three POWER5 processors in a shared pool).

Partition Load Manager has a relatively long latency and cannot manage workload peaks that are of short duration. PLM manages the medium-term and long-term trends; it can handle the necessary migration of resources as operations move from the daily transactions to the overnight batch and back again.

PLM can be used to optimally configure servers with stable workloads. By setting the initial partition configuration with minimal resources, leaving unassigned resources in the free pool (memory and processor), PLM will move just enough resources (assuming that they are available) in to each partition to satisfy the workload. This alleviates the need to perform any precise estimation regarding the distribution of the server hardware between partitions.

This chapter details suggestions for AIX WLM with WebSphere, in a specific scenario (where there is a desire to share LPARs among a number of WebSphere applications). For more information about combining the forms of workload management, refer to *Introduction to pSeries Provisioning*, SG24-6389, and *PowerVM Virtualization on IBM System p: Introduction and Configuration Fourth Edition*, SG24-7940.

AIX 5L and WebSphere XD

This chapter describes how WebSphere Extended Deployment (XD) can be optimized in an AIX 5L environment. It provides a brief discussion of dynamic logical partitioning and the system resource utilization efficiency that these two products can achieve. It provides a brief discussion of dynamic logical partitioning and the system resource utilization efficiency that these two products can achieve on IBM POWER systems.

The five autonomic managers that are functional parts of WebSphere Extended Deployment are also discussed. Finally, processor and memory dynamic reconfiguration events (DR events) are covered, including how DR events can be used to integrate with dynamic logical partitioning (DLPAR).

The chapter also examines the dynamic behavior of WebSphere XD and explains how, when there are changes to AIX 5L memory or CPU resources, WebSphere XD can respond by aligning its own resource usage in tandem with a resource change in a partition that is running the AIX 5L operating system.

9.1 WebSphere XD and DLPAR integration

The AIX 5L operating system has new features to improve server manageability. A key feature is the ability to divide a system into logical partitions (LPARs), which can be resized on demand. You can easily add or remove physical memory (RAM), processors (CPU), and I/O devices from the partitions. A logical extension of this feature is the ability to move these resources among the partitions without having to reboot a partition or a system. This feature is called dynamic logical partitioning (DLPAR).

To leverage these new features, WebSphere XD dynamically modifies its own resource consumption for scalability and improved resource utilization. WebSphere XD can monitor and optimize their application server environments or make recommendations based on the observed data. This capability is referred to as *dynamic operations*.

9.1.1 Dynamic operations

Dynamic operations consists of autonomic managers that unify the infrastructure to maximize utilization through defined business goals. These autonomic managers monitor performance metrics, analyze the monitored data, offer a plan for running various actions, and can start these actions in response to the flow of work.

WebSphere Extended Deployment offers the following autonomic managers as part of the functions of dynamic operations:

- ▶ Autonomic request flow manager
This controls the order of requests into the application server tier and the rate of request flows. Using classification and the defined service goals, the autonomic request flow manager decides how and when to dispatch HTTP requests to the next tier.
- ▶ Dynamic workload manager (DWLM)
This performs load balancing across available application servers. In particular, for a given request flow, DWLM balances requests across the available nodes to regulate response times. DWLM dynamically updates the application status as the application placement controller modifies the infrastructure for the application that is running.
- ▶ Application placement controller
This creates and removes application instances to manage HTTP requests. The application placement controller can dynamically address periods of

intense workflow that would otherwise require the manual intervention of a system administrator.

► Health management

This maintains a robust application server environment through the use of health policies to identify the criteria that require action. When the criteria are met, action is taken to ensure that the environment remains healthy.

These autonomic managers and the on-demand router (ODR) are the primary functional parts of dynamic operations. An ODR can be defined and started before any service policies are defined, but operational policies can be defined before the appearance of the work to which they apply. However, if policies are not defined, the early work is handled by the default policies.

When work enters the ODR, an optimization effort achieves a balance of performance results. As the work flows, the dynamic workload manager balances the load. As work variations change and the balance of work in the nodes is upset, the application placement controller, autonomic request flow manager, and dynamic workload manager rebalance applications that are running to ensure efficient workflows. The combination of these autonomic managers provides a seamless, end-to-end dynamic runtime ability.

The autonomic controllers within WebSphere XD make use of a number of estimates, but the following two estimates are the most relevant to the integration of DPLARs:

- The quantity of resources required by each request in a given service class is estimated by a work profiler, which updates its estimates as requests are processed, thus enabling the system to adapt to changes in application profiles over time.
- The WebSphere node agent that is running on each node estimates the quantity of resources including memory capacity and CPU bandwidth that are available at that node. The node agent publishes information about the available resources by following these procedures:
 - Entering that information into a distributed data structure that is accessible to the WebSphere XD autonomic controllers
 - Sending updates of that information to WebSphere XD components that request them.

With WebSphere XD, when the WebSphere node agent is running in an AIX DLPAR, it periodically monitors the CPU and memory resources that are available to the partition. When resource availability changes, the node agent updates the WebSphere internal data structures and actively notifies any components, including the application controller, of the new resource levels.

Because the autonomic managers dynamically adjust their calculations as those values change, the routing and application placement decisions that WebSphere XD makes in the clusters and the applications that it manages quickly reflect the new allocation of resources between partitions.

9.2 Dynamic reconfiguration manager

An AIX 5L subsystem known as the Dynamic Reconfiguration manager (DR manager) executes change requests for dynamic LPAR resources and notifies applications about resource changes in the partition. By default, an application does not receive any information about changes in the resources; instead, the application has to register itself explicitly in order to receive such resource change notifications.

The resource change notification is referred to as a *dynamic reconfiguration event* (DR event). A DR event can be related to memory, CPU, or PCI I/O slot. It is the responsibility of the DR manager to fulfill the DR event request. The DR manager notifies registered applications at various stages of the DR event.

When notified, the applications can take the necessary actions to maintain integrity and scalability with either an increase or a decrease in available resources. An application registers to receive DR event notifications by installing a DR script through the **drmgr** command (for installing and configuring DLPAR), or by trapping a newly introduced **SIGRECONFIG** signal. Another alternative is for the application to poll a partition periodically. WebSphere XD implementation is based on the polling mechanism.

Note: WebSphere XD can only benefit from the memory and CPU resource changes.

9.2.1 Processor DR events

Consider what happens when more virtual processors are assigned to a partition that is in use as part of a dynamic cluster that is managed by WebSphere XD. This can happen as the result of manual administrator action via the HMC, or the **drmgr** command for System p.

As a result of the processor DR event, when a change occurs to the number of virtual processors that are available to a partition in which an WebSphere XD node agent is running, the node agent makes that information available to the WebSphere XD autonomic controller. The controllers read that new information and dynamically make use of it in their calculations.

Because the flow controller is aware that more processing power is available to service requests on the node, it will route correspondingly more requests to that node if there is sufficient demand. Additionally, if the placement manager knows that more processing power is available to run applications, it will start additional application servers on that node, assuming sufficient demand and sufficient available memory.

9.2.2 Memory DR events

Similarly, as a result of a memory DR event, when there is a change in the amount of memory assigned to a partition that is in use in a WebSphere XD dynamic cluster, the change and the new memory size is detected by the node agent and is published to the other components of the system. When a node has more memory, the application placement manager will be able to start more application servers running in that node (all other things being equal), thus resulting in improved service to the corresponding applications.

9.3 Performance and scalability with DLPAR

The necessity for static overprovisioning is reduced because WebSphere XD is able to dynamically adapt and adjust its behavior, both in request routing and application placement, as the resources allocated to partitions change. Not every application needs to be simultaneously preallocated with the maximum amount of resources that it will need at peak demand, because resources can be dynamically allocated as needs change.

Combining the dynamic resource partitioning available through dynamic logical partitions with the dynamic operations of WebSphere XD enables the data center to be reconfigured through a combination of manual actions and autonomic controllers, as demand and business goals change, without requiring redeployment or static reconfiguration.

9.3.1 WebSphere Partitioning Facility with DLPAR

If WebSphere Partitioning Facility (WPF)-enabled applications are running within a cluster consisting of DLPAR-capable machines, then it is possible to allocate additional central processing unit (CPU) resources to mitigate latency or higher demand scenarios without suffering even one partition subset outage. Refer to *Using WebSphere Extended Deployment V6.0 To Build an On Demand Production Environment*, SG24-7153, for details about WebSphere Partitioning Facility.

However, if you are utilizing a blade center or Linux cluster, for example, and want to even out the load against the given cluster members, then an outage for at least a subset of the endpoint partitions will occur. This outage occurs because the partitioning facility needs to take partitions offline during the move and then reactive them on another physical blade.

In the case of a DLPAR-capable machine, more resources can be provided to handle partitions that are receiving an abnormally high number of transaction requests.

SMP machines preferred in partitioned implementations

Two or more processors run your requests faster than one for a runnable queue of N entries, and four processors are twice as fast, in general. If you cannot segregate the applications with lighter central processing unit (CPU) workloads on to a LPAR, then adding more processors to a DLPAR might be more appropriate, because SMPs are inherently less susceptible to the problems associated with mixed CPU load scenarios. The node agent makes this information available to WebSphere XD autonomic controllers. The controllers read that new information and dynamically make use of it in their calculations.

Schedule the operating system to use shorter times

Some operating systems use time periods up to 200ms, which is too long when many threads can be scheduled. Shortening the maximum time value to 10-30ms can help by allowing more threads per second to run through the available CPUs. This can lower the scheduling latency at the expense of delaying threads that take longer than 10ms to complete.

Some application-specific tuning might be required, but try different values to see what works best for your application. More context switches might be required, but most modern microprocessors are capable of billions of instructions per second.

Simultaneous Multi-Threading

Simultaneous Multi-Threading (SMT) is the ability of a single physical processor to simultaneously dispatch instructions from more than one hardware thread context. It is a feature of the POWER5 processor and will be available at the same time as shared processors. There are two hardware threads per processor. SMT is designed to take advantage of the superscalar nature of the POWER5 processor, so that more instructions can be executed at the same time. The basic concept is that no single application can fully saturate the processor, so it is better to have multiple applications providing input at the same time.

SMT is expected to be used primarily in environments, where the speed of an individual transaction is not as important as the total number of transactions that can be performed. It is expected to increase the throughput of workloads with large or frequently changing working sets, such WebSphere Application Server.

Archived

Implementation scenario

This chapter describes our test installation and the configuration methods we used in the development of this book. The goal was to provide an optimized infrastructure for the project team when running operations such as:

- ▶ Provisioning LPARs using a standardized operating system image
- ▶ Installing and customizing WebSphere
- ▶ Controlling and changing dynamically allocated hardware resources
- ▶ Controlling and changing virtual I/O devices
- ▶ Deploying sample WebSphere Application Trade 6.1

In addition, we tried to arrange our installation to support the Application Lifecycle Management required to operate our processing resources in an efficient and optimized way.

First we discuss the scenario implementation and configuration, explaining how we optimized the system to simplify it and make it available as fast as possible. Then we show how you can implement the scenario.

The sample scenario demonstrates the use of Dynamic LPAR (DLPAR) with WebSphere Application Server V6 Network Deployment. Installing and configuring WebSphere Application Server and the sample application Trade 6.1 is also covered. To illustrate the effect from changes involving POWER5 resources, Rational Performance Tester was used to run load against Trade 6.1, and we discuss the results that were recorded.

10.1 Scenario overview

This section provides an overview of our sample scenario, including the software required, a description of our sample topology, and the applications installed in our test environment.

10.1.1 Software products

We used an LPAR in our POWER5 test environment. The following version of the operating system was employed:

- ▶ IBM AIX 5.3 Maintenance Level 5 with 64-bit kernel

The following IBM software was used for the WebSphere implementation:

- ▶ IBM WebSphere JDK™ 5
- ▶ IBM WebSphere Application Server Network Deployment V6.1
- ▶ IBM DB2 UDB V9.1

To obtain more information about the minimum hardware and software requirements for WebSphere and DB2 UDB, refer to these sites:

<http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html>

<http://www.ibm.com/software/data/db2/udb/sysreqs.html>

Setting up the test environment

To set up our test environment, we performed these tasks.

1. Set up LPARS by creating profiles for memory and CPU allocation profiles using the HMC graphical user interface
2. Create virtual SCSI and network devices using the HMC GUI
3. Install and configure the VIO Server
 - a. Advanced VIO setup mirror VIOS with backup/restore
 - b. Create storage pools for the client root volume group (rootvg) and application volume group
 - c. Create a shared Ethernet adapter on the VIOS
4. Install and configure CSM on the management server

Note: A sample CSM adapter definition file is included in “CSM adapter definition file: p550q_lpar_adapters” on page 471 in Appendix A, “Sample files” on page 423.

5. Install NIM on the management server
 - a. Create CSM/NIM resources
 - b. Obtain the adapter Ethernet MAC address
 - c. Configure CSM input and add the nodes
 - d. Csm2nim nodes
 - e. Allocate NIM resources and start bos_inst from the mksysb resource
6. Perform basic operating system installation and configuration
 - a. Install all required AIX Filesets and GNU software
 - b. Configure AIX in terms of deploying NFS remote mount, SSH and IOCP
 - c. Preconfigure AIX in terms of creating the standard Filesystem /usr/IBM on the logical volume WebSphere Application Server _LV
 - d. Perform basic TCP/IP tuning
 - e. Perform basic AIX tuning (ulimit settings, maxuproc settings, alog settings)
 - f. Create the mksysb for later installation of the other partitions
7. Set up NFSv4 on Trinity using nfsroot and exname to share the /exports filesystem used to store the installation directory as well configuration files/scripts
8. Monitor via perfagent.server installed on Brazos; client software is automatically installed
9. Perform basic DB2 installation:
 - a. Install DB2 on Brazos as the DB2 Server
 - b. Install DB2 Client into the NFS remote mounted dir
 - c. Create user db2rc1 on all client systems and creating symbolic links
10. Install WebSphere amd Trade61

Sample installation layout

Table 10-1 lists the systems used in our test environment.

Table 10-1 Managed system resources - test environment

Managed system name	Logical name	Resources
p5-9113-550-SN104790E	P550	2 processors 4 GBtpe RAM
p5+-9133-55A-SN10D1FAG	P550Q	8 processors 32 GBtpe RAM

We configured our test system as shown in Figure 10-1 in order to subsequently install the optimized environment.

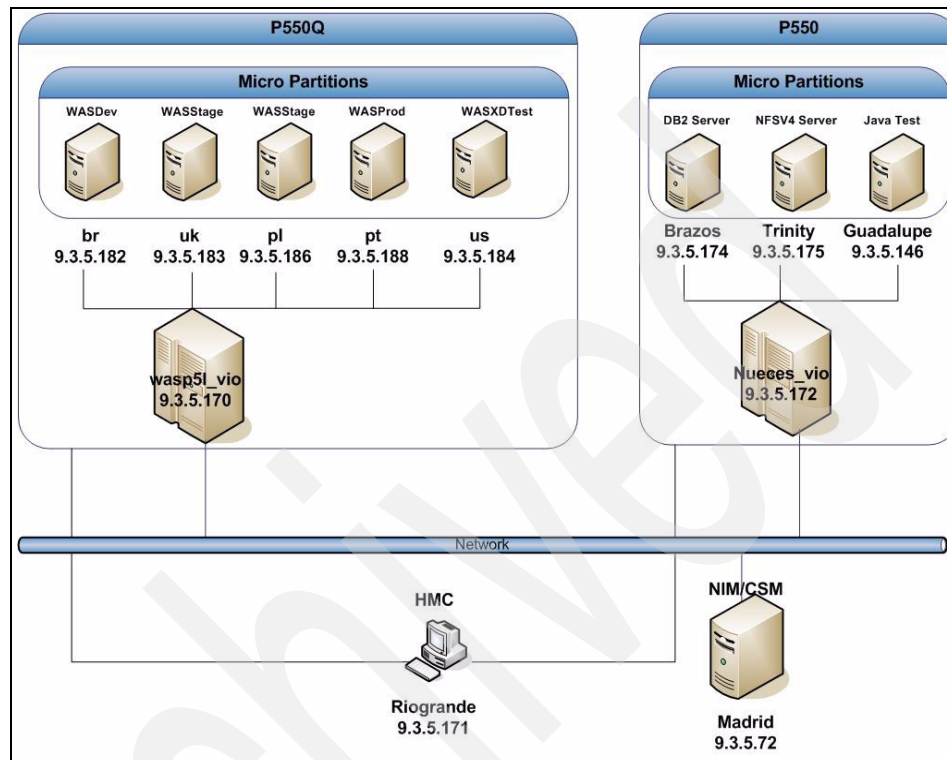


Figure 10-1 Our test environment LPAR definition layout

10.1.2 The sample WebSphere-related scenario

Our sample topology demonstrates the following key features:

- ▶ The ability to add to or remove processor resources using the P5 server management tool.
- ▶ The virtualization capability of P5.
- ▶ The ability to reconfigure LPARs dynamically.

Figure 10-2 on page 389 illustrates the sample scenario and includes the following items:

- ▶ A single LPAR
- ▶ A single instance of the application server
- ▶ A deployment manager managing WebSphere Application Server
- ▶ An application server cell, running IBM WebSphere Application Server Network Deployment V6

- ▶ A dedicated database server running IBM DB2 UDB V9.1
- ▶ A Windows XP-based test machine running the Rational Performance Tester.

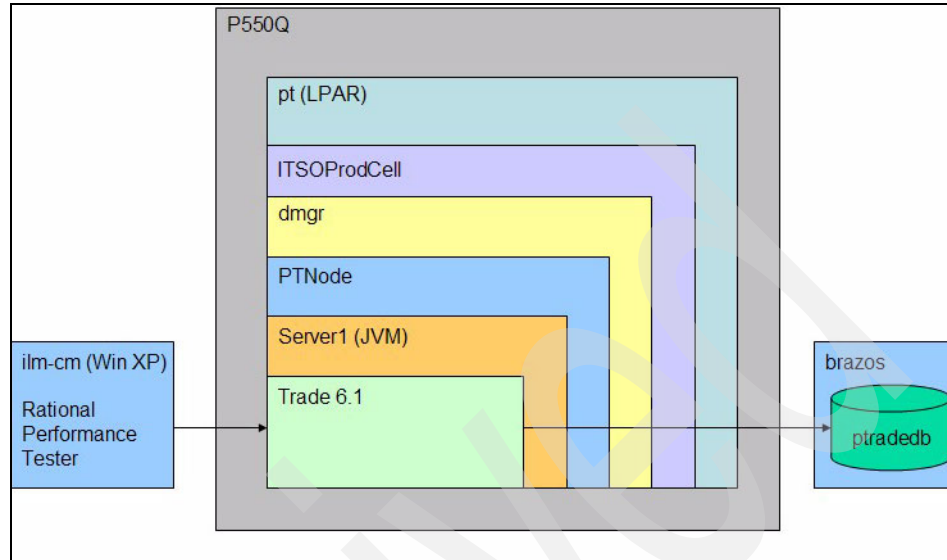


Figure 10-2 Sample Trade 6.1 stack of software scenario

Sample WebSphere installation layout

We defined the layout of the WebSphere installation from the perspective of Application Lifecycle. We created three WebSphere Cells, which allocated four LPARs. We used the definitions listed in Table 10-2 and a logical layout for the WebSphere Application Server as referenced in Figure 10-3 on page 390.

Table 10-2 Our scenario's WebSphere cell definition

Cell name	Cell purpose	Hostname/LPAR	Purpose
ITSOCCell	Development	br.itsc.austin.ibm.com	Deployment Manager, AppServer
ITSOSTageCell	Staging	uk.itsc.austin.ibm.com	Deployment Manager
		pt.itsc.austin.ibm.com	AppServer
ITSOProdCell	Production	pl.itsc.austin.ibm.com	Deployment Manager, AppServer

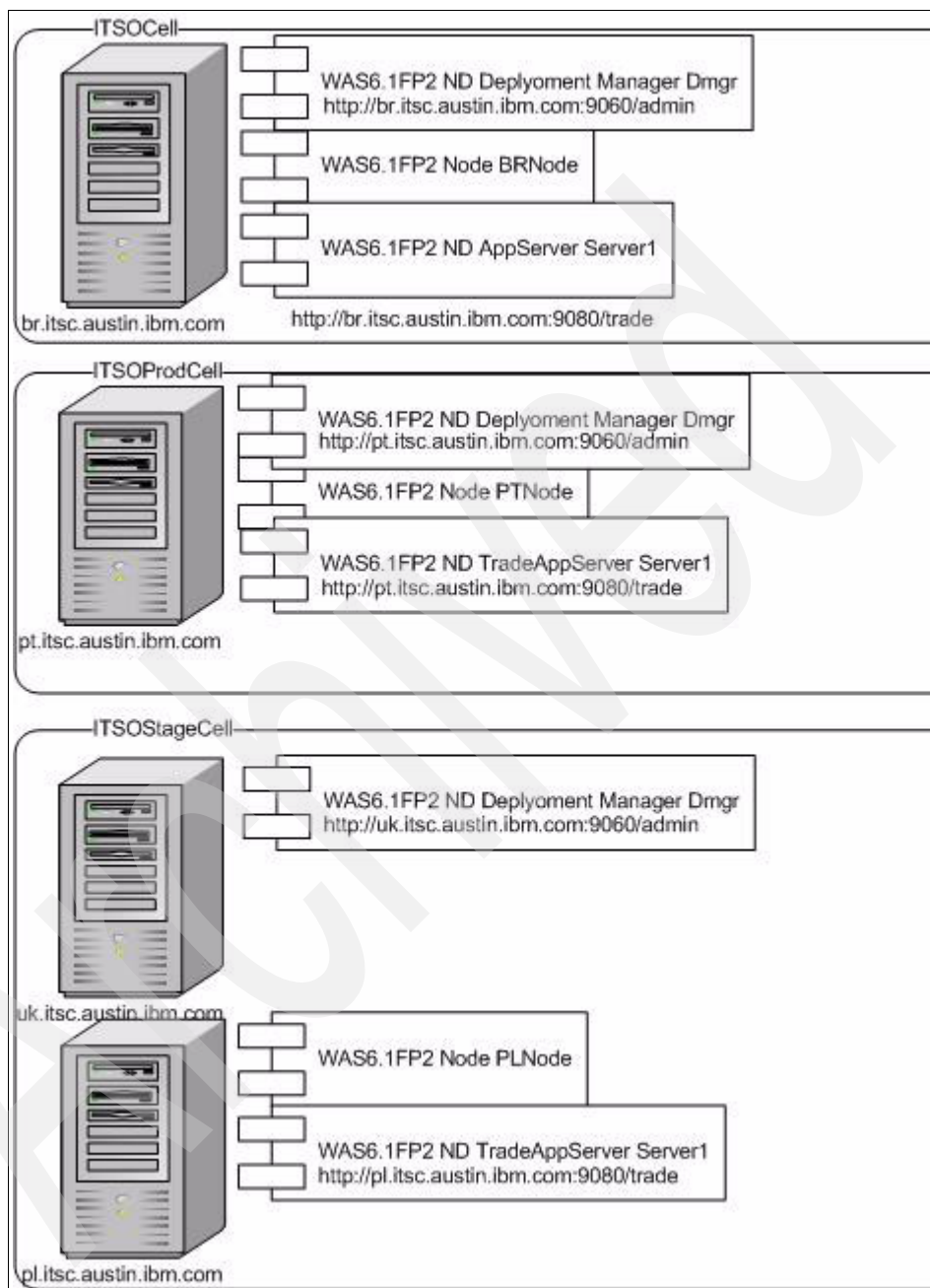


Figure 10-3 Our scenario's WebSphere cell definition layout

10.2 Installation summary

Table 10-3 summarizes the installation procedure. These steps need to be carried out before proceeding with the rest of this chapter. We will describe how we installed each individual component as we optimized the installation process.

Table 10-3 Summary of WebSphere-related installation steps

Step	Action	System
1	Install IBM DB2 UDB V9.1 server	brazos
2	Install IBM DB2 V9.1 client	pt
3	Catalog DB2 node on clients	pt
4	Install IBM WebSphere Application Server Network Deployment V6	pt
5	Install Fix Pack 2 for IBM WebSphere Application Server Network Deployment V6	pt
6	Install Trade 6.1	pt
7	Install Rational Performance Tester	ilm-cm

Initial definitions for partitions

We created a set of partitions within the HMC using the initial settings shown in Table 10-4.

Table 10-4 Initial memory and processor layout

Partition name	Memory (Min/Max/Desired)	Processor units (Min/Max/Desired)	Virtual processor (Min/Max/Desired)
br	4/8/4	0,5/2,0/1,0	1/4/2
uk	4/8/4	0,5/2,0/0,5	1/4/1
pl	6/8/6	1,0/4,0/2,0	1/4/2
pt	3/10/5	1/4,0/2,0	1/4/2
us	1/8/1	0,5/2,0/0,5	1/4/2

The NFSv4 filesystem used

We decided to use NFSv4 along with NFSRoot and the exname option. This enabled us to export filesystems from different physical locations and mount these directories on the client by using a single NFS mount command. We implemented the layout that is illustrated in Figure 10-4 on page 392.

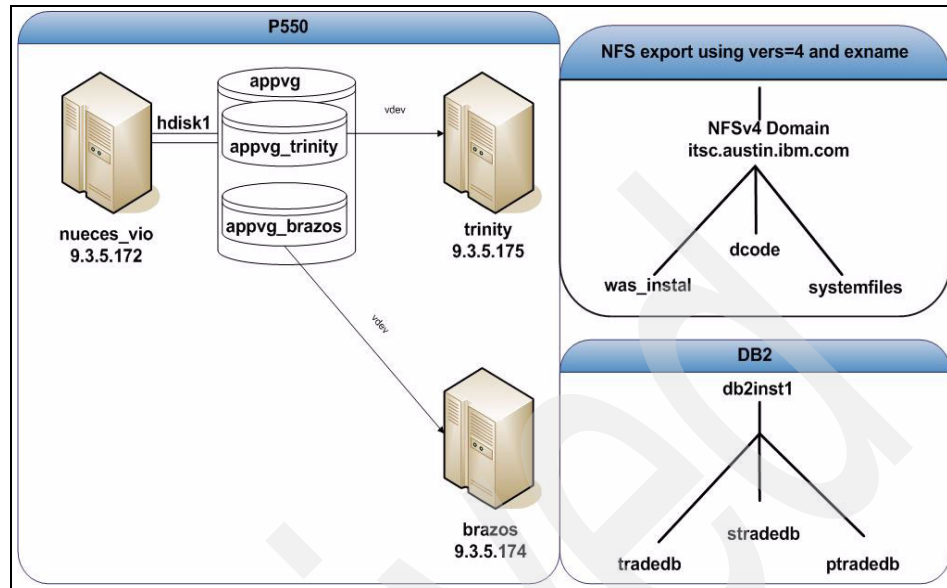


Figure 10-4 NFSv4 filesystem layout

For more information about NFS v4, refer to *Securing NFS in AIX: An Introduction to NFS v4 in AIX 5L Version 5.3*, SG24-7204.

10.2.1 System description

The standalone server (CSM and NIM server) running AIX 5.3 with the latest main ten ace level has the structure displayed in Example 10-1.

Example 10-1 CSM-managed server filesystems

#df	Filesystem	512-blocks	Free	%Used	Iused	%Iused	Mounted on
	/dev/hd4	98304	30752	69%	2041	9%	/
	/dev/hd2	2490368	56536	98%	32204	11%	/usr
	/dev/hd9var	131072	101912	23%	676	5%	/var
	/dev/hd3	98304	94928	4%	75	1%	/tmp
	/dev/hd1	32768	22296	32%	46	2%	/home
	/proc	-	-	-	-	-	/proc
	/dev/hd10opt	360448	83816	77%	4177	10%	/opt
	/dev/nimsource	35553280	7954920	78%	9908	1%	
	/export/lpp_source						
	/dev/nimspot	3080192	2115688	32%	12973	4%	/export/spot
	/dev/nimbundle	2097152	1311552	38%	44	1%	
	/export/bundles						

/dev/nimmksysb	6291456	1491080	77%	19	1%
/export/mksysb					

Note that all the filesystems are prepared to be exported (/dev/nimmksysb).

We used NIM and CSM to create our LPAR. Then we created a mksysb of the server (NEW LPAR) and stored it under the standalone server. This will simplify future installations and also allow fast restoration of the server if problems occur.

10.3 Installing WebSphere Application Server

WebSphere Application Server is supplied with an installation wizard and the Installation Factory. We used the installation wizard so we would be able to install and bring up the WebSphere Server in a repeatable and automated fashion.

WebSphere Application Server 6.1 is structured so there is a single binary installation, but it supports multiple profiles (for example, deployment managers and application server). Consequently, the IBM Installer can install binary files and create profiles in one step.

We optimized the installation step by using silent installation. Installing WebSphere Application Server Network Deployment using silent installation refers to using a file to supply installation options without user interaction. To configure the installation, change the options in the response file before you issue the installation command. Silent installation mode does not accept interactive installation options. To specify non-default options during a silent installation, you must use the response file.

Using silent installation offers you the following benefits:

1. You can automate the installation tasks.
2. You can avoid prerequisite checking, which could fail for environments on a newer fix level.
3. You can avoid manual configuration tasks.
4. You can ensure repeatable and consistent installation.

Because the WebSphere Application Server install image is exported on a NFS filesystem and mounted under the /exports directory, we further optimized the installation process by invoking the installation wizard remotely by using the **dsb** command as shown in Example 10-2 on page 394.

Example 10-2 Silent installation of WebSphere Application Server remotely

```
# dsh -s -n pt /exports/was_instal/was61/WebSphere Application Server
/install -options
"/exports/systemfiles/was_response/pt.responsefile.txt" -silent
```

The response file is customized for each node, depending on the topology. Refer to “WebSphere: responsefile.nd.txt” on page 424 in Appendix A, “Sample files” on page 423, for the response file we used for the installation of our sample scenario. Example 10-3 highlights the changes made in our response file with comments.

Example 10-3 WebSphere Application Server response file

```
#####
#
# License Acceptance
#
# This must be set to “true” so that you accept all IBM license terms associated with
# this product, which is necessary for installing WebSphere Application Server

-OPT silentInstallLicenseAcceptance="true"

#####

# Operating System Prerequisite Checking
#
# The operating system prerequisite checking was disabled because of the higher
# level fixes installed on the system
#
-OPT disableOSPrereqChecking="true"

#####

# Non-blocking Prerequisite Checking
#
# The non blocking prerequisite checking was disabled to allow the installer to
# continue with the installation and log the warnings even though the prerequisite
# checking has failed.
#
-OPT disableNonBlockingPrereqChecking="true"

#####

# Install a New Copy
#
# For a new WebSphere Application Server installation, the installType must be set to
“installNew”

-OPT installType="installNew"
```

```
#####

# The selection state of the "Application Server samples" feature.
#
# We do not require the installation of the sample application.
#
-OPT feature="noFeature"

#####

# AIX Default Install Location:

# We used the default installation path for WebSphere Application Server
#
-OPT installLocation="/usr/IBM/WebSphere/AppServer"

#####

# Profile Creation Selection
#
# By setting the profileType to "cell" we imply both the Dmgr and AppSrv will be hosted
# on the same system. This will create a Dmgr profile and AppSrv profile on the same
# system.
#
-OPT profileType="cell"

#####

# Administrative Security
#
# We do not want to enable AdminConsole security
#
-OPT PROF_enableAdminSecurity="false"

#####

# Deployment Manager Profile name
#
# The Dmgr profile name must be specified
#
-OPT PROF_dmgrProfileName=Dmgr

#####

# Application Server Profile name
#
# The AppSrv profile name must be specified
#
-OPT PROF_appServerProfileName=AppSrv

#####

#
# Host name
```

```

#
# The host name must be specified
#
-OPT PROF_hostName=pt

#####
#
# Deployment Manager Node name
#
# The Dmgr node name must be specified
#
-OPT PROF_nodeName=ITS0CellManager

#####
#
# Application Server Node name
#
# The AppSrv node name must be specified
#
-OPT PROF_appServerNodeName=PTNode

#####
#
# Cell name
#
# The cell name must be specified
#
-OPT PROF_cellName=ITS0Cell

#####
#
# Default Ports
#
# Use WebSphere Application Server default ports
#
-OPT PROF_defaultPorts="true"

#####
#
# Validate Ports
#
# This is set to true to ensure they are not reserved or in use.
# Otherwise, no port validation checking will occur.
#
# Valid value:
#   true - enables port validation.
#
-OPT PROF_validatePorts="true"

```

10.3.1 Installing the WebSphere Application Server 6.1 Fix Pack 2

Fix Pack 2 is the latest fix pack for WebSphere Application Server. Before installing this fix pack, however, you must update the Update Installer. The Update Installer must be downloaded separately because it is not packaged as part of the Fix Pack download.

Installing the Update Installer

In our case, we optimized the installation by defining a response file for the silent installation of the Update Installer. Refer to “Update Installer: response.txt” on page 452 in Appendix A, “Sample files” on page 423, for a copy of the Update Installer response file. Example 10-4 shows the command for installing the Update Installer.

Example 10-4 Silent installing the Update Installer

```
# dsh -s -n pt /exports/was_intall/updateinstaller/UpdateInstaller/install
-options "/export/was_instal/UpdateInstaller/install.updiinstaller.txt" -silent
```

Installing Fix Pack 2

We optimized the installation of the Fix Pack 2 by customizing the response file. Refer to “WebSphere V6.1 Fix Pack 2: fp2.response.txt” on page 455 in Appendix A, “Sample files” on page 423 for a copy of the response file. Example 10-5 shows the command for installing Fix Pack 2.

Example 10-5 Silent installation of Fix Pack 2

```
# dsh -s -n pt /usr/IBM/WebSphere/UpdateInstaller/update.sh -options
"/exports/was_intall/was61fp2/install.response.txt" -silent
```

10.4 Installing and configuring Trade 6.1

This section explains how we installed and configured the IBM Trade Performance Benchmark Sample for WebSphere Application Server (called Trade 6.1 throughout this book) into our lab environment.

Trade 6 is the fourth generation and latest release of the WebSphere end-to-end Trade benchmark and performance sample application. The Trade benchmark was designed and developed to cover the significantly expanding programming model and performance technologies associated with WebSphere Application Server. This application provides a real-world workload, enabling performance research and verification test of the Java 2 Platform, Enterprise Edition (J2EE)

1.4 implementation in WebSphere Application Server, including key performance components and features.

Trade 6.1 is targeted for WebSphere Application Server 6.1. Updates in this package include hidden support for iSeries database for a single server, modifications to the Trade source and database schema required by a change in ejbdeploy-down mapping in WebSphere Application Server 6.1. Long run support is now the default option and is configurable via the web.xml file. Changes to the installation and configuration jac1 scripts were made to reflect the new 6.1 SIBus security function. When security is enabled, users and groups need to be explicitly granted permission to access the SIBus.

Overall, the Trade application is primarily used for performance research on a wide range of software components and platforms. Trade's design enables performance research on J2EE 1.4 including the new Enterprise Java Bean 2.1 component architecture, message-driven beans, transactions (1-phase, 2-phase commit) and Web services (SOAP, WSDL, JAX-RPC, enterprise Web services). Trade 6 also drives key WebSphere Application Server performance components such as dynamic caching, WebSphere Edge Server, and EJB caching.

For details about how Trade 6 exploits all of these functions, refer to the document tradeTech.pdf, which can be found inside the installation package. Trade 6 demonstrates several new features in WebSphere V6.1. All Trade versions are WebSphere version-dependent, so Trade 6.1 will only work with WebSphere Application Server V6.

Note: The most common configurations for Trade 6.1 are single server and horizontal clustering scenarios. For our scenario, we used a single server but do *not* take advantage of horizontal or vertical clustering.

10.4.1 Trade 6.1 installation summary

The following actions have to be performed in order to set up and install Trade 6.1:

- ▶ Download the Trade 6.1 installation package.
- ▶ Set up and configure the Trade database.
- ▶ Configure the WebSphere cell.
- ▶ Install the application.
- ▶ Regenerate the Web server plug-in, which is plugin-cfg.xml. (This is only necessary if you are using a Web server. For our scenario, we use the internal Web server of the application server.)
- ▶ Restart servers.

10.4.2 Download the Trade 6.1 installation package

You can download the Trade 6.1 installation package from the following Web site:

<http://pulsar.raleigh.ibm.com/>

After we downloaded the file `tradeInstall.zip`, it was placed in our NFS as a install image.

10.4.3 Set up and configure the `tradedb` database

Trade 6.1 uses a database to store its data. We used of DB2/UDB V9.1 in our environment. We created a DB2 database called “`ptradedb`” on the database server, and DB2 client access must be set up on each application server node.

DB2 server

We used the default DB2 instance on the server (`db2inst1`). In your case, follow these steps.

1. Log on to the database server as the DB2 administrator and start a DB2 shell using the `db2cmd` command.
2. Copy the file `Table.ddl` from the Trade 6 install directory to the DB2 server.

Then execute the DB2 commands shown in Example 10-6 to create the `tradedb` database. In our scenario, we called our database `ptradedb`, as shown in the example.

Example 10-6 DB2 commands to create the `tradedb` database

```
db2 create db ptradedb
db2 connect to ptradedb
db2 -tvf Table.ddl
db2 disconnect all
db2 update db config for ptradedb using logfilsiz 1000
db2 update db cfg for ptradedb using maxappls 100
db2stop force
db2start
```

DB2 clients

Because Trade 6.1 uses the DB2 universal JDBC Type 4 driver, you do not need to configure the DB2 clients on the application server systems any more (as was

needed for previous versions of Trade). If you encounter issues when accessing the database, follow these steps to resolve the connection problems.

1. Log on to the application server machine (pt) as the DB2 instance owner and start a db2 shell.

2. Using this shell, catalog the remote database server db (thus creating a local node representing it) by using the following command:

```
db2 catalog tcpip node <your_local_node> remote
<db2_server_hostname> server
<port_from_the_services_file_on_your_OS_(50000)>
```

In our case, the correct command was:

```
db2 catalog tcpip node pt remote brazos server 50000
```

3. Catalog the remote database ptradedb:

```
db2 catalog database ptradedb as ptradedb at node
<your_local_node>
```

In our case, the correct command was:

```
db2 catalog database ptradedb as ptradedb at node pt
```

4. Verify the database connectivity from the DB2 client machine (pt):

```
db2 connect to tradedb user <db_user> using <password>
```

In our case, the following command displayed the information shown in Example 10-7:

```
db2 connect to ptradedb user db2rccli using <password>
```

Example 10-7 Database connectivity from DB2 clients to DB2 server

Database Connection Information

Database server	= DB2/AIX64 9.1.0
SQL authorization ID	= DB2RCL1
Local database alias	= PTRADEDDB

10.4.4 Install Trade 6.1 using the installation script

To prepare your J2EE environment to run Trade 6.1, you must create several resources (such as JDBC and JMS resources). For example, on each application server node there must exist a JDBC data source that points to the database tradedb you just created. In our scenario, we only had one application server.

The configuration steps could be performed using the WebSphere Administrative Console, but that would take a while. Fortunately, however, Trade 6.1 comes with

a .jacl script that can be used to configure and install Trade 6 in a clustered environment. However, this script file also requires significant manual interaction, such as entering the paths for DB2 client path and DB2 server name.

In our case, we optimized the script by coding a number of values as defaults so the script could be reused for installing other environments. Example 10-8 shows the changes we made to trade.jacl.

Example 10-8 Customization of trade.jacl

```
# Customized options for database classpaths

set DB2JccPath
"/home/db2rc11/sql1lib/java/db2jcc.jar;/home/db2rc11/sql1lib/java/db2jcc_license_cu.jar"

set DB2NativePath          "/opt/IBM/db2/V9.1/lib64"

set DB2CliPath              "/home/db2rc11/sql1lib/java/db2java.zip"

# Default options for database user name
set DefaultUserDB2          "db2rc11"

# Default options for database password
set DefaultPasswdDB2        "login1"

# DB2 database options
set DefaultDatabaseName     "stradedb"
set DefaultDB2Hostname      "brazos.itsc.austin.ibm.com"

# Silent install properties for Managed Node
# - Modify these properties to specify the target node and server
set TargetNodeName           "PLNode"
set TargetServerName         "Server1"
```

A copy of the entire trade.jacl installation script can be found in “Trade 6.1 Installation script: trade.jacl” on page 456 in Appendix A, “Sample files” on page 423, with the customizations highlighted.

From the /exports/was_install/trade61/tradeinstall directory, install Trade 6.1 by issuing the following command:

```
<WebSphere Application Server_HOME>/bin/wsadmin.sh -f trade.jacl
```

This command performs the installation in a two-step process. The first step creates the necessary JDBC and JMS resources. The second step installs the application ear file.

Note: Because we used a variable for the JDBC driver path, we now needed to set the value for this variable according to our DB2 client installation. In **your case, go to Environment -> WebSphere Variables** in the Administrative Console. Click **DB2 UNIVERSAL_JDBC_DRIVER_PATH** and enter the path to your SQLLIB/java directory (for example, /home/db2rc11/sql11b/java).

If you are installing a single server configuration, restart the WebSphere Application Server instance by using the **startManager.sh** and **startNode.sh** commands.

10.4.5 Working with Trade 6.1

The last step is to verify the Trade 6.1 installation. First test whether Trade 6.1 runs on your application server. You can connect to Trade 6.1 without a Web server by connecting directly to the Web Container Inbound Chain of one of your application servers. Open the Trade 6.1 start page using the following URL (the port number depends on your configuration):

http://<host_name>:9080/trade

In our environment, the URL was:

<http://pt.itsc.austin.ibm.com:9080/trade>

If this is successful, you should see the overview page as shown in Figure 10-5 on page 403.

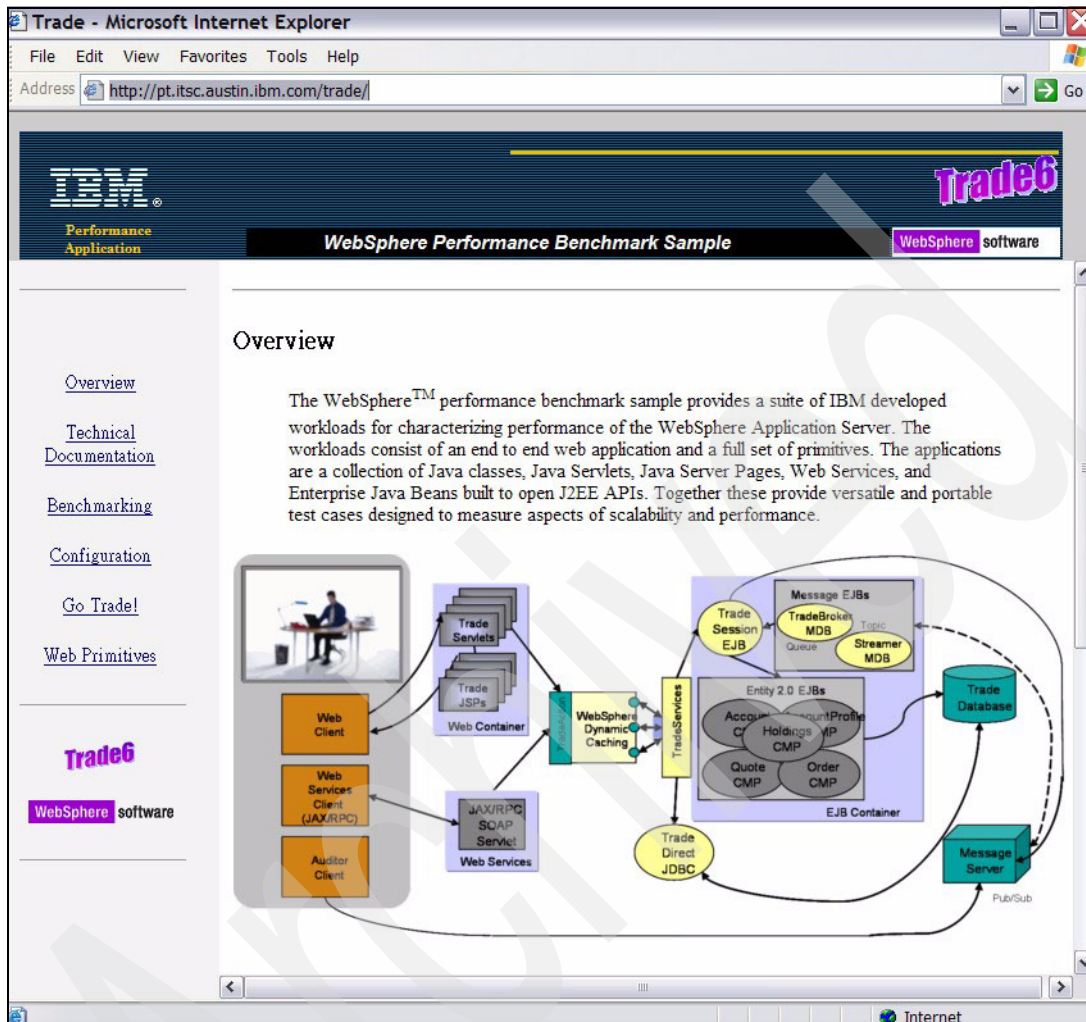


Figure 10-5 Trade 6.1 Overview page

Before using the Trade 6 application, you must populate the database, as explained here:

1. Click the **Configuration** link on the left side of the Trade 6 index page.
2. In the Configuration utilities window, click **(Re-)populate Trade Database**. Wait until the database population has been completed.
3. Before closing the window, review the default Trade 6 configuration, then click **Update Config** button.

4. After populating the database, run the following DB2 commands to update DB2 statistics:

```
db2 connect to ptradedb user <db_user> using <password>  
db2 reorgchk update statistics
```
5. Click the **Go Trade!** link on the left side of the Trade 6.1 Overview page. You are forwarded to the Trade 6.1 login page shown in Figure 10-6
6. Click the **Log in** button and start to use Trade 6.1.

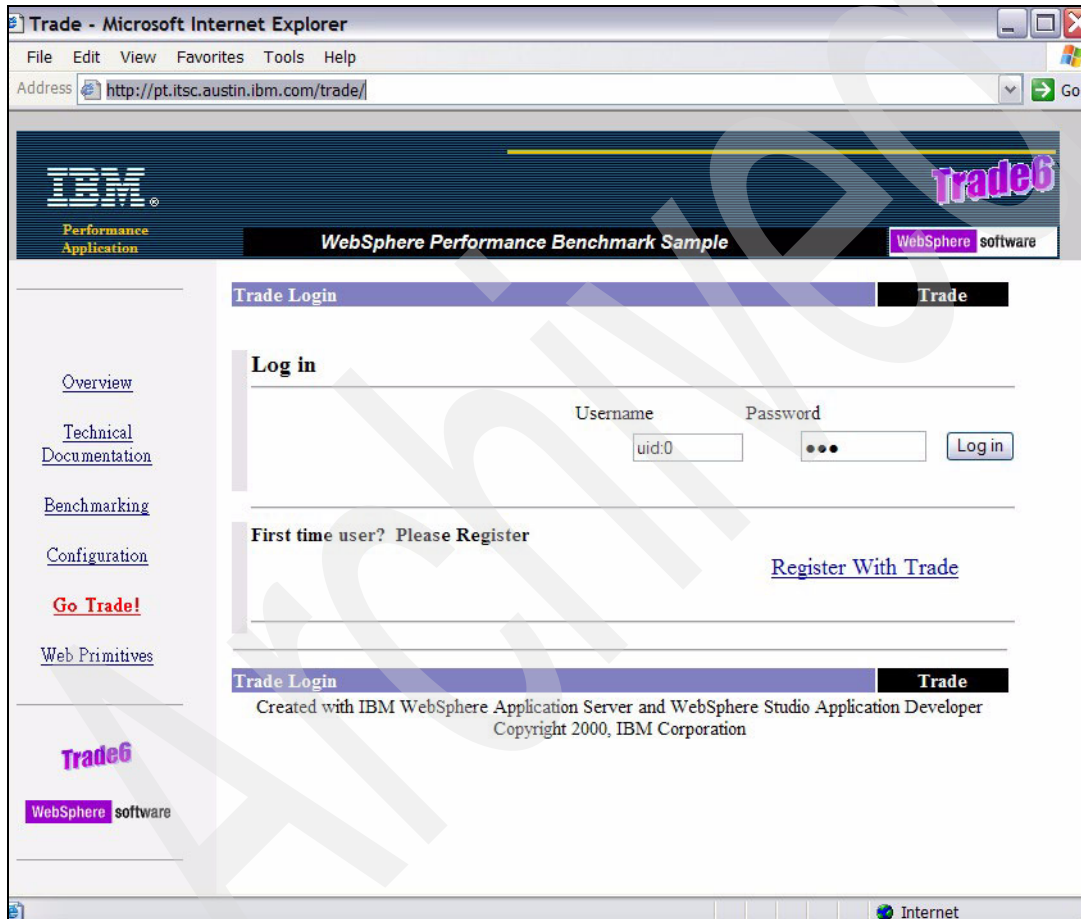


Figure 10-6 Trade 6.1 Log in page

10.5 Performance testing

Application performance testing is an important component of the software deployment cycle. It becomes even more important with respect to Web applications, because an external user's tolerance for slow applications is generally much lower than that of a captive internal audience. Performance testing has been a subject of many products and white papers recently, and has spawned a new IT industry dedicated to Web application testing.

10.5.1 General application performance testing requirement

When performance testing a Web application, several requirements must be determined either through interpretation of data from an existing application that performs similar work, or from best-guess estimates. Those requirements are:

- ▶ Average request rate

What is the expected number of users who will access this application? This is generally expressed in hits per month, day, hour, or minute, depending on volumes. This rate should be reevaluated regularly.

- ▶ Peak request rate

How many pages will need to be served per second? This also should be reevaluated regularly.

- ▶ Average concurrent users

What is the average number of users accessing the application at the same time during regular usage hours? This rate should be planned for, expected, and reevaluated on a regular basis.

- ▶ Peak concurrent users

What is the maximum number of concurrent users that will visit your site during peak time?

- ▶ Regular usage hours

This value defines your off-peak hours. It is required to simulate a realistic workload.

- ▶ Peak usage hours

During this time, most of your traffic will occur and a performance degradation would impact most of your users.

- ▶ Site abandonment rate

How long will a user stay on your page before the user leaves the site or closes the browser?

The user base, especially for Web applications, is a difficult number to determine, especially if this is an application that is performing a new function on a Web site. Use existing Web server measurements to provide a “best-guess” number based on current traffic patterns for the site.

If capturing these numbers is not possible, then determine the “breaking point” for the application within the intended deployment infrastructure. This allows you to monitor after the application is live and provide increased capacity, thus possibly avoiding a negative user response due to load.

10.5.2 Scenario overview

The goal of our scenario is not to demonstrate performance tuning Trade 6.1, but to demonstrate the effect of dynamic resource allocation. Therefore, we did not fine-tune Trade 6.1 for performance. We installed and configured Trade 6.1 “out of the box” without any specific tuning. As a result, the performance figures shown in this chapter are not realistic and should not be used to reflect the performance of the application or hardware.

In a production environment, your deployed application may need to support thousands of concurrent users. This is normally supported by distributing the load across a number of Web servers and clustered application servers. For our scenario, however, we did not cluster our Trade 6.1 and did not use a Web server. We used the internal Web server for server1 at the default port 9080.

The key feature we demonstrated is the use of DLPAR to dynamically increase CPU resource by adding more processors to the running LPAR. We deliberately assigned our LPAR with a minimal number of processors and ran a load test against it. The CPU utilization and response time of Trade 6.1 were recorded. While the load test was still running, we dynamically allocated more processors to the LPAR to demonstrate the improved CPU utilization and subsequently the improved response time.

To simulate a high workload to be applied to Trade 6.1, we needed a load testing tool to measure the throughput and response time as though it was in a real test scenario. There are a number of tools available for this purpose. Some are available without charge, and others are not. We used IBM Rational Performance Tester for creating our test case, as explained in the following section.

In the next section, we demonstrate how to create a test case for load testing Trade 6.1.

10.5.3 IBM Rational Performance Tester

IBM Rational Performance Tester 6.1 (RPT 6.1) is a multiuser system performance test product hosted in the Eclipse shell with a Java-based execution engine. The focus of IBM Rational Performance Tester 6.1 is multiuser testing of Web applications.

IBM Rational Performance Tester 6.1 is built from an entirely new code base running in the Eclipse shell. Its predecessor, IBM Rational Performance Tester 6.0, is a Win32-based product with Rational Robot and Rational Test Manager as its primary components. IBM Rational Performance Tester 6.1 provides major improvements in the areas of ease of use and scalability.

Here we briefly introduce the functions of RPT 6.1, in particular how we used it to record and run a performance test. We do not compare it (feature-wise) to other products in this chapter, because the suitability of any of these products depends highly on your requirements. This Eclipse-based version of Rational Performance Tester is available for Windows and Linux (Red Hat Enterprise Linux WS and SUSE® Linux Enterprise Server) platforms.

Efficient performance of multiuser Web applications is a necessity, not a luxury. Even small performance deficiencies and scalability failures can slow business to a halt or cause customers to take their business elsewhere. To capture performance problems before deployment, software development and test teams must proactively measure the ability of their applications to rapidly and accurately support multiple, concurrent users. IBM Rational Performance Tester was designed to address this need.

IBM Rational Performance Tester is a load and performance testing solution that is useful for teams who are concerned about the scalability of their Web-based applications. Combining ease-of-use features with flexibility, Rational Performance Tester simplifies test creation, execution, and data analysis to help teams ensure the ability of their applications to accommodate required user loads before the applications are deployed.

To perform a simple load test with IBM Rational Performance Tester, execute these steps:

1. Record a performance test.
2. Create a performance schedule.
3. Run a performance schedule.

In the following sections, we explain these steps in more detail.

Record a performance test

Before running any tests, you have to create your test project in the workspace and record a Performance Test case. Follow these steps:

1. Open your IBM Rational Software Development Platform. After it starts, it should look like Figure 10-7.

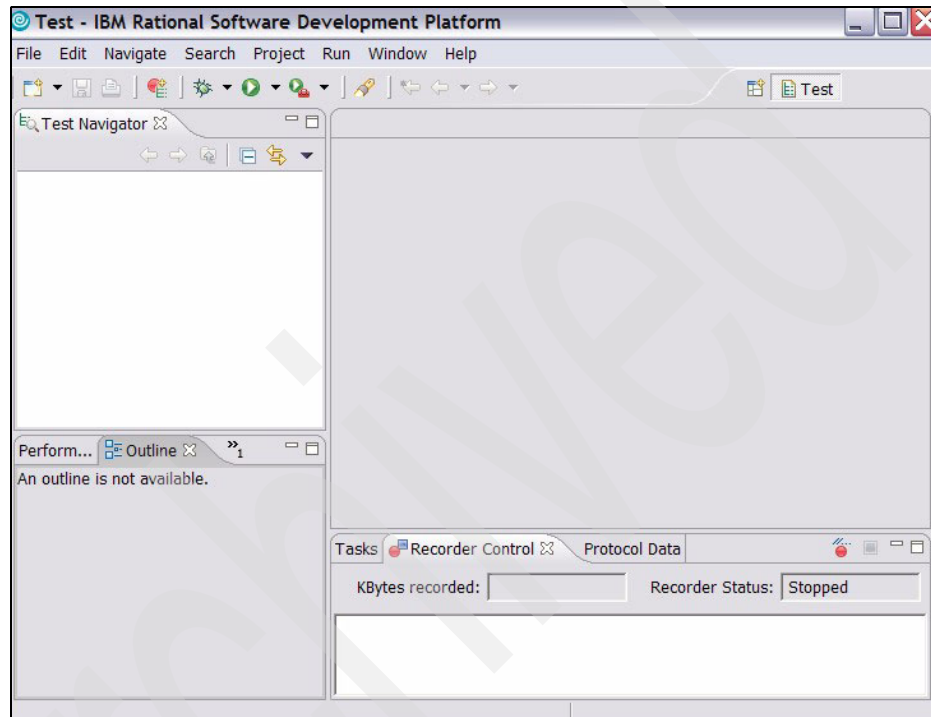


Figure 10-7 Rational Software Development Platform workspace

2. Then click **File -> New -> Performance Test Project** to create a new test project called GoTrade in your workspace. Click **Finish** as shown in Figure 10-8 on page 409.

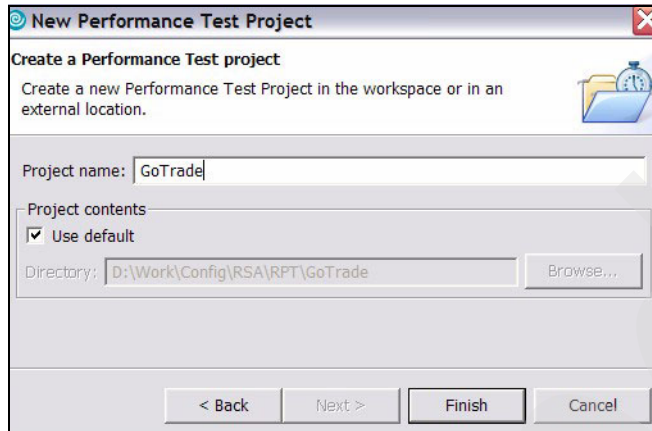


Figure 10-8 Create New Performance Test Project wizard

3. Click **Next** on the Record Test dialog box to select the performance test type for your case, as shown in Figure 10-9.

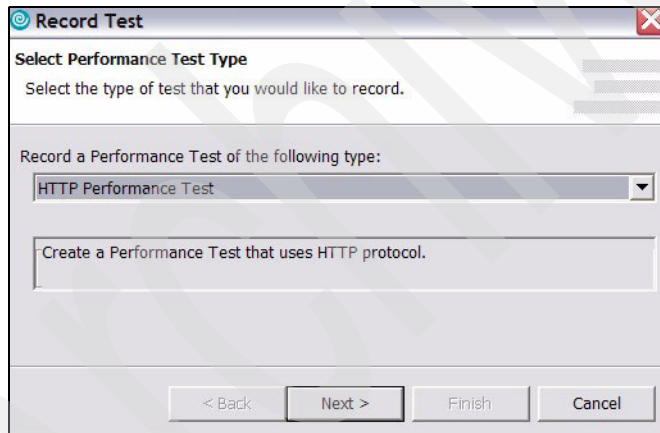


Figure 10-9 Record performance test project wizard

4. Type in TradeTest.rec as the name for your recording in the HTTP Proxy Recorder dialog. Click **Finish**, as shown in Figure 10-10 on page 410, to start recording your test case.

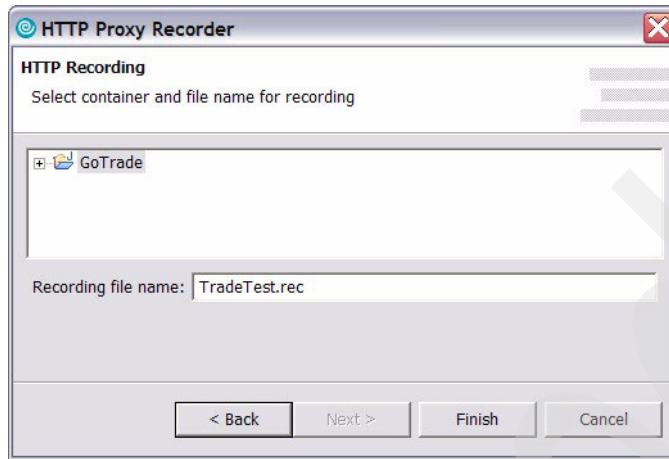


Figure 10-10 Create new HTTP recording dialog

5. The HTTP Recorder will now start up and when it finishes initializing, it brings up an Internet Explorer® window introduction page stating that you should not record confidential data.
6. Type in `http://http1.itso.ibm.com/trade/scenario` at the address field and press Enter. A page similar to the one in Figure 10-11 on page 411 is displayed.

Normally, users do not work with the same set of data on each interaction with your application. During a stress test, this should also be true of your simulated users. It is easier to make your simulated users appear to be working with varied data if the stress testing tool supports data input from lists, files, or a database. This type of input is provided by the preceding URL; it is designed to exercise different pages within Trade 6.1.

At this point, you have finished recording and can start customizing the recorded script.

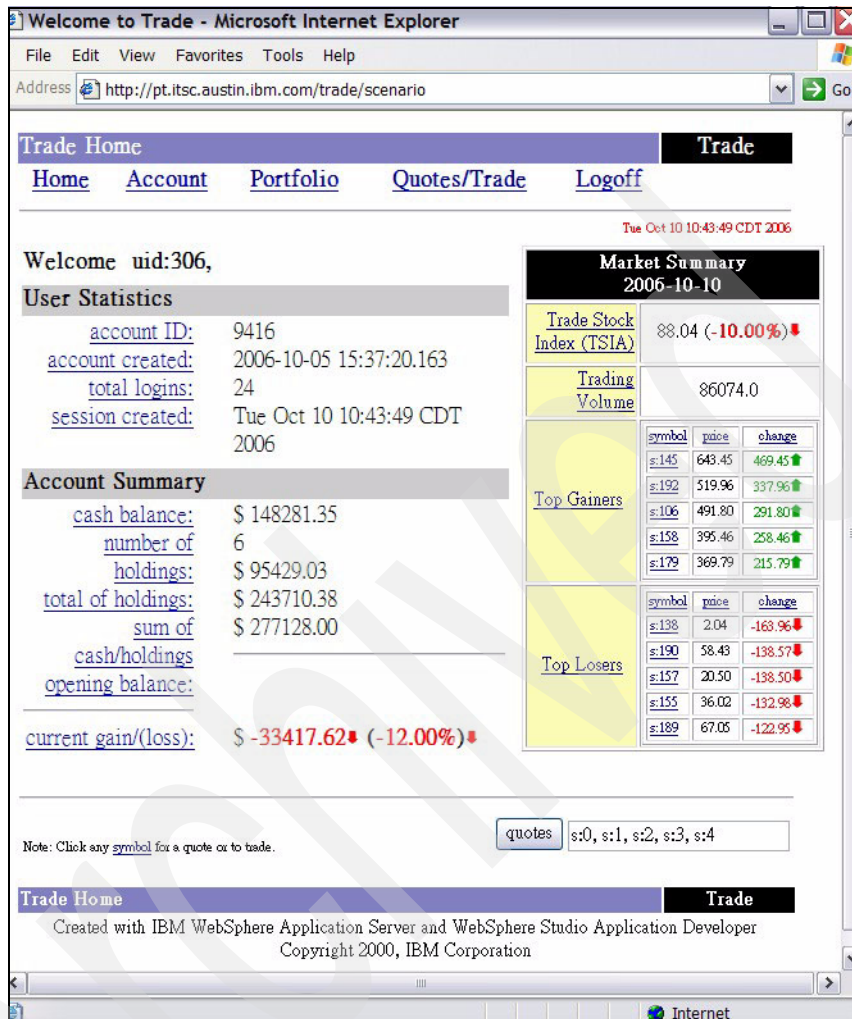


Figure 10-11 Internet Explorer recording

7. Close Internet Explorer; this displays the recorded script inside your workspace.

Creating a performance schedule

After setting up your project and recording a scenario, you can create a performance schedule. In a performance schedule, you basically assemble multiple scenarios with loops, delays, and other items to design a performance workload that matches the scenario that you want to simulate as closely as possible.

Follow these steps:

1. Click **File** -> **New** -> **Performance Schedule** to create a new test schedule called Trade6Schedule in your workspace. Then click **Finish**, as shown in Figure 10-12.

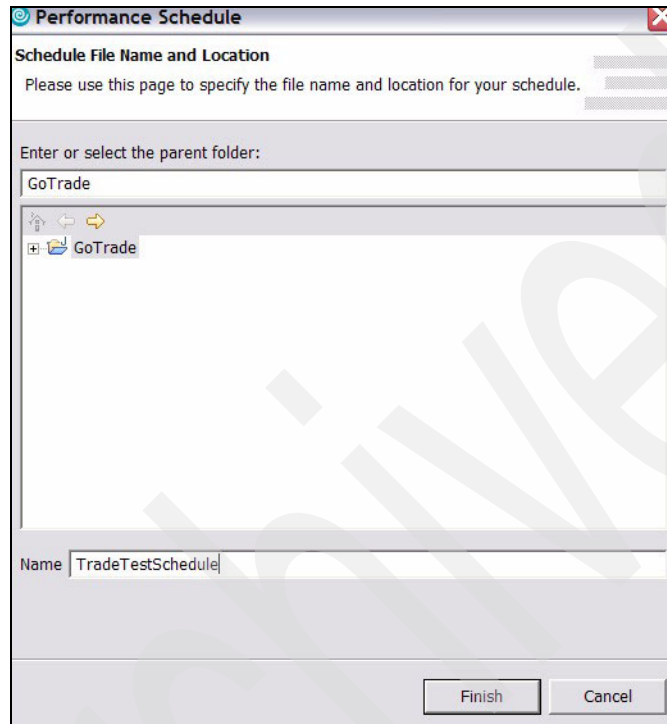


Figure 10-12 Create new Performance Schedule wizard

2. Right-click **User Group 1**, then click **Add** -> **Loop**.

Note: Loops let you repeat tests and run a test at a rate that you specify.

3. Click **Loop** and set the Number of iterations to 200.
4. Right-click **Loop** -> **Add**-> **Test**, select **GoTrade** from the list and click **OK**.
5. Click **TradeTestScheldule**, and change the Number of users under the **Schedule Elements Details** to 1000 as shown in Figure 10-13 on page 413.

Normally, you would specify how many simulated users are running each script or set of tasks, allowing you to vary the number of simulated users over time. This enables you to start with a small number of users and ramp up slowly to a greater number of users.

However, you have not set the delay between starting each user in order to avoid a ramp-up time and keep the test schedule short.

6. Click **Stop running the schedule after an elapsed time** and change it to Stop after: 15 minutes, as shown in Figure 10-13.

Schedule Element Details

TradeTestSchedule

General

Schedule name: TradeTestSchedule

User Load

Number of users: 1000

☐ Add a delay between starting each user

Delay: 0 milliseconds

☒ Stop running the schedule after an elapsed time

Stop after: 15 minutes

Think Time

Modify the duration of think time delays:

Use the recorded think time

☒ Limit think times to a maximum value

Maximum think time: 1 seconds

Figure 10-13 Trade test schedule details

7. Change the **Maximum think time** to 1 seconds.

Real-world users do not request one Web page immediately after another. There are generally delays between viewing one Web page and the next. The term “think time” is the standard way of expressing the addition of a delay into a test script to more realistically simulate user behavior.

Many stress testing tools support randomly generated think times based on a statistical distribution. In this case, the think time as deliberately set to a very small value in order to overload the application.

8. Click **File -> Save** to save the changes.

Run a performance schedule

Now that everything is recorded and set up, you can run your tests.

1. Right-click **TradeTestSchedule**, then click **Run -> Performance Schedule**.
2. The workspace displays a Performance Summary Report similar to the one in Figure 10-14. This report is continuously updated during the test until all virtual test users have finished executing.

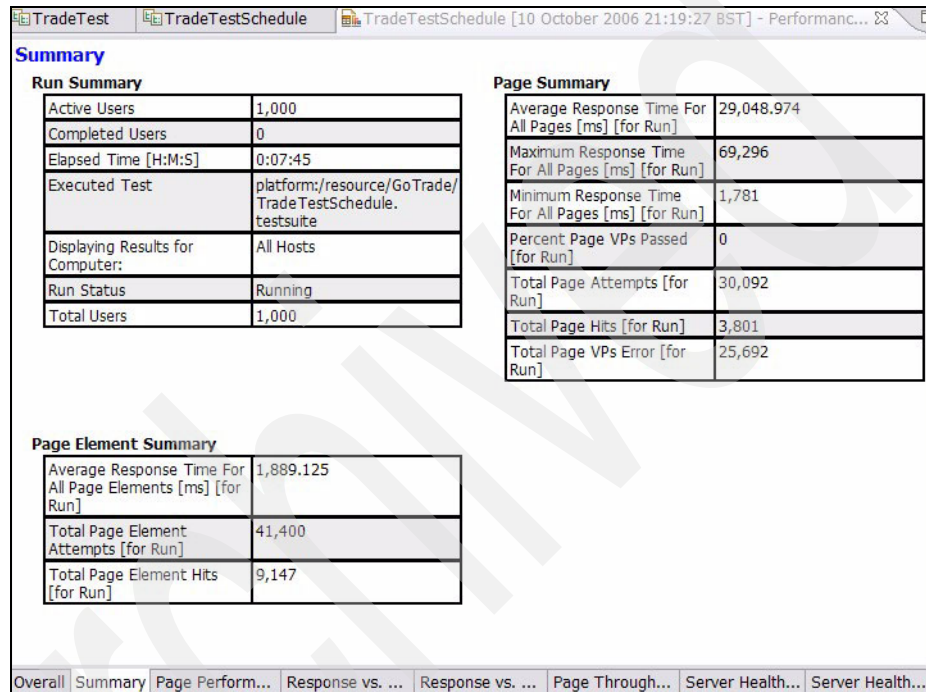


Figure 10-14 Performance Schedule Summary Report

One of the reports you can now look at to review the test results is the Response versus Time report shown in Figure 10-15 on page 415. This report plots the average response time over time during your test execution.

A typical workstation running a stress testing tool will likely begin bottlenecking when approximately 200 virtual users are running. To simulate a greater number of users, you can distribute the stress testing load across multiple workstations. Many of the available stress testing tools support distribution of load and you will certainly want this feature for large scale stress testing. However, we found that it was unnecessary to spread this load across more than one workstation.

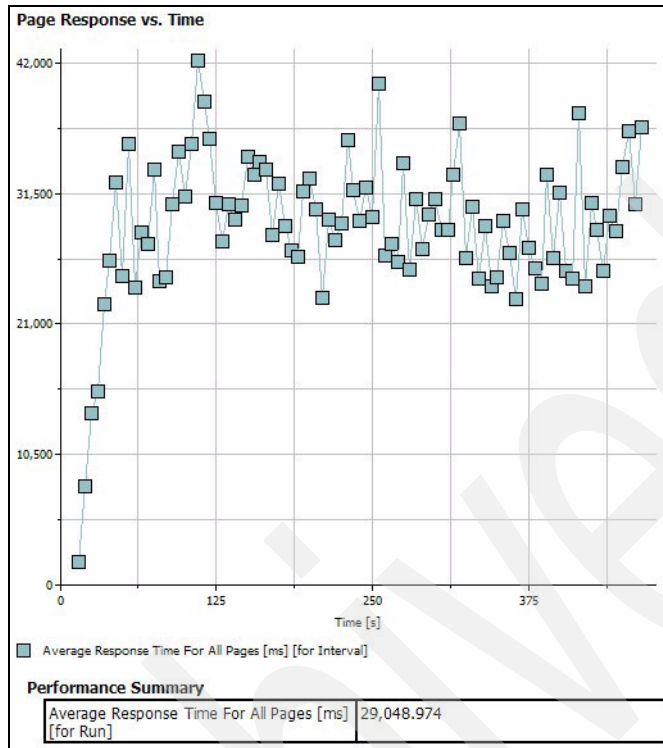


Figure 10-15 Rational Performance Tester response versus time graph

At this point, you can load test the scenario.

10.5.4 Scenario testing

The scenario was initially configured with the LPAR having just one physical processing unit. In our case, we used the Web SM tool to configure this. Figure 10-16 on page 416 shows the Web SM add processor resources wizard, and it illustrates the current processing units being set to one. It also shows there were 1.8 processing units available for this LPAR, which we will be adding to this LPAR to increase its processor resources at a later stage.

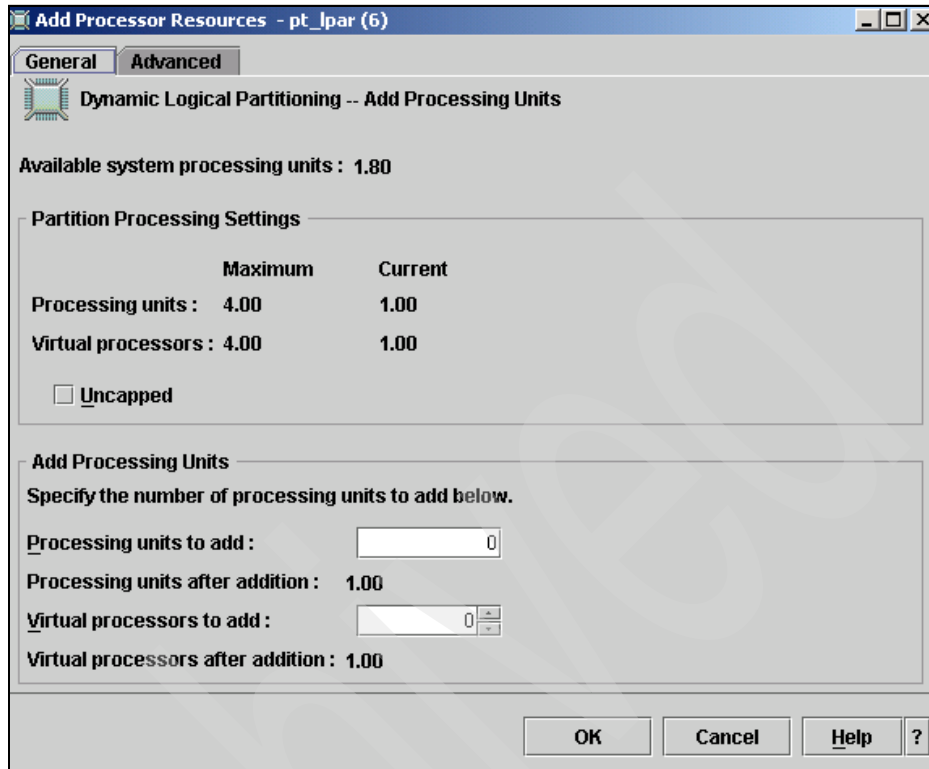


Figure 10-16 Web SM Add Processor Resources wizard

You can also use the **bindprocessor** command to show the current processor resources. Figure 10-17 shows the output from the **bindprocessor -q** command. Processors 0 and 1 represent processor identifiers and are assigned to this LPAR.

```
[0:root@pt:]/home/root # bindprocessor -q
The available processors are: 0 1
[0:root@pt:]/home/root #
```

Figure 10-17 The **bindprocessor** command showing two available processors

After the test passed its ramp-up phase, we began monitoring processor utilization with either **topas** or **lparstat**.

The **topas** command reports selected statistics about the activity on the local system. The program extracts statistics from the system with an interval specified by the monitoring interval in seconds. The output of this command has a cross-partition display. This panel displays metrics similar to the **lparstat** command for all the AIX partitions it can identify as belonging to the same

hardware platform. It actually display more performance metrics than we required.

Note: `topas` must be restarted if the processing units has changed for an LPAR. It does not recognize changes and will continue to report incorrect performance data.

`lparstat` provides a display of current LPAR-related parameters, as well as utilization statistics for the LPAR. An interval mechanism retrieves numbers of reports at a certain interval. We were only interested in processor capacity, and `lparstat` is better suited to this task.

Figure 10-18 shows the output from `lparstat 1`. It clearly shows the system is constantly under load at around 100%. The parameter `lcpu=2` indicates there are currently two processors assigned to this LPAR. This equates to one processing unit.

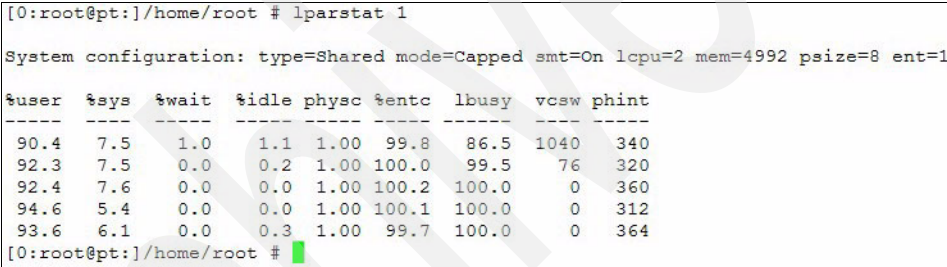


Figure 10-18 `lparstat` showing CPU utilization for one processing unit

After completing the load test, a set of test results were generated, including spreadsheets, charts, and reports for analysis.

The summary for the Web pages that we extrapolated is shown in Table 10-5.

Table 10-5 Performance summary for Web pages

Average Response Time For All Pages [ms] [for Run]	66,450.057
Maximum Response Time For All Pages [ms] [for Run]	273,812
Minimum Response Time For All Pages [ms] [for Run]	9,625
Percent Page VPs Passed [for Run]	0
Total Page Attempts [for Run]	12,730
Total Page Hits [for Run]	12,180

Total Page VPs Failed [for Run]	20
Total Page VPs Error [for Run]	2

The metric we were particularly interested in was Average Response Time For All Pages, which showed 66.45 seconds. Also notice there were not many failures, so the test was conducted successfully.

We ran the test again, but with increased processing resource. Figure 10-19 shows the available processing units from Web SM increased from 1 to 2.8, but the virtual processors increased to 4.

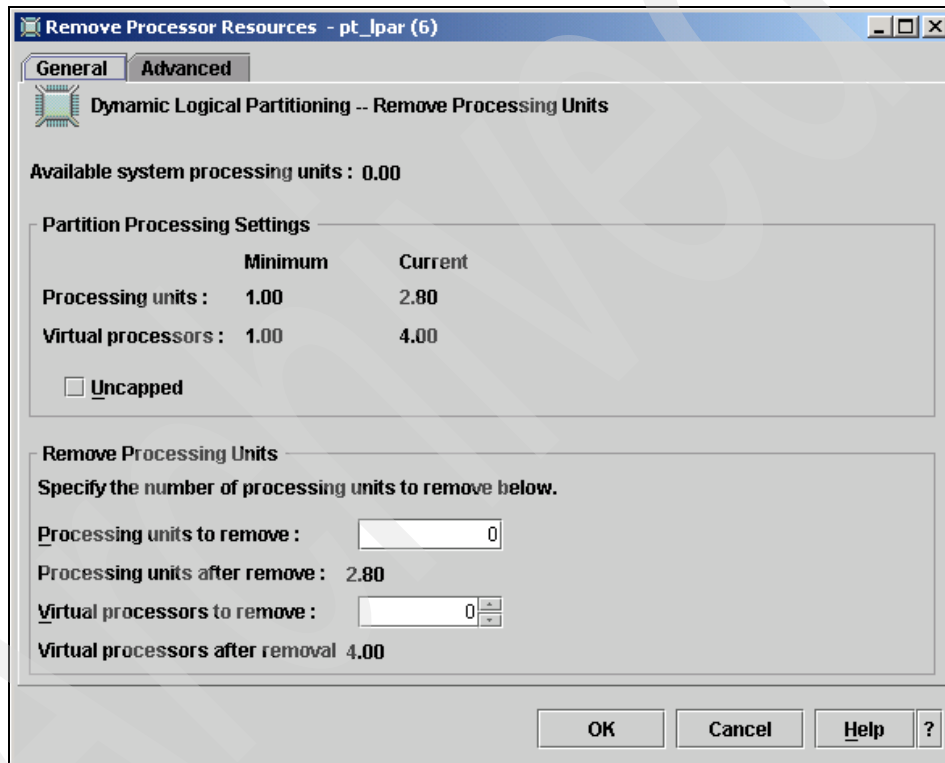


Figure 10-19 Web SM Remove Processor Resources wizard

When the **bindprocessor** command was invoked it returned eight available processors, as shown in Figure 10-20 on page 419. This illustrated that 2.8 processing units were being spread across these eight processors.

```
[0:root@pt:]/home/root # bindprocessor -q
The available processors are: 0 1 2 3 4 5 6 7
[0:root@pt:]/home/root #
```

Figure 10-20 The bindprocessor command showing eight available processors

Again, after the load test settled, we were able to monitor processor utilization. Figure 10-21 shows that processor utilization had dropped significantly to around 40%. The lcpu metric now showed eight available processors.

```
[0:root@pt:]/home/root # lparstat 1
System configuration: type=Shared mode=Capped smt=On lcpu=8 mem=4992 psize=8 ent=2.80

%user  %sys  %wait  %idle  physc  %entc  lbusy  vcs  phint
-----
31.9    5.2    0.7    62.1   1.11   39.8   23.6   2495  149
31.3    5.1    1.2    62.4   1.09   39.1   22.5   2878  127
39.7    3.2    0.0    57.2   1.32   47.2   21.5   1844  80
47.5    6.1    0.8    45.5   1.58   56.6   35.1   1980  200
29.8    4.6    1.9    63.7   1.06   37.9   21.5   2430  116
```

Figure 10-21 lparstat showing CPU utilization for 2.8 processing units

This significant drop in processor utilization resulted in an improved average response time of 53.03 seconds, as shown in Table 10-6. Although the processor utilization dropped more than 50%, we did not see similar improvement with the response time. Obviously, further performance tuning was required. However, we were only interested in demonstrating the effect of dynamically allocating more processor resources to our LPAR and believed we had achieved that.

Table 10-6 Second test - performance summary for Web pages

Average Response Time For All Pages [ms] [for Run]	53,034.346
Maximum Response Time For All Pages [ms] [for Run]	221,766
Minimum Response Time For All Pages [ms] [for Run]	6,266
Percent Page VPs Passed [for Run]	0
Total Page Attempts [for Run]	16,378
Total Page Hits [for Run]	15,879
Total Page VPs Error [for Run]	1

Figure 10-22 shows a comparison of the page response versus time for the two tests conducted. These charts illustrate that the tests were consistent and repeatable.

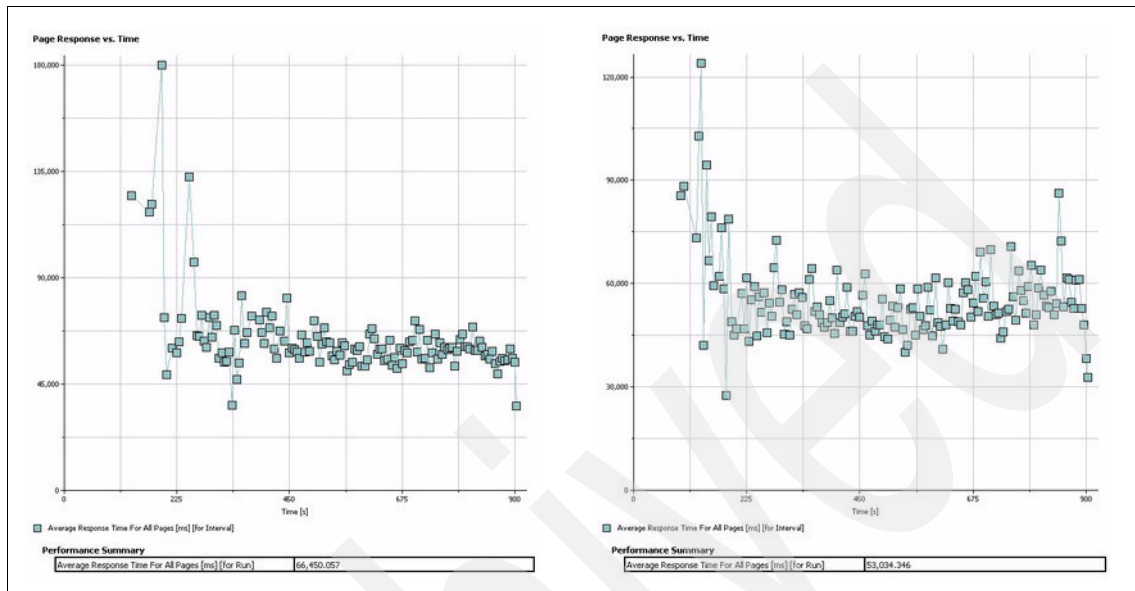


Figure 10-22 Page response time comparison for 1 and 2.8 processing unit tests

Dynamic testing

The preceding tests were conducted independently, with processors units assigned up front. There were no changes to the resources throughout the tests. We carried out the testing in this way to obtain an accurate measurement of response time for comparison.

However, we also wanted to determine how our scenario would respond to dynamic allocation of resources. This effect can be observed by repeating the test with a longer test duration, so we set the schedule to stop after 20 minutes. This would give ample time for ramp up and a stable period for testing the dynamic allocation of processing units.

So, we reset the processing units back to 1 using the Web SM tool. The test was then started and processor utilization was monitored with **lparstat**. After the test had passed the ramp-up phase and settled down, we dynamically assigned more processing units to the LPAR. We made similar observations, in which processing utilization dropped to around 40% after we increased the processing units to 2.8, as before.

Improved response time was observed, as shown in the average page response time chart displayed in Figure 10-23.

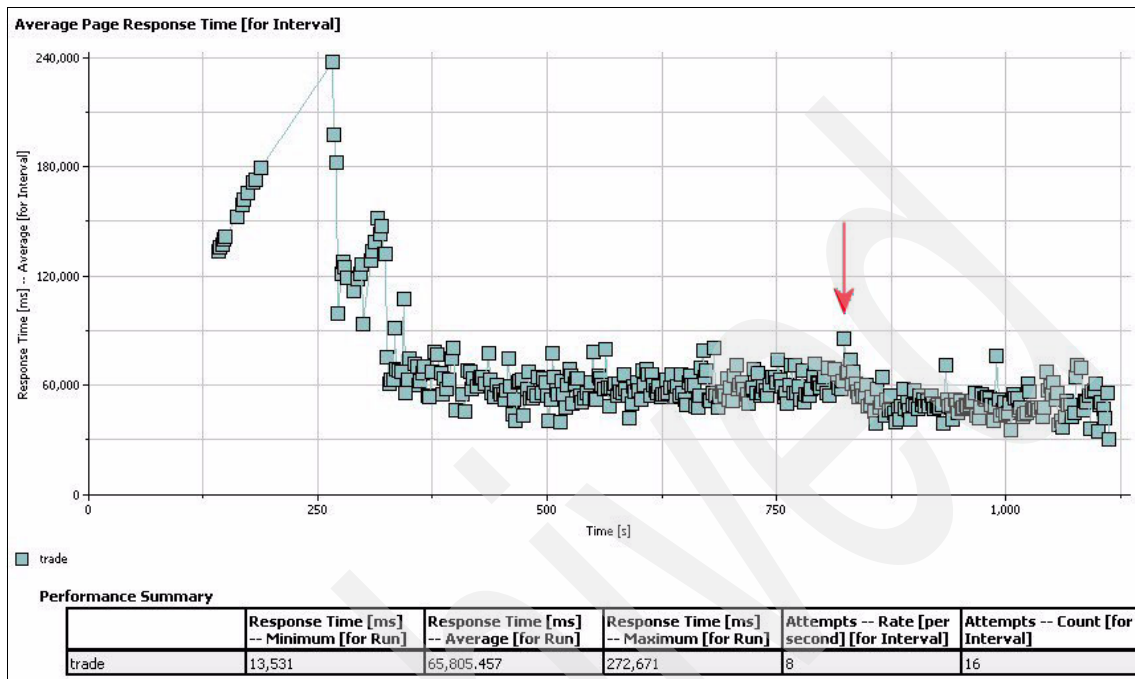


Figure 10-23 Impact of dynamic resource change to page response

The arrow on the chart indicates the point where more processing units were allocated to this LPAR. Although the improvement in response time was not as significant as the processing utilization, it does show the trend and the effect of dynamically allocating more processing units “on the fly”.

This scenario has shown that processor units can be dynamically added to a running WebSphere environment for improved processor utilization and response time. The improvement in response time will vary from application to application, and our Trade 6.1 application has room for improvement—but it does illustrate that WebSphere Application Server can take advantage of extra processor capacity without any changes.

As you can see, it is possible to monitor processor utilization in a WebSphere Application Server environment and run a script that will add more processing units to the LPAR when processor utilization hits a certain threshold. It is also possible to remove processing units if needed in a different LPAR. You can leverage the virtualization capability of POWER5 in an On Demand WebSphere Application Server environment.

Sample files

This appendix includes the following sample files:

- ▶ “WebSphere: responsefile.nd.txt” on page 424
- ▶ “Update Installer: response.txt” on page 452
- ▶ “WebSphere V6.1 Fix Pack 2: fp2.response.txt” on page 455
- ▶ “Trade 6.1 Installation script: trade.jacl” on page 456
- ▶ “CSM adapter definition file: p550q_lpar_adapters” on page 471

These files are also available for download, and the downloading instructions are explained in Appendix B, “Additional material” on page 473.

WebSphere: responsefile.nd.txt

Example: A-1 WebSphere response file

```
#####
#
# V6.1 InstallShield Options File
#
# Wizard name: Install
# Wizard source: setup.jar
#
# This file can be used to configure Install with the options specified below
# when the wizard is run with the "-options" command line option. Read each
# setting's documentation for information on how to change its value.
# Enclose all values within a single pair of double quotes.
#
# A common use of an options file is to run the wizard in silent mode. This lets
# the options file author specify wizard settings without having to run the
# wizard in graphical or console mode. To use this options file for silent mode
# execution, use the following command line arguments when running the wizard:
#
# -options "D:\installImage\WebSphere Application Server \responsefile.nd.txt"
# -silent
#
#####

#####
#
# License Acceptance
#
# Valid Values:
# true - Accepts the license. Will install the product.
# false - Declines the license. Install will not occur.
#
# If no install occurs, this will be logged to a temporary log file in the
# user's temporary directory.
#
# By changing the silentInstallLicenseAcceptance property in this response file
# to "true", you agree that you have reviewed and agree to the terms of the
# IBM International Program License Agreement accompanying this program, which
# is located at CD_ROOT\was.primary.pak\repository\legal\lafiles. If you do not
# agree to these terms, do not change the value or otherwise download, install,
# copy, access, or use the program and promptly return the program and proof of
# entitlement to the party from whom you acquired it to obtain a refund of the
# amount you paid.
#
-OPT silentInstallLicenseAcceptance="true"

#####
#
# NonRoot Install Settings
#
```

```

# The option indicates whether you accept the limitations associated with installing
# as a non-root user, as follows:
#
#   - Creation of a Windows or Linux service for WebSphere Application Server
#   - Native registration with the operating system
#
# Port conflicts may occur with other installations of WebSphere Application Server that
# are not registered with the operating system.
#
# See the information center,
# (http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/rins\_nonroot.html)
# for more information on performing these installation actions after installation and
# avoiding port conflicts.
#
# Valid Values:
#   true - Accepts the limitations. Will install the product.
#   false - Do not accept the limitations. Install will not occur.
#
# Uncomment the following only if you are installing as a non-root user. Specify
# one of the valid options listed above before proceeding to install:
#
#-OPT allowNonRootSilentInstall="true"

#####
#
# Operating System Prerequisite Checking
#
# If you want to disable operating system prerequisite checking, uncomment
# the following line. This will notify the installer to continue with
# the installation and log the warnings even though the prerequisite checking
# has failed.
#
#-OPT disableOSPrereqChecking="true"

#####
#
# Non-blocking Prerequisite Checking
#
# If you want to disable non-blocking prerequisite checking, uncomment
# the following line. This will notify the installer to continue with
# the installation and log the warnings even though the prerequisite checking
# has failed.
#
#-OPT disableNonBlockingPrereqChecking="true"

#####
#
# Install Type Settings
#
# The installType option designates the type of installation that will be
# performed.
#
# The default installType option is to install a new copy of WebSphere

```

```

# Application Server.
#
# Valid Values:
#   installNew - default value, installs a new copy.
#   addFeature - add features to an existing installation.

#####
#
# Install a New Copy
#
# To install a new copy, be sure that the installLocation option is set to a
# new install location.
#

-OPT installType="installNew"

#####
#
# Incremental Install
#
# If you are installing additional features on top of an existing installation,
# (e.g. incremental install), uncomment the following line. This will notify
# the installer that you are doing an incremental install.
#
#-OPT installType="addFeature"
#
# Define the new features you want to install on top of an existing WebSphere
# Application Server. The only additional feature available in WebSphere
# Application Server is Application Server Samples.
#
# Ensure the feature option below is set to "samplesSelected" (-OPT
# feature="samplesSelected")
# and the installLocation option is set to an "existing" WebSphere Application
# Server install location.
#

#####
#
# Create Profile for an Existing V6.1 Installation
#
# Valid values:
#   true - creates a profile for an existing installation
#   false - does not create a profile
#
# To create a profile for an existing installation, uncomment the following
# entry. Comment out the "installType" option above since "installType"
# and "createProfile" options cannot be specified at the same time.
#
# Be sure the installLocation option is set to your existing install location.
#
#-OPT createProfile="true"

#####
#

```

```

# "Application Server samples" feature
#
# The selection state of the "Application Server samples" feature.
#
# Valid options:
#
#   samplesSelected - Indicates that the feature is selected for installation.
#   noFeature - Indicates that the feature is not selected for installation,
#               this is the default option.
#
# For example, to select "Application Server samples" for installation, use
#
#   -OPT feature="samplesSelected"
#
# Note if feature="samplesSelected" and PROF_enableAdminSecurity="true",
# you must provide a password by uncommenting and specifying a value for
# PROF_samplesPassword in Administrative Security section below
#
-OPT feature="noFeature"

#####
#
# Install Location
#
# The install location of the product. Specify a valid directory into which the
# product should be installed. If the directory contains spaces, enclose it in
# double-quotes as shown in the Windows example below. Note that spaces in the
# install location is only supported on Windows operating systems. Maximum path
# length is 60 characters for Windows 2000 and Windows XP.
#
# Below is the list of default install locations for each supported operating
# system when you're installing as a root user. By default, in this response
# file, the Windows install location is used. If you want to use the default
# install location for another operating system, uncomment the appropriate
# default install location entry and then comment out the
# Windows operating system entry below.
#
# AIX Default Install Location:
#
# -OPT installLocation="/usr/IBM/WebSphere/AppServer"
#
# HP-UX, Solaris or Linux Default Install Location:
#
# -OPT installLocation="/opt/IBM/WebSphere/AppServer"
#
# i5OS Default Install Location:
#
# -OPT installLocation="/QIBM/IBM/WebSphere/AppServer/V61/<productOffering>"
#
# Windows Default Install Location:
#
# -OPT installLocation="C:\Program Files\IBM\WebSphere\AppServer"
#
# If you are installing as non-root user on Unix or non-administrator on

```

```

# Windows, the following default install locations are suggested. Be sure you
# have write permission for the install location chosen.
#
# AIX Default Install Location:
#
#-OPT installLocation="<user's home>/IBM/WebSphere/AppServer"
-OPT installLocation="/usr/IBM/WebSphere/AppServer"
#
# HP-UX, Solaris or Linux Default Install Location:
#
#-OPT installLocation="<user's home>/IBM/WebSphere/AppServer"
#
# Windows Default Install Location:
#
#-OPT installLocation="C:\IBM\WebSphere\AppServer"

#####
#
# Profile Creation Selection
#
# Use this option to indicate the type of profile you would like to create.
#
# Valid Values:
#
#   cell - two profiles will be created, one with a deployment manager and
#         another with a managed node that is pre-federated into the cell.
#   deploymentManager - a profile will be created with a deployment manager.
#   standAlone - a profile will be created with a stand alone Application
#               server.
#   custom - a profile will be created with an empty node
#   none - a profile will not be created during installation.
#
# This option is required for the installation on ND. Do not comment it out!
#
# For example, to select "cell" profile creation for installation, use
#
#-OPT profileType="custom"
-OPT profileType="cell"

#####
#
# Administrative Security
#
# Choose whether to enable Administrative security during the installation
# process.
#
# If profileType="custom", Administrative security should be disabled, use
#-OPT PROF_enableAdminSecurity="false"
#
# Valid Values:
#   true - Administrative security is enabled, user name and
#         password required.
#   false - Administrative security is not enabled.

```



```

#
#-OPT PROF_enableAdminSecurity="true"
-OPT PROF_enableAdminSecurity="false"

#####
#
# Security Options
#
# The following two options should only be uncommented when
# PROF_enableAdminSecurity="true".
#
# If PROF_enableAdminSecurity="false", the following entries will be
# disregarded.
#
# If PROF_enableAdminSecurity="true", the following entries are required and must
# be filled out before proceeding with silent installation.
#
# Specify the user name for administrative security
#
# Valid Values:
# a character string - do not use the following characters:
# /, \, *, , , , : , ; , = , + , ? , | , < , > , & , % , ' , " , ]]> , # , $ , ^ , { , }
# Note: a period (.) is not valid if it is the first character.
#
-OPT PROF_adminUserName=
#
# Specify the password for the user specified in PROF_adminUserName
#
# Valid Values:
# a character string
#
-OPT PROF_adminPassword=
#
# The following option should only be uncommented when feature="samplesSelected" and
# PROF_enableAdminSecurity="true"
# Specify the password for samples user
#
# Valid Values:
# a character string
#
-OPT PROF_samplesPassword=

#####
#####
#
# The following options are related to creating profiles and can only be used
# when installType="installNew" or createProfile="true". The options are dependent
# on the profileType option specified in the previous section.
#

#####
#####
#

```

```

# Cell Profile
#
# If profileType="cell", use the following profile creation options:
#
#####
#
# Deployment Manager Profile name
#
# Specify the name of the profile for the deployment manager. The profile
# name must be unique for this WebSphere Application Server installation.
#
# Valid Values:
#   a character string - do not use the following characters:
#   /, \, *, , , ., :, ;, =, +, ?, |, <, >, &, %, ', ", ]]>, #, $, ^, {, }
# Note: a period (.) is not valid if it is the first character.
#
-OPT PROF_dmgrProfileName=Dmgr

#####
#
# Application Server Profile name
#
# Specify the name of the profile for the application server. The profile
# name must be unique for this WebSphere Application Server installation.
#
# Valid Values:
#   a character string - do not use the following characters:
#   /, \, *, , , ., :, ;, =, +, ?, |, <, >, &, %, ', ", ]]>, #, $, ^, {, }
# Note: a period (.) is not valid if it is the first character.
#
-OPT PROF_appServerProfileName=AppSrv

#####
#
# Profile path
#
# Specify a valid directory to contain the files that define the run-time environment,
# such as commands, configuration files, and log files.
#
# Valid Values: An empty directory, the user must have proper permissions to the
# directory,
# there must be adequate disk space, total path cannot be greater than 80 (windows)
#
-OPT PROF_profilePath=

#####
#
# Default Profile
#
# Uncomment the following line to make this profile the default target of commands
# that do not use their profile parameter.
# Note that the first profile created for an installation is always marked as
# the default profile.
#

```

```

# Valid Value:
#     true - make this the default profile.
#
#-OPT PROF_isDefault="true"

#####
#
# Host name
#
# Specify the host name for the Deployment Manager. The host name is the domain
# name system (DNS) name (short or long) or the IP address of this computer.
# Valid Values: a valid hostname or IP address
#
-OPT PROF_hostName=pt

#####
#
# Deployment Manager Node name
#
# Specify the node name for the deployment manager. Node names within a cell
# must be unique.
#
# Valid Values:
#   a character string - do not use the following characters:
#   /, \, *, , , , : , ; , = , + , ? , | , < , > , & , % , ' , " , ]]> , # , $ , ^ , { , }
# Note: a period (.) is not valid if it is the first character.
#
-OPT PROF_nodeName=ITS0CellManager

#####
#
# Application Server Node name
#
# Specify the node name for the Application Server. Node name under one cell
# has to be unique.
#
# Valid Values:
#   a character string - do not use the following characters:
#   /, \, *, , , , : , ; , = , + , ? , | , < , > , & , % , ' , " , ]]> , # , $ , ^ , { , }
# Note: a period (.) is not valid if it is the first character.
#
-OPT PROF_appServerNodeName=PTNode

#####
#
# Cell name
#
# Specify the cell name for the profile to be created.
#
# Valid Values:
#   a character string - do not use the following characters:
#   /, \, *, , , , : , ; , = , + , ? , | , < , > , & , % , ' , " , ]]> , # , $ , ^ , { , }
# Note: a period (.) is not valid if it is the first character.
#

```

-OPT PROF_cellName=ITS0Cell

```
#####  
#  
# Starting Port  
#  
# Specify the starting port number for generating all ports for the profile.  
# Do not use this parameter with the PROF_portsFile or PROF_defaultPorts parameters.  
#  
# Valid values: a positive integer port value, within the valid port range  
#  
#-OPT PROF_startingPort=  
  
#####  
#  
# Node Starting Port  
#  
# Specify the starting port number for the node portion of the cell.  
# Do not use this parameter with the PROF_nodePortsFile parameter.  
#  
# Valid values: a positive integer port value, within the valid port range  
#  
#-OPT PROF_nodeStartingPort=  
  
#####  
#  
# Port File  
#  
# Specify the path to a property file containing the desired port values for  
# the new profile. Do not use this parameter with the PROF_startingPort or  
# PROF_defaultPorts parameters.  
#  
# Valid values: A fully qualified path to a valid ports property file  
#  
#-OPT PROF_portsFile=  
  
#####  
#  
# Node Port File  
#  
# Specify the path to a property file containing the desired port values for  
# the new profile. Do not use this parameter with the PROF_startingPort or  
# PROF_defaultPorts parameters.  
#  
# Valid values: A fully qualified path to a valid ports property file that defines  
# port settings for the node portion of the cell.  
# Do not use this parameter with the PROF_nodeStartingPort parameter.  
#  
#-OPT PROF_nodePortsFile=  
  
#####  
#  
# Default Ports  
#
```

```

# Uncomment the following line to assign the default port values.
# Otherwise unique port values will be assigned.
# Do not use this parameter with the PROF_portsFile or PROF_startingPort parameters.
#
# Valid value:
#   true - use WebSphere Application Server default ports.
#
-OPT PROF_defaultPorts="true"

#####
#
# Validate Ports
#
# Uncomment the following line to validate the port values to ensure they are
# not reserved or in use. Otherwise, no port validation checking will occur.
#
# Valid value:
#   true - enables port validation.
#
-OPT PROF_validatePorts="true"

#####
#
# WinService Check
#
# Specify whether you want to run this server as a windows service
#
# Valid values:
#   true - run as Windows service.
#   false - do not run as Windows service.
#
#-OPT PROF_winserviceCheck="true"

#####
#
# WinService Account Type
#
# Specify the type of the owner account of the Windows service to create.
# Uncomment the following ONLY if PROF_winserviceCheck="true"
#
# Valid values: specifieduser or localsystem
#
#-OPT PROF_winserviceAccountType=

#####
#
# WinService User Name
#
# Specify the user name for the windows service. Uncomment the
# following ONLY if PROF_winserviceCheck="true"
#
# Valid values: a valid user name for the current system
#
#-OPT PROF_winserviceUserName=

```

```
#####
#
# WinService Password
#
# Specify the password for the user specified by the winserviceUserName parameter
# Uncomment the following ONLY if PROF_winserviceCheck="true"
#
# Valid Values:
#   a character string - do not use the following characters:
#   /, \, *, ., : , ; , = , + , ? , | , < , > , & , % , ' , " , ]]> , # , $ , ^ , { , }
# Note: a period (.) is not valid if it is the first character.
#
#-OPT PROF_winservicePassword=

#####
#
# WinService Startup Type
#
# Specify the start up method for the windows service
# Uncomment the following ONLY if PROF_winserviceCheck="true"
#
# Valid values:
#   manual - windows service must be started manually.
#   automatic - windows service will start automatically after reboot.
#   disabled - service is disabled.
#
#-OPT PROF_winserviceStartupType="automatic"

#####
#
# LinuxService Check
#
# Specify whether you want to run this server as a Linux service.
#
# Note that the root permission is required to enable Linux service.
#
# Valid values:
#   true - run as a Linux service.
#   false - do not run as a Linux service.
#
#-OPT PROF_enableService="true"

#####
#
# LinuxService User Name
#
# Specify the name of the user you wish this service to be run as. Uncomment the
# following ONLY if PROF_enableService="true". Linux services can only be created
# from profile creation if the current user is the root user.
#
# Valid values: a valid Linux user name
#
#-OPT PROF_serviceUserName=
```

```
#####
#
# WebServer Check
#
# Specify whether you wish to set the web server definition.
#
# Valid values:
#   true - enable the creation of a webserver definition.
#   false - do not enable the creation of a webserver definition.
#
-OPT PROF_webServerCheck="true"

#####
#
# WebServer Type
#
# Specify the type of web server you are creating.
# Uncomment the following if and ONLY if PROF_webServerCheck="true".
#
# Valid values (case sensitive): IHS | IIS | SUNJAVASYSTEM | DOMINO | APACHE |
# HTTPSERVER_ZOS
#
-OPT PROF_webServerType=IHS

#####
#
# WebServer OS
#
# Specify the operating system of the system where the web server is
# installed. Uncomment the following if and ONLY if PROF_webServerCheck="true".
#
# Valid values: linux | windows | aix | hpux | solaris | os390 | os400
#
-OPT PROF_webServerOS=

#####
#
# WebServer Name
#
# Specify the name of the web server. Uncomment the following if and ONLY if
# PROF_webServerCheck="true".
#
# Valid Values:
#   a character string - do not use the following characters:
#   /, \, *, , , , : , ; , = , + , ? , | , < , > , & , % , ' , " , ]]> , # , $ , ^ , { , }
#   Note: a period (.) is not valid if it is the first character.
#
-OPT PROF_webServerName=webserver1

#####
#
# WebServer Hostname
#
```

```

# Specify the hostname of the system with the web server. Uncomment the following
# if and ONLY if PROF_webServerCheck="true".
#
# Valid values: a valid hostname or IP address
#
-OPT PROF_webServerHostname=br

#####
#
# WebServer Port
#
# Specify the port from which the web server can be accessed. Uncomment the
# following if and ONLY if PROF_webServerCheck="true".
#
# Valid values: a valid port number
#
-OPT PROF_webServerPort=80

#####
#
# WebServer Install Path
#
# Specify the installation path of the web server (local or remote).
# Uncomment the following if and ONLY if PROF_webServerCheck="true".
#
# Valid values: a valid directory path
#
-OPT PROF_webServerInstallPath=/usr/IBM/HTTPServer

#####
#
# WebServer Plugin Path
#
# Specify the path to the plugins that will be used by this web server.
# Uncomment the following if and ONLY if PROF_webServerCheck="true".
#
# Valid values: a valid directory path
#
-OPT PROF_webServerPluginPath=/usr/IBM/HTTPServer/Plugins

#####
# Omit Action
#
# Use this option to omit the config action specified in the Application Server
# profile.
#
# Valid values: one of the following optional config actions
#   deployAdminConsole
#   defaultAppDeployAndConfig
#   samplesInstallAndConfig
#
#-OPT PROF_omitAction=

#####

```



```

#
# Deployment Manager Profile
#
# if profileType="deploymentManager", use the following profile creation
# options:
#
#####
#
# Profile name
#
# Specify the name of the profile for the Deployment Manager. The profile
# name must be unique for this WebSphere Application Server installation.
#
# Valid Values:
#   a character string - do not use the following characters:
#   /, \, *, , , , : , ; , = , + , ? , | , < , > , & , % , ' , " , ]]> , # , $ , ^ , { , }
# Note: a period (.) is not valid if it is the first character.
#
#-OPT PROF_profileName=

#####
#
# Profile path
#
# Specify a valid directory to contain the files that define the run-time environment,
# such as commands, configuration files, and log files.
#
# Valid Values: An empty directory, the user must have proper permissions to the
# directory,
# there must be adequate disk space, total path cannot be greater than 80 (windows)
#
#-OPT PROF_profilePath=

#####
#
# Default Profile
#
# Uncomment the following line to make this profile the default target of commands
# that do not use their profile parameter.
# Note that the first profile created for an installation is always marked as
# the default profile.
#
# Valid Value:
#   true - make this the default profile.
#
#-OPT PROF_isDefault="true"

#####
#
# Host name
#
# Specify the host name for the Deployment Manager. The host name is the domain
# name system (DNS) name (short or long) or the IP address of this computer.
# Valid Values: a valid hostname or IP address

```

```

#
#-OPT PROF_hostName=

#####
#
# Deployment Manager Node name
#
# Specify the node name for the deployment manager. Node names within a cell
# must be unique.
#
# Valid Values:
#   a character string - do not use the following characters:
#   /, \, *, , , : , ; , = , + , ? , | , < , > , & , % , ' , " , ]]> , # , $ , ^ , { , }
# Note: a period (.) is not valid if it is the first character.
#
#-OPT PROF_nodeName=

#####
#
# Cell name
#
# Specify the cell name for the profile to be created.
#
# Valid Values:
#   a character string - do not use the following characters:
#   /, \, *, , , : , ; , = , + , ? , | , < , > , & , % , ' , " , ]]> , # , $ , ^ , { , }
# Note: a period (.) is not valid if it is the first character.
#
# -OPT PROF_cellName=

#####
#
# Starting Port
#
# Specify the starting port number for generating all ports for the profile.
# Do not use this parameter with the PROF_portsFile or PROF_defaultPorts parameters.
#
# Valid values: a positive integer port value, within the valid port range
#
#-OPT PROF_startingPort=

#####
#
# Port File
#
# Specify the path to a property file containing the desired port values for
# the new profile. Do not use this parameter with the PROF_startingPort or
# PROF_defaultPorts parameters.
# Valid values: A fully qualified path to a valid ports property file
#
#-OPT PROF_portsFile=

#####
#

```

```

# Default Ports
#
# Uncomment the following line to assign the default port values.
# Otherwise unique port values will be assigned.
# Do not use this parameter with the PROF_portsFile or PROF_startingPort parameters.
#
# Valid value:
#   true - use WebSphere Application Server default ports.
#
#-OPT PROF_defaultPorts="true"

#####
#
# Validate Ports
#
# Uncomment the following line to validate the port values to ensure they are
# not reserved or in use. Otherwise, no port validation checking will occur.
#
# Valid value:
#   true - enables port validation.
#
#-OPT PROF_validatePorts="true"

#####
#
# WinService Check
#
# Specify whether you want to run this server as a windows service
#
# Valid values:
#   true - run as Windows service.
#   false - do not run as Windows service.
#
#-OPT PROF_winserviceCheck="true"

#####
#
# WinService Account Type
#
# Specify the type of the owner account of the Windows service to create.
# Uncomment the following ONLY if PROF_winserviceCheck="true"
#
# Valid values: specifieduser or localsystem
#
#-OPT PROF_winserviceAccountType=

#####
#
# WinService User Name
#
# Specify the user name for the windows service. Uncomment the
# following ONLY if PROF_winserviceCheck="true"
#
# Valid values: a valid user name for the current system

```

```

#
#-OPT PROF_winserviceUserName=

#####
#
# WinService Password
#
# Specify the password for the user specified by the winserviceUserName parameter
# Uncomment the following ONLY if PROF_winserviceCheck="true"
#
# Valid Values:
#   a character string - do not use the following characters:
#   /, \, *, , , , : , ; , = , + , ? , | , < , > , & , % , ' , " , ]]> , # , $ , ^ , { , }
# Note: a period (.) is not valid if it is the first character.
#
#-OPT PROF_winservicePassword=

#####
#
# WinService Startup Type
#
# Specify the start up method for the windows service
# Uncomment the following ONLY if PROF_winserviceCheck="true"
#
# Valid values:
#   manual - windows service must be started manually.
#   automatic - windows service will start automatically after reboot.
#   disabled - service is disabled.
#
# -OPT PROF_winserviceStartupType="automatic"

#####
#
# LinuxService Check
#
# Specify whether you want to run this server as a Linux service.
#
# Note that the root permission is required to enable Linux service.
#
# Valid values:
#   true - run as a Linux service.
#   false - do not run as a Linux service.
#
#-OPT PROF_enableService="true"

#####
#
# LinuxService User Name
#
# Specify the name of the user you wish this service to be run as. Uncomment the
# following ONLY if PROF_enableService="true". Linux services can only be created
# from profile creation if the current user is the root user.
#
# Valid values: a valid Linux user name

```

```

#
#-OPT PROF_serviceUserName=

#####
# Omit Action
#
# Use this option to omit the config action specified
#
# Valid values: A valid name for an optional config action: deployAdminConsole
#
#-OPT PROF_omitAction="deployAdminConsole"

#####
#
# Stand-Alone Profile
#
# if profileType="standAlone", you may use the following profile creation
# options:
#
#####
#
# Profile name
#
# Specify the name of the profile for the Application Server. The profile
# name must be unique for this WebSphere Application Server installation.
#
# Valid Values:
#   a character string - do not use the following characters:
#   /, \, *, ., :, ;, =, +, ?, |, <, >, &, %, ', ", ]], #, $, ^, {, }
# Note: a period (.) is not valid if it is the first character.
#
#-OPT PROF_profileName=

#####
#
# Profile path
#
# Specify a valid directory to contain the files that define the run-time environment,
# such as commands, configuration files, and log files.
#
# Valid Values: An empty directory, the user must have proper permissions to the
# directory,
# there must be adequate disk space, total path cannot be greater than 80 (windows)
#
#-OPT PROF_profilePath=

#####
#
# Default Profile
#
# Uncomment the following line to make this profile the default target of commands
# that do not use their profile parameter.
# Note that the first profile created for an installation is always marked as
# the default profile.

```

```

#
# Valid Value:
#   true - make this the default profile.
#
#-OPT PROF_isDefault="true"

#####
#
# Host name
#
# Specify the host name for the Application Server. The host name is the domain
# name system (DNS) name (short or long) or the IP address of this computer.
# Valid Values: a valid hostname or IP address
#
#-OPT PROF_hostName=

#####
#
# Application Server Node name
#
# Specify the node name for the Application Server. Node name under one cell
# has to be unique.
#
# Valid Values:
#   a character string - do not use the following characters:
#   /, \, *, , , : , ; , = , + , ? , | , < , > , & , % , ' , " , ]]> , # , $ , ^ , { , }
# Note: a period (.) is not valid if it is the first character.
#
#-OPT PROF_nodeName=

#####
#
# Cell name
#
# Specify the cell name for the profile to be created.
#
# Valid Values:
#   a character string - do not use the following characters:
#   /, \, *, , , : , ; , = , + , ? , | , < , > , & , % , ' , " , ]]> , # , $ , ^ , { , }
# Note: a period (.) is not valid if it is the first character.
#
# -OPT PROF_cellName=

#####
#
# Starting Port
#
# Specify the starting port number for generating all ports for the profile.
# Do not use this parameter with the PROF_portsFile or PROF_defaultPorts parameters.
#
# Valid values: a positive integer port value, within the valid port range
#
#-OPT PROF_startingPort=

```

```
#####
#
# Port File
#
# Specify the path to a property file containing the desired port values for
# the new profile. Do not use this parameter with the PROF_startingPort or
# PROF_defaultPorts parameters.
# Valid values: A fully qualified path to a valid ports property file
#
#-OPT PROF_portsFile=

#####
#
# Default Ports
#
# Uncomment the following line to assign the default port values.
# Otherwise unique port values will be assigned.
# Do not use this parameter with the PROF_portsFile or PROF_startingPort parameters.
#
# Valid value:
#   true - use WebSphere Application Server default ports.
#
#-OPT PROF_defaultPorts="true"

#####
#
# Validate Ports
#
# Uncomment the following line to validate the port values to ensure they are
# not reserved or in use. Otherwise, no port validation checking will occur.
#
# Valid value:
#   true - enables port validation.
#
#-OPT PROF_validatePorts="true"

#####
#
# WinService Check
#
# Specify whether you want to run this server as a windows service
#
# Valid values:
#   true - run as Windows service.
#   false - do not run as Windows service.
#
#-OPT PROF_winserviceCheck="true"

#####
#
# WinService Account Type
#
# Specify the type of the owner account of the Windows service to create.
# Uncomment the following ONLY if PROF_winserviceCheck="true"
```

```

#
# Valid values: specifieduser or localsystem
#
#-OPT PROF_winserviceAccountType=

#####
#
# WinService User Name
#
# Specify the user name for the windows service. Uncomment the
# following ONLY if PROF_winserviceCheck="true"
#
# Valid values: a valid user name for the current system
#
#-OPT PROF_winserviceUserName=

#####
#
# WinService Password
#
# Specify the password for the user specified by the winserviceUserName parameter
# Uncomment the following ONLY if PROF_winserviceCheck="true"
#
# Valid Values:
#   a character string - do not use the following characters:
#   /, \, *, , , : , ; , = , + , ? , | , < , > , & , % , ' , " , ] ] > , # , $ , ^ , { , }
# Note: a period (.) is not valid if it is the first character.
#
#-OPT PROF_winservicePassword=

#####
#
# WinService Startup Type
#
# Specify the start up method for the windows service
# Uncomment the following ONLY if PROF_winserviceCheck="true"
#
# Valid values:
#   manual - windows service must be started manually.
#   automatic - windows service will start automatically after reboot.
#   disabled - service is disabled.
#
# -OPT PROF_winserviceStartupType="automatic"

#####
#
# LinuxService Check
#
# Specify whether you want to run this server as a Linux service.
#
# Note that the root permission is required to enable Linux service.
#
# Valid values:
#   true - run as a Linux service.

```



```

# false - do not run as a Linux service.
#
#-OPT PROF_enableService="true"

#####
#
# LinuxService User Name
#
# Specify the name of the user you wish this service to be run as. Uncomment the
# following ONLY if PROF_enableService="true". Linux services can only be created
# from profile creation if the current user is the root user.
#
# Valid values: a valid Linux user name
#
#-OPT PROF_serviceUserName=

#####
#
# WebServer Check
#
# Specify whether you wish to set the web server definition.
#
# Valid values:
# true - enable the creation of a webserver definition.
# false - do not enable the creation of a webserver definition.
#
#-OPT PROF_webServerCheck="true"

#####
#
# WebServer Type
#
# Specify the type of web server you are creating.
# Uncomment the following if and ONLY if PROF_webServerCheck="true".
#
# Valid values (case sensitive): IHS | IIS | SUNJAVASYSTEM | DOMINO | APACHE |
# HTTPSERVER_ZOS
#
#-OPT PROF_webServerType=

#####
#
# WebServer OS
#
# Specify the operating system of the system where the web server is
# installed. Uncomment the following if and ONLY if PROF_webServerCheck="true".
#
# Valid values: linux | windows | aix | hpux | solaris | os390 | os400
#
#-OPT PROF_webServerOS=

#####
#

```

```

# WebServer Name
#
# Specify the name of the web server. Uncomment the following if and ONLY if
# PROF_webServerCheck="true".
#
# Valid Values:
#   a character string - do not use the following characters:
#   /, \, *, ., ., ., :, ;, =, +, ?, |, <, >, &, %, ', ", ]]>, #, $, ^, {, }
# Note: a period (.) is not valid if it is the first character.
#
#-OPT PROF_webServerName=

#####
#
# WebServer Hostname
#
# Specify the hostname of the system with the web server. Uncomment the following
# if and ONLY if PROF_webServerCheck="true".
#
# Valid values: a valid hostname or IP address
#
#-OPT PROF_webServerHostname=

#####
#
# WebServer Port
#
# Specify the port from which the web server can be accessed. Uncomment the
# following if and ONLY if PROF_webServerCheck="true".
#
# Valid values: a valid port number
#
#-OPT PROF_webServerPort=

#####
#
# WebServer Install Path
#
# Specify the installation path of the web server (local or remote).
# Uncomment the following if and ONLY if PROF_webServerCheck="true".
#
# Valid values: a valid directory path
#
#-OPT PROF_webServerInstallPath=

#####
#
# WebServer Plugin Path
#
# Specify the path to the plugins that will be used by this web server.
# Uncomment the following if and ONLY if PROF_webServerCheck="true".
#
# Valid values: a valid directory path
#

```

```

#-OPT PROF_webServerPluginPath=

#####
#
# Omit Action
#
# Use this option to omit the config action specified
#
# Valid values: one of the following optional config actions
#     deployAdminConsole
#     defaultAppDeployAndConfig
#     samplesInstallAndConfig
#
#-OPT PROF_omitAction=

#####
#
# Developer Server
#
# Uncomment the following line to configure the default application server with
# development environment settings.
# The developer server has a smaller footprint and quicker startup time.
# Otherwise the default configuration values for an application server will be assigned.
#
# Valid value:
#     true - mark this default server for development purposes only.
#
#-OPT PROF_isDeveloperServer="true"

#####
#
# Custom Profile
#
# if profileType="custom", you may use the following profile creation options:
#
#####
#
# Profile name
#
# Specify the name of the profile for the Application Server. The profile
# name must be unique for this WebSphere Application Server installation.
#
# Valid Values:
#     a character string - do not use the following characters:
#     /, \, *, , , : , ; , = , + , ? , | , < , > , & , % , ' , " , ] ] > , # , $ , ^ , { , }
# Note: a period (.) is not valid if it is the first character.
#
#-OPT PROF_profileName=

#####
#
# Profile path
#
# Specify a valid directory to contain the files that define the run-time environment,

```

```

# such as commands,configuration files, and log files.
#
# Valid Values: An empty directory, the user must have proper permissions to the
directory,
# there must be adequate disk space, total path cannot be greater than 80 (windows)
#
#-OPT PROF_profilePath=

#####
#
# Default Profile
#
# Uncomment the following line to make this profile the default target of commands
# that do not use their profile parameter.
# Note that the first profile created for an installation is always marked as
# the default profile.
#
# Valid Value:
#     true - make this the default profile.
#
#-OPT PROF_isDefault="true"

#####
#
# Deployment Manager Hostname
#
# Specify the hostname or address of the machine where the deployment manager
# is running.
#
# Valid values: a valid deployment manager hostname
#
#-OPT PROF_dmgrHost="localhost"

#####
#
# Deployment Manager Port
#
# Specify the SOAP port of the deployment manager.
#
# Valid values: a valid port number
#
#-OPT PROF_dmgrPort="8879"

#####
#
# Deployment Manager Admin User Name
#
# Specify an administrative username that can be authenticated, if
# administrative security on the deployment manager is enabled.
#
# Valid values: a valid deployment manager admin username
#
#-OPT PROF_dmgrAdminUserName=

```

```
#####
#
# Deployment Manager Admin Password
#
# Specify a password that can be authenticated, if administrative security
# on the deployment manager is enabled.
#
# Valid values: a valid deployment manager password
#
#-OPT PROF_dmgrAdminPassword=

#####
#
# Host name
#
# Specify the host name for the Application Server. The host name is the domain
# name system (DNS) name (short or long) or the IP address of this computer.
# Valid Values: a valid hostname or IP address
#
#-OPT PROF_hostName=

#####
#
# Application Server Node name
#
# Specify the node name for the Application Server. Node name under one cell
# has to be unique.
#
# Valid Values:
#   a character string - do not use the following characters:
#   /, \, *, , , : , ; , = , + , ? , | , < , > , & , % , ' , " , ]]> , # , $ , ^ , { , }
# Note: a period (.) is not valid if it is the first character.
#
#-OPT PROF_nodeName=

#####
#
# Cell name
#
# Specify the cell name for the profile to be created.
#
# Valid Values:
#   a character string - do not use the following characters:
#   /, \, *, , , : , ; , = , + , ? , | , < , > , & , % , ' , " , ]]> , # , $ , ^ , { , }
# Note: a period (.) is not valid if it is the first character.
#
#-OPT PROF_cellName=

#####
#
# Port File
#
# Specify the path to a property file containing the desired port values for
```

```

# the new profile. Do not use this parameter with the PROF_startingPort or
# PROF_defaultPorts parameters.
# Valid values: A fully qualified path to a valid ports property file
#
#-OPT PROF_portsFile=

#####
#
# Federate Later
#
# Specify whether federation of the node will be done at a later time.
#
# Valid Values:
#   true - federate this node later.
#   false - federate this node now.
#-OPT PROF_federateLater="false"

#####
#
# Tracing Control
#
# The trace output format can be controlled via the option
#-OPT traceFormat=text
#
# The choices for the format are 'text' and 'XML'. By default, both formats will be
# produced, in two different trace files.
#
# If only one format is required, use the traceFormat option to specify which one, as
# follows:
#
# Valid Values:
#   text -Lines in the trace file will be in a plain text format for easy readability.
#   XML -Lines in the trace file will be in the standard Java logging XML format which
# can be viewed using any text or XML editor or using the Chainsaw tool from Apache
# (http://logging.apache.org/log4j/docs/chainsaw.html).
#
#
# The amount of trace info captured can be controlled using the option:
#-OPT traceLevel=WARNING
#
# Valid Values:
#
# Level Numerical LevelDescription
# OFF 0 No trace file is produced
# SEVERE 1 Only severe errors are output to trace file
# WARNING 2 Messages regarding non-fatal exceptions and warnings are added to trace
# file
# INFO 3 Informational messages are added to the trace file (this is the
# default trace level)
# CONFIG 4 Configuration related messages are added to the trace file
# FINE 5 Tracing method calls for public methods
# FINER 6 Tracing method calls for non public methods except getters and setters

```

```
# FINEST 7 Trace all method calls, trace entry/exit will include parameters and  
return value
```

Archived

Update Installer: response.txt

Example: A-2 Installer response

```
#####
#
# License Acceptance
#
# Valid Values:
#     true - Accepts the license. Will install the product.
#     false - Declines the license. Install will not occur.
#
# If no install occurs, this will be logged to a temporary log file in the
# user's temporary directory.
#
# By changing the silentInstallLicenseAcceptance property in this response file
# to "true", you agree that you have reviewed and agree to the terms of the
# IBM International Program License Agreement accompanying this program, which is
# located at <CD_ROOT>\updi.primary.pak\repository\updi.legal\lafiles. If you do
# not agree to these terms, do not change the value or otherwise download, install,
# copy, access, or use the program and promptly return the program and proof of
# entitlement to the party from whom you acquired it to obtain a refund of the
# amount you paid.
#
#
-OPT silentInstallLicenseAcceptance="true"

#####
#
# NonRoot Install Settings
#
# The option indicates whether you accept the limitations associated with installing
# as a non-root user, which are detailed in the documentation. (specifics to be add)
#
# Valid Values:
#     true - Accepts the limitations. Will install the product.
#     false - Do not accept the limitations. Install will not occur.
#
# Uncomment the following only if you're installing as a non-root user. Specify one of
# the
# valid options listed above before proceeding to install:
# -OPT allowNonRootSilentInstall="true"

#####
# Operating System Prerequisite Checking
#
# If you want to disable operating system prerequisite checking, uncomment
# the following line. This will notify the installer to continue with
# the installation and log the warnings even though the prerequisite checking
# has failed.
#
-OPT disableOSPrereqChecking="true"
```



```
#####
# Existing Installation Checking
#
# If you want to disable the checking for existing Update Installer, uncomment
# the following line. This will notify the installer to continue with the
# installation and log the warnings even though the prerequisite checking has
# failed.
#
-OPT disableEarlyPrereqChecking="true"

#####
#
# Install Location
#
# The install location of the product. Specify a valid directory into which the
# product should be installed. If the directory contains spaces, enclose it in
# double-quotes as shown in the Windows example below. Note that spaces in the
# install location is only supported on Windows operating systems. Maximum path
# length is 60 characters for Windows 2000 and Windows XP.
#
# Below is the list of default install locations for each supported operating
# system when you're installing as a root user. By default, in this response file,
# the Windows install location is used. If you want to use the default install
# location for another operating system, uncomment the appropriate default install
# location entry (by removing '#') and then comment out (by adding '#') the
# Windows operating system entry below.
#
# AIX Default Install Location:
#
# -OPT installLocation="/usr/IBM/WebSphere/UpdateInstaller"
#
# HP-UX, Solaris or Linux Default Install Location:
#
# -OPT installLocation="/opt/IBM/WebSphere/UpdateInstaller"
#
# i5OS Default Install Location:
#
# -OPT installLocation="/QIBM/IBM/WebSphere/UpdateInstaller/V61/<productOffering>"
#
# Windows Default Install Location:
#
# -OPT installLocation="C:\Program Files\IBM\WebSphere\UpdateInstaller"
-OPT installLocation="/usr/IBM/WebSphere/UpdateInstaller"

# If you are installing as non-root user on Unix or non-administrator on Windows,
# the following default install locations are suggested. Be sure you have write
# permission for the install location chosen.
#
# AIX Default Install Location:
#
# -OPT installLocation="<user's home>/IBM/WebSphere/UpdateInstaller"
#
```

```
# HP-UX, Solaris or Linux Default Install Location:
#
# -OPT installLocation="<user's home>/IBM/WebSphere/UpdateInstaller"
#
# i5OS Default Install Location:
#
# <not applicable>
#
# Windows Default Install Location:
#
# -OPT installLocation="C:\IBM\WebSphere\UpdateInstaller"
```

WebSphere V6.1 Fix Pack 2: fp2.response.txt

Example: A-3 Fix Pack response

```
#####
#
# This is the silent install response file for installing maintenance packages
# using the update installer.
#
# A common use of an options file is to run the wizard in silent mode. This lets
# the options file author specify wizard settings without having to run the
# wizard in graphical or console mode. To use this options file for silent mode
# execution, *uncomment* and modify the parameters defined within.
#
# Use the following command line when running the wizard from the update
# installer directory:
#
#     update -options responsefiles/install.txt -silent
#
# Please enclose all values within a single pair of double quotes.
#
#####

#####
# Used to input the maintenance package full filename specification to be installed.
# Edit as appropriate.
#
# ie. -W maintenance.package="C:\Program
Files\IBM\WebSphere\AppServer\updateinstaller\maintenance\PQ20029.pak"
#
# Note: If no package is specified, a default of the last downloaded maintenance
# package will be used (based on timestamp).
#
-W maintenance.package="/exports/was_intall/was61fp2/6.1.0-WS-WebSphere Application
Server -AixPPC64-FP0000002.pak"

#####
# Used to input the product install location that will be updated.
# ie. -W product.location="C:\Program Files\IBM\WebSphere\AppServer"
#
# Note: The product install location should always been specified, and it should
# always be the full path.
#
-W product.location="/usr/IBM/WebSphere/AppServer"

#####
#
# Do not edit these values.
#
-W update.type="install"
```

Trade 6.1 Installation script: trade.jacl

Example: A-4 trade.jacl installation script

```
#-----
# trade.jacl - Configure Trade
#-----
#
# Author: Christopher Blythe
#
# This script is designed to configure the JDBC and JMS resource required by
# the Trade application. Both single server and clustered environments are
# supported. A "silent install" option is also supported after manually setting
# the default config options in this jacl file.
#
# To invoke the script type:
# wsadmin -f trade.jacl [all|configure|install|start|uninstall\]
#   where:  all      - configures JDBC and JMS resources and installs the app
#              configure - only configures the JDBC and JMS resource
#              install  - installs the Trade ear
#              start    - starts the Trade application on a single server
#              uninstall - uninstalls the Trade application on a single server
#
# If no parameters are specified, "all" is assumed!
#

$AdminConfig setValidationLevel NONE

puts "trade.jacl"

set dir [pwd]
set wasPerfJacl [file join $dir resource_scripts.jacl]
source $wasPerfJacl

set SilentInstall      "false"
set DefaultTradeAppName "Trade"
set DefaultEarFile     "trade.ear"

#-----
# Basic Cluster Properties
#-----

set DefaultClusterName      "TradeCluster"
set DefaultClusterDescription "Trade Cluster Config"
set DefaultClusterMember    "TradeServer"
set DefaultClusterInstall   "no"

set DefaultMemberWeight     2
set DefaultPreferLocal      "true"

#-----
```

```

# JDBC Driver and DataSource Config Parameters
#-----

# Default options for database classpath
# Note: wsadmin parses the command line based on ";" regardless of platform type
#set DB2JccPath
"c:/sql1lib/java/db2jcc.jar;c:/sql1lib/java/db2jcc_license_cu.jar;c:/sql1lib/java/db2jcc_li
cense_cisuz.jar"
set DB2JccPath
"/home/db2rc11/sql1lib/java/db2jcc.jar;/home/db2rc11/sql1lib/java/db2jcc_license_cu.jar"
#set DB2NativePath
"/usr/lpp/db2/db2810/jcc/lib"
set DB2NativePath
"/opt/IBM/db2/V9.1/lib64"
#set DB2ClibPath
"c:/sql1lib/java/db2java.zip"
set DB2ClibPath
"/home/db2rc11/sql1lib/java/db2java.zip"
set OraclePath
"c:/oracle/product/10.1.0/db_1/jdbc/lib/odbc14.jar"
set DerbyPath
"%\\${WebSphere Application Server
_INSTALL_ROOT}\\derby/lib/derby.jar"
set DB2iSeriesNativePath
"/QIBM/ProdData/Java400/ext/db2_classes.jar"
set DB2iSeriesToolboxPath
"/QIBM/ProdData/HTTP/Public/jt400/lib/jt400.jar"

# Default options for database user name
set DefaultUserDB2
"db2rc11"
set DefaultUserOracle
"trade"

# Default options for database password
set DefaultPasswdDB2
"login1"
set DefaultPasswdOracle
"trade"

# Datasource properties
set DefaultDataSourceName
"TradeDataSource"
set DefaultAuthAliasName
"TradeDataSourceAuthData"
set DefaultStmtCacheSize
60
set ClusterMEDatasourceName
"MEDDataSource"

# Global Security properties for JMS
set DefaultOSAuthAlias
"TradeOSUserIDAuthData"

# EJB Deploy database types
# Note: The default backend deploy types are now DB2UDB_V82, DB2UDBOS390_V8,
# DB2UDBISERIES_V54, and ORACLE_V10G. If you are using a database versions
# other than these (ie. DB2UDB_V81, ORACLE_V9I, DB2UDBOS390_V7, etc.), you
# will need to modify these parameters by hand.
set DB2Deploy
"DB2UDB_V82"
set DB2zOSDeploy
"DB2UDBOS390_V8"
set DB2iSeriesDeploy"DB2UDBISERIES_V54"
set OracleDeploy
"ORACLE_V10G"
set DerbyDeploy
"DERBY_V10"

#-----
# Default Properties for Silent Install
#
# NOTE: Silent Install only functions for SINGLE Server

```

```

#-----

# Generic database options
# Options: db2 db2cli oracle db2zos
# Note: Installations on z/OS are currently limited to single server.
# Therefore, the script will terminate if db2zos is chosen for the
# database type and a cluster install is performed.
set DefaultDbType "db2"

set DefaultXA "true"
set DefaultPathName ${DB2JccPath}
set DefaultNativePathName ${DB2NativePath}
set DefaultUser ${DefaultUserDB2}
set DefaultPasswd ${DefaultPasswdDB2}

# DB2 database options
# Note: The default DB2 Universal Driver Type has been changed to Type 4
# since DB2 V82 now supports XA with the Type 4 driver. If you would like
# to use previous versions of DB2, this MUST be changed to Type 2.
set DefaultDatabaseName "stradedb"
set DefaultDB2JccDriverType 4
set DefaultDB2Hostname "brazos.itsc.austin.ibm.com"
set DefaultDB2Port "50000"

# Oracle database options
set DefaultOracleSID "tradedb"
set DefaultOracleHostname "localhost"
set DefaultOraclePort 1521

# Deployment options
set DefaultRunEJBDeploy "true"
set DefaultRunWSDeploy "true"
set DefaultEJBDeployDBType ${DB2Deploy}

# Messaging security options
# Note: If global security is enabled or will be enabled at some point and
# time, the OSAuthData alias should be updated with a valid system (OS) id
# and password. For cluster configurations, LDAP, Windows Active Direcotry
# or some other form of centralized authenticatoion mechanism must be used
# to validate the userid.
set SecurityEnabled "false"
set DefaultOSUser "LocalOSUserID"
set DefaultOSPasswd "password"

# Silent install properties for Managed Node
# - Modify these properties to specify the target node and server
set TargetNodeName "PLNode"
set TargetServerName "Server1"

# Silent install properties for Cluster
# - Append items to this list for the desired nodes and cluster members
set ClusterNodeNames [list "nodeName1" "nodeName2"]
set ClusterMemberNames [list "serverName1" "serverName2"]

```

```

#-----
# JMS (Messaging) Config Parameters
#-----

#set reliability      "ASSURED_PERSISTENT"
set reliability       "EXPRESS_NONPERSISTENT"

#set deliveryMode     "Persistent"
set deliveryMode      "NonPersistent"

# Queue/Topic Names
set brokerSIBDest     "TradeBrokerJSD"
set topicSpace        "Trade.Topic.Space"
set brokerJMSQCF      "TradeBrokerQCF"
set streamerJMSTCF    "TradeStreamerTCF"
set brokerQueue       "TradeBrokerQueue"
set streamerTopic     "TradeStreamerTopic"
set brokerMDB         "TradeBrokerMDB"
set streamerMDB       "TradeStreamerMDB"

#-----
# Other parameters and options
#-----

set CmdOptions        [list "all" "configure" "install" "start" "uninstall"]
set DefaultOptions    [list "yes" "no"]
set BooleanOptions    [list "true" "false"]
set DatabaseOptions   [list "db2" "db2cli" "oracle" "db2zos" "derby" "iSeriesNative"
                          "iSeriesToolbox"]
set DriverTypeOptions [list 2 4]

#-----
# Basic App Administration Procedures
#-----

proc printUsageAndExit {} {
    puts ""
    puts "Usage: wsadmin -f trade.jacl \"[all|configure|install|start|uninstall]\""
    puts ""
    puts "  where: all          - configures JDBC and JMS resources and installs the app"
    puts "         configure    - only configures the JDBC and JMS resource"
    puts "         install      - installs the Trade ear"
    puts "         start        - starts the Trade application on a single server"
    puts "         uninstall    - uninstalls the Trade application on a single server"
    puts ""
    puts "  If no parameters are specified, \"all\" is assumed!"
    puts ""
}

```

```

    exit
}

#-----
# Parse Command Line
#-----

if {[length $argv] == 0} {
    set operation "all"
} elseif {[lsearch -glob $CmdOptions [lindex $argv 0]] > -1} {
    set operation [lindex $argv 0]
} else {
    printUsageAndExit
}

puts ""
puts "-----"
puts " Trade Install/Configuration Script"
puts ""
puts " Operation:  ${operation}"
puts " Silent:    ${SilentInstall}"
puts "-----"

#-----
# Trade configuration procedures
#-----

if ${SilentInstall} == "false" && (${operation} == "configure" || ${operation} ==
"all")) {

    set SecurityEnabled [getValidInput "Global security is \\\(or will be\\) enabled
\\(true|false\\) \\\[${SecurityEnabled}\\]:" $SecurityEnabled $BooleanOptions]

    set DefaultClusterInstall [getValidInput "Is this a cluster installation \\\(yes\\|no\\)
\\[${DefaultClusterInstall}\\]:" $DefaultClusterInstall $DefaultOptions]

    if ${DefaultClusterInstall} == "yes" {

        set forZOS [getValidInput "Is this installation intended for z/OS \\\(yes\\|now\\)
\\[no\\]:" "no" $DefaultOptions]

        if ${forZOS} == "yes" {
            puts " "
            puts "-----"
            puts " Note: Trade installations on z/OS platforms are"
            puts " currently limited to single server environments."
            puts " Exiting the install script!"
            puts "-----"

            exit
        }
    }
}

```



```

set cell [getCellId]
set cellName [getName $cell]
set scope $cell

puts " "
puts "-----"
puts " Collecting Cluster and Cluster Member Info"
puts " "
puts " Note: Before proceeding, all nodes intended for"
puts " use in this cluster must be federated with the"
puts " deployment manager using the addNode command!"
puts " To ensure that this process goes smoothly, it"
puts " is also important to verify that each node can"
puts " ping the other cluster nodes based on the host"
puts " names configured within the WebSphere profile"
puts " creation tool."
puts "-----"

set continue [getValidInput "Have all nodes been federated and network
connectivity \verified? \[yes\|no\] \[yes\]:" "yes" $DefaultOptions]
if {$continue == "no"} {
    exit
}

# Obtain cluster name and create cluster

set DefaultClusterName [getInput "Please enter the cluster name
\[{$DefaultClusterName}\]:" $DefaultClusterName]

# Obtain cluster member details from the user, generating a list
# of member/node pairs to be used to create the cluster members

set addNodes "yes"

set ClusterNodeNames [list ]
set ClusterMemberNames [list ]

# Continue adding node/member pairs until the user enters "no"
while {$addNodes == "yes"} {

    set node [getNodeId]
    lappend ClusterNodeNames [getName $node]
    set count [llength $ClusterNodeNames]

    set member [getInput "Please enter the cluster member name
\[{$DefaultClusterMember}${count}\]:" "${DefaultClusterMember}${count}"]
    lappend ClusterMemberNames $member

    # List cluster node/member pairs
    puts ""
    puts "Current Cluster Nodes and Members:"
    set idx 0
    foreach item $ClusterNodeNames {

```

```

        set tmp [lindex $ClusterMemberNames $idx]
        set idx [expr $idx + 1]
        puts " ${item} - ${tmp}"
    }

    set addNodes [getValidInput "Add more cluster members \ (yes\|no\ ) \[yes\]:"
"yes" $DefaultOptions]
    }

    puts ""
    puts "Cluster information obtained..."
} else {
    # Obtain node name and id for scope

    puts "-----"
    puts " Collecting Single Server or Managed Server Info"
    puts "-----"

    set node [getNodeId]
    set TargetNodeName [getName $node]
    set scope $node

    set server [getServerId]
    set TargetServerName [getName $server]
}

puts "-----"
puts " Collecting Database/Datasource Information"
puts "-----"

set DefaultDbType [getValidInput "Select the backend database type
\ (db2\|oracle\|db2zos\ ) \[${DefaultDbType}\]:" $DefaultDbType $DatabaseOptions]

if ${DefaultDbType} == "db2" {
    set DefaultPathName ${DB2JccPath}
    set DefaultUser ${DefaultUserDB2}
    set DefaultPasswd ${DefaultPasswdDB2}
    set DefaultNativePathName ""
} elseif ${DefaultDbType} == "oracle" {
    set DefaultPathName ${OraclePath}
    set DefaultUser ${DefaultUserOracle}
    set DefaultPasswd ${DefaultPasswdOracle}
    set DefaultNativePathName ""
} elseif ${DefaultDbType} == "db2cli" {
    set DefaultPathName ${DB2CliPath}
    set DefaultUser ${DefaultUserDB2}
    set DefaultPasswd ${DefaultPasswdDB2}
    set DefaultNativePathName ""
} elseif ${DefaultDbType} == "derby" {
    set DefaultDatabaseName
"\${APP_INSTALL_ROOT}\|\${CELL}\|Trade.ear/tradeDB"
    set DefaultPathName ${DerbyPath}
    set DefaultUser "app"
    set DefaultPasswd "dummy"
}

```

```

        set DefaultNativePathName      ""
    } elseif ${DefaultDbType} == "db2zos" {
        set DefaultPathName             ${DB2JccPath}
        set DefaultUser                 ${DefaultUserDB2}
        set DefaultPasswd               ${DefaultPasswdDB2}
        set DefaultNativePathName      ${DB2NativePath}
    } elseif ${DefaultDbType} == "iSeriesNative" {
        set DefaultPathName             ${DB2iSeriesNativePath}
        set DefaultUser                 ${DefaultUserDB2}
        set DefaultPasswd               ${DefaultPasswdDB2}
    } elseif ${DefaultDbType} == "iSeriesToolbox" {
        set DefaultPathName             ${DB2iSeriesToolboxPath}
        set DefaultUser                 ${DefaultUserDB2}
        set DefaultPasswd               ${DefaultPasswdDB2}
    }
}

puts ""
puts "NOTE: wsadmin requires \";\";\" for delimiting the database"
puts "driver path regardless of platform!"
set DefaultPathName [getInput "Please enter the database driver path
\${DefaultPathName}\":\" $DefaultPathName]

if {[string first "db2" ${DefaultDbType}] >= 0 || ${DefaultDbType} == "derby" } {
    if ${DefaultDbType} == "db2zos" {
        set DefaultDB2JccDriverType 2
        set DefaultXA                "false"

        set DefaultNativePathName    [getInput "Please enter the driver native library
path \${DefaultNativePathName}\":\" $DefaultNativePathName]
    }

    set DefaultDatabaseName    [getInput "Please enter the database name (location)
\${DefaultDatabaseName}\":\" $DefaultDatabaseName]

    if {[string first "db2" ${DefaultDbType}] >= 0 && ${DefaultDbType} != "db2cli" } {
        #set DefaultDB2JccDriverType    [getValidInput "Please enter the JDBC driver
type \2\|4\ \${DefaultDB2JccDriverType}\":\" $DefaultDB2JccDriverType
$DriverTypeOptions]
        if ${DefaultDB2JccDriverType} == 4 {
            set DefaultDB2Hostname    [getInput "Please enter the DB2 database
hostname \${DefaultDB2Hostname}\":\" $DefaultDB2Hostname]
            set DefaultDB2Port        [getInput "Please enter the DB2 database port
number \${DefaultDB2Port}\":\" $DefaultDB2Port]
        }
    }
} elseif ${DefaultDbType} == "oracle" {
    set DefaultOracleHostname [getInput "Please enter the Oracle database hostname
\${DefaultOracleHostname}\":\" $DefaultOracleHostname]
    set DefaultOracleSID      [getInput "Please enter the Oracle database SID
\${DefaultOracleSID}\":\" $DefaultOracleSID]
    set DefaultOraclePort     [getInput "Please enter the Oracle listener port number
\${DefaultOraclePort}\":\" $DefaultOraclePort]
} elseif {[string first "iSeries" ${DefaultDbType}] >= 0 } {

```

```

        set DefaultDatabaseName [getInput "Please enter the database name
\[{DefaultDatabaseName}\]:" $DefaultDatabaseName]
        if {${DefaultDbType} == "iSeriesToolbox"} {
            set DefaultDB2Hostname[getInput "Please enter the Toolbox database hostname
\[{DefaultDB2Hostname}\]:" $DefaultDB2Hostname]
        }
    }

    if {${DefaultDbType} != "derby"} {
        set DefaultUser [getInput "Please enter the database username
\[{DefaultUser}\]:" $DefaultUser]
        set DefaultPasswd [getInput "Please enter the database password
\[{DefaultPasswd}\]:" $DefaultPasswd]
    }

    if {${SecurityEnabled} == "true"} {
        puts "-----"
        puts " Collecting Security Information for JMS"
        puts " "
        puts " Note: The supplied authentication data must"
        puts " correspond to a valid system (OS) userid and"
        puts " password. For cluster environments, LDAP or"
        puts " another supported centralized authentication"
        puts " mechanism must be configured to validate the"
        puts " userid."
        puts "-----"

        set DefaultOSUser [getInput "Please enter the system username
\[{DefaultOSUser}\]:" $DefaultOSUser]
        set DefaultOSPasswd [getInput "Please enter the system password
\[{DefaultOSPasswd}\]:" $DefaultOSPasswd]
    }
}

if {$operation == "all" || $operation == "configure"} {

    # If this is a cluster install, create the cluster and cluster members
    if {${DefaultClusterInstall} == "yes"} {
        # Exit install if this cluster install is intended for z/OS
        if {${DefaultDbType} == "db2zos"} {
            puts " "
            puts "-----"
            puts " Note: Trade installations on z/OS platforms are"
            puts " currently limited to single server environments."
            puts " Exiting the install script!"
            puts "-----"

            exit
        }

        # Obtain cell name and id for scope

        set cell [getCellId]
    }
}

```

```

set cellName [getName $cell]
set scope $cell

puts ""
puts "-----"
puts " Configuring Cluster and Cluster Members"
puts " Scope: $scope"
puts "-----"

# Create the cluster
set cluster [createCluster $DefaultClusterName $DefaultPreferLocal
$DefaultClusterDescription $scope]

set idx 0
foreach item $ClusterNodeNames {
    set node [$AdminConfig getid "/Node:$item/"]
    set member [lindex $ClusterMemberNames $idx]
    set idx [expr $idx + 1]

    # Create the cluster member
    set clusterMember [createClusterMember $member $node $DefaultMemberWeight
$cluster]

    # Enable SIB Service on the cluster member
    enableSIBService $clusterMember
}

puts ""
puts "-----"
puts " Cluster Configuration Completed!!!"
puts "-----"
} else {
    set scope [$AdminConfig getid "/Node:${TargetNodeName}/"]
}

# Create the JDBC/Datasource config objects

puts ""
puts "-----"
puts " Configuring JDBC/Datasource Resources"
puts " Scope: $scope"
puts "-----"

createJAASAuthData $DefaultAuthAliasName $DefaultUser $DefaultPasswd

set provider [createJDBCProvider $DefaultDbType $DefaultPathName
$DefaultNativePathName $DefaultXA $scope]

if {[string first "db2" ${DefaultDbType}] >= 0 || ${DefaultDbType} == "derby"} {
    set jccParms [subst {}]
    if ${DefaultDbType} == "db2" || ${DefaultDbType} == "db2zos"} {
        set jccParms [subst ${DefaultDB2JccDriverType $DefaultDB2Hostname
$DefaultDB2Port}]
    }
}

```

```

        createDB2orCloudscapeDatasource $DefaultDatasourceName
        "jdbc/${DefaultDatasourceName}" $DefaultStmtCacheSize $provider $DefaultDbType
        $DefaultAuthAliasName "Trade6 Datasource" $scope $DefaultDatabaseName $jccParms
    } elseif ${DefaultDbType} == "oracle" {
        createOracleDatasource $DefaultDatasourceName "jdbc/${DefaultDatasourceName}"
        $DefaultStmtCacheSize $provider $DefaultAuthAliasName "Trade6 Datasource" $scope
        $DefaultOracleSID $DefaultOracleHostname $DefaultOraclePort
    } elseif {[string first "iSeries" ${DefaultDbType}] >= 0} {
        set optParms [subst {}]
        if ${DefaultDbType} == "iSeriesToolbox" {
            set optParms [subst ${DefaultDB2Hostname}]
        }
        createISeriesDataSource $DefaultDatasourceName "jdbc/${DefaultDatasourceName}"
        $DefaultStmtCacheSize $provider $DefaultDbType $DefaultAuthAliasName "Trade6 Datasource"
        $scope $DefaultDatabaseName $optParms
    }

    if ${DefaultClusterInstall} == "yes" {
        # Create a non-XA provider and datasource for the MES
        set meProvider [createJDBCProvider $DefaultDbType $DefaultPathName
        $DefaultNativePathName "false" $scope]

        if {[string first "db2" ${DefaultDbType}] >= 0 || ${DefaultDbType} == "derby"} {
            createDB2orCloudscapeDatasource $ClusterMEDatasourceName
            "jdbc/${ClusterMEDatasourceName}" $DefaultStmtCacheSize $meProvider $DefaultDbType
            $DefaultAuthAliasName "Share ME Datasource" $scope $DefaultDatabaseName $jccParms
        } elseif ${DefaultDbType} == "oracle" {
            createOracleDatasource $ClusterMEDatasourceName
            "jdbc/${ClusterMEDatasourceName}" $DefaultStmtCacheSize $meProvider
            $DefaultAuthAliasName "Shared ME Datasource" $scope $DefaultOracleSID
            $DefaultOracleHostname $DefaultOraclePort
        } elseif {[string first "iSeries" ${DefaultDbType}] >= 0} {
            createISeriesDataSource $ClusterMEDatasourceName
            "jdbc/${ClusterMEDatasourceName}" $DefaultStmtCacheSize $meProvider $DefaultDbType
            $DefaultAuthAliasName "Shared ME Datasource" $scope $DefaultDatabaseName $optParms
        }
    }

    puts ""
    puts "-----"
    puts " JDBC Resource Configuration Completed!!!"
    puts "-----"

    # Create the JMS config objects

    puts ""
    puts "-----"
    puts " Configuring JMS Resources"
    puts " Scope: $scope"
    puts "-----"

    createJAASAuthData $DefaultOSAuthAlias $DefaultOSUser $DefaultOSPasswd

```

```

if ${DefaultClusterInstall} == "yes" {

    set SIBusName [createSIBus $DefaultClusterName $DefaultOSAuthAlias]

    addSIBusMember $SIBusName "false" "jdbc/${ClusterMEDatasourceName}"
$DefaultClusterName

    #set temp [lindex $ClusterNodeNames 0]
    #copySIBJmsRatoCell [$AdminConfig getid "/Node:${temp}/"] $scope

    # Create additional MEs
    set idx 0
    foreach item $ClusterMemberNames {
        if ${idx} > 0 {
            set temp [subst ${DefaultClusterName}]
            createMessageEngine $SIBusName "false" "jdbc/${ClusterMEDatasourceName}"
$temp
        }
        set idx [expr $idx + 1]
    }

    set parms [list -bus $DefaultClusterName -cluster $DefaultClusterName]
    set meList [$AdminTask listSIBEngines $parms]
    set baseName "Policy for ME"
    set baseSchema "IBMME"

    set idx 0
    foreach engine $meList {
        set name "${baseName}${idx}"
        set schema "${baseSchema}${idx}"
        set serverName [lindex $ClusterMemberNames $idx]
        set idx [expr $idx + 1]

        # Create the HA policy to pin servers
        createOneOfNPOLICY $name 30 $serverName [getName $engine]

        # Modify the MEs to use the same datastore
        if ${DefaultDbType} == "oracle" {
            set alias "${schema}AuthAlias"
            createJAASAuthData $alias $schema $DefaultPasswd

            modifyMEDataStore [getName $engine] $alias $schema
        } else {
            modifyMEDataStore [getName $engine] $DefaultAuthAliasName $schema
        }
    }

    set target [subst ${DefaultClusterName}]

} else {
    set SIBusName [createSIBus [getName $scope] $DefaultOSAuthAlias]

    set target [subst ${TargetNodeName} $TargetServerName]

```

```

        addSIBusMember $SIBusName "true" "dummy" $target
    }

    if {{SecurityEnabled} == "true"} {
        createSIBusSecurityRole $SIBusName $DefaultOSUser
    }

    # Create the Trade Broker Queue and Trade TopicSpace Destinations

    createSIBDestination $SIBusName $brokerSIBDest "Queue" $reliability $target
    createSIBDestination $SIBusName $topicSpace "TopicSpace" $reliability [subst {}]

    createJMSConnectionFactory $SIBusName $brokerJMSQCF "Queue" "jms/$brokerJMSQCF"
    $DefaultOSAuthAlias $scope
    createJMSConnectionFactory $SIBusName $streamerJMSTCF "Topic" "jms/$streamerJMSTCF"
    $DefaultOSAuthAlias $scope

    createJMSQueue $brokerQueue "jms/$brokerQueue" $brokerSIBDest $deliveryMode $scope
    createJMSTopic $streamerTopic "jms/$streamerTopic" $topicSpace $deliveryMode $scope

    createMDBActivationSpec $brokerMDB "eis/$brokerMDB" $SIBusName "jms/$brokerQueue"
    "javax.jms.Queue" $DefaultOSAuthAlias $scope
    createMDBActivationSpec $streamerMDB "eis/$streamerMDB" $SIBusName
    "jms/$streamerTopic" "javax.jms.Topic" $DefaultOSAuthAlias $scope

    puts ""
    puts "-----"
    puts " JMS Resource Configuration Completed!!!"
    puts "-----"

    puts ""
    puts "Saving..."
    $AdminConfig save
}

#-----
# Trade install procedures
#-----

if {$operation == "all" || $operation == "install"} {
    puts " "
    puts "-----"
    puts " Installing Trade"
    puts "-----"

    if {{SilentInstall} == "false" && $operation == "install"} {
        set DefaultClusterInstall [getValidInput "Is this a cluster installation
        \ (yes\|no\)\ \[${DefaultClusterInstall}]:" $DefaultClusterInstall $DefaultOptions]
        if {{DefaultClusterInstall} == "no"} {

```



```

        set targetNode      [getNodeId]
        set TargetNodeName  [getName $targetNode]
        set targetServer    [getServerId]
        set TargetServerName [getName $targetServer]
    } else {
        set DefaultClusterName [getInput "Please enter the cluster name
\${DefaultClusterName}\":" $DefaultClusterName]
    }
    set DefaultDbType [getValidInput "Select the backend database type
\(\db2\|db2zos\|oracle\) \${DefaultDbType}\":" $DefaultDbType $DatabaseOptions]
}

if {\${DefaultClusterInstall} == "yes"} {
    set target      [subst {\${DefaultClusterName}}]
} else {
    set target      [subst {\$TargetNodeName \$TargetServerName}]
}

if {\${DefaultDbType} == "db2"} {
    set DefaultEJBDeployDBType ${DB2Deploy}
} elseif {\${DefaultDbType} == "oracle"} {
    set DefaultEJBDeployDBType ${OracleDeploy}
} elseif {\${DefaultDbType} == "db2cli"} {
    set DefaultEJBDeployDBType ${DB2Deploy}
} elseif {\${DefaultDbType} == "db2zos"} {
    set DefaultEJBDeployDBType ${DB2zOSDeploy}
} elseif {\${DefaultDbType} == "derby"} {
    set DefaultEJBDeployDBType ${DerbyDeploy}
} elseif {[string first "iSeries" \${DefaultDbType}] >= 0} {
    set DefaultEJBDeployDBType ${DB2iSeriesDeploy}
}

installApp \$DefaultTradeAppName \$DefaultEarFile \$DefaultRunEJBDeploy
\$DefaultRunWSDeploy "true" "true" \$DefaultEJBDeployDBType $target

puts ""
puts "-----"
puts " Trade Installation Completed!!!"
puts "-----"

puts ""
puts "Saving..."
$AdminConfig save
}

if {\$operation == "start"} {
    puts " "
    puts "-----"
    puts " Starting Trade"
    puts " NOTE: Only executes for a single node"
    puts "-----"

    if {\${DefaultClusterInstall} == "no"} {

```

```

        startApp $DefaultTradeAppName
    }
}

#-----
# Trade uninstall procedures
#-----

if {$operation == "uninstall"} {
    puts " "
    puts "-----"
    puts " Stopping and Uninstalling Trade"
    puts " NOTE: Only executes for a single node"
    puts "-----"

    if ${DefaultClusterInstall} == "no" {
        stopApp $DefaultTradeAppName
        uninstallApp $DefaultTradeAppName
    }
}

puts ""
puts "Saving config..."
$AdminConfig save

```

CSM adapter definition file: p550q_lpar_adapters

Example: A-5 CSM adapter definition file

```
###CSM_ADAPTERS_STANZA_FILE###--do not remove this line
#---Stanza Summary-----
#   Date: Tue Sep 19 17:12:11 CDT 2006
#   Stanzas Added: 3
#---End Of Summary-----
```

```
de.itsc.austin.ibm.com:
  MAC_address=922430002002
  adapter_type=ent
  cable_type=N/A
  install_gateway=9.3.5.41
  location=U9133.55A.10D1FAG-V2-C2-T1
  machine_type=install
  netaddr=9.3.5.187
  interface_type=en
  subnet_mask=255.255.255.0
  adapter_duplex=auto
  adapter_speed=auto
```

```
uk.itsc.austin.ibm.com:
  MAC_address=922430004002
  adapter_type=ent
  cable_type=N/A
  install_gateway=9.3.5.41
  location=U9133.55A.10D1FAG-V4-C2-T1
  machine_type=install
  netaddr=9.3.5.183
  interface_type=en
  subnet_mask=255.255.255.0
  adapter_duplex=auto
  adapter_speed=auto
```

```
us.itsc.austin.ibm.com:
  MAC_address=922430005002
  adapter_type=ent
  cable_type=N/A
  install_gateway=9.3.5.41
  location=U9133.55A.10D1FAG-V5-C2-T1
  machine_type=install
  netaddr=9.3.5.184
  interface_type=en
  subnet_mask=255.255.255.0
  adapter_duplex=auto
  adapter_speed=auto
```

```
br.itsc.austin.ibm.com:
```

```
MAC_address=922430003002
adapter_type=ent
cable_type=N/A
install_gateway=9.3.5.41
location=U9133.55A.10D1FAG-V3-C2-T1
machine_type=install
netaddr=9.3.5.182
interface_type=en
subnet_mask=255.255.255.0
adapter_duplex=auto
adapter_speed=auto
pt.itsc.austin.ibm.com:
MAC_address=922430006002
adapter_type=ent
cable_type=N/A
install_gateway=9.3.5.41
location=U9133.55A.10D1FAG-V6-C2-T1
machine_type=install
netaddr=9.3.5.188
interface_type=en
subnet_mask=255.255.255.0
adapter_duplex=auto
adapter_speed=auto

pl.itsc.austin.ibm.com:
MAC_address=922430007002
adapter_type=ent
cable_type=N/A
install_gateway=9.3.5.41
location=U9133.55A.10D1FAG-V7-C2-T1
machine_type=install
netaddr=9.3.5.186
interface_type=en
subnet_mask=255.255.255.0
adapter_duplex=auto
adapter_speed=auto
```

Additional material

This book refers to additional material that can be downloaded from the Internet as described here.

Locating the Web material

The Web material associated with this book is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

<ftp://www.redbooks.ibm.com/redbooks/SG247347>

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG247347.

Using the Web material

The additional Web material that accompanies this book includes the following files:

<i>File name</i>	<i>Description</i>
SG247347_AddMat.ZIP	A ZIP file containing sample files used in this book.

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks publications

For information about ordering these publications, see “How to get IBM Redbooks” on page 477. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *Introduction to pSeries Provisioning*, SG24-6389
- ▶ *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392
- ▶ *WebSphere Application Server V6 Planning and Design WebSphere Handbook Series*, SG24-6446
- ▶ *AIX 5L Practical Performance Tools and Tuning Guide*, SG24-6478
- ▶ *WebSphere Application Server Network Deployment V6: High Availability Solutions*, SG24-6688
- ▶ *Implementing High Availability Cluster Multi-Processing (HACMP) Cookbook*, SG24-6769
- ▶ *Cluster Systems Management Cookbook for pSeries*, SG24-6859
- ▶ *Effective System Management Using the IBM Hardware Management Console for pSeries*, SG24-7038
- ▶ *Partitioning Implementations for IBM eServer p5 Servers*, SG24-7039
- ▶ *Using WebSphere Extended Deployment V6.0 To Build an On Demand Production Environment*, SG24-7153
- ▶ *IBM eServer Certification Study Guide eServer p5 and pSeries Enterprise Technical Support AIX 5L V5.3*, SG24-7197
- ▶ *Securing NFS in AIX: An Introduction to NFS v4 in AIX 5L Version 5.3*, SG24-7204
- ▶ *LPAR Simplification Tools Handbook*, SG24-7231
- ▶ *Hardware Management Console V7 Handbook*, SG24-7491
- ▶ *PowerVM Virtualization on IBM System p: Introduction and Configuration Fourth Edition*, SG24-7940

- ▶ *IBM System p Advanced POWER Virtualization (PowerVM) Best Practices*, REDP-4194
- ▶ *Case Study: AIX and WebSphere in an Enterprise Infrastructure*, REDP-4436

Other publications

These publications are also relevant as further information sources:

- ▶ *IBM Java 5 Diagnostics Guide*, SC34-6550-07
- ▶ *IBM SDK User Guide*
<http://www-128.ibm.com/developerworks/java/jdk/aix/>
- ▶ Java Virtual Machine Specification
<http://java.sun.com/docs/books/jvms/>
- ▶ Java Language Specification
<http://java.sun.com/docs/books/jls/>
- ▶ *CSM for AIX 5L and Linux V1.5 Planning and Installation Guide*
<http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/topic/com.ibm.cluster.csm.doc/csm15/am7i112019.html?>
- ▶ *IBM Reliable Scalable Cluster Technology Administration Guide*
<http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.rsct.doc/rsctbooks.html>
- ▶ “Java Technology; IBM Style: Class Sharing”
<http://www-128.ibm.com/developerworks/java/library/j-ibmjava4/>
- ▶ “Java Technology; IBM Style: Garbage collection policies, Part 1”
<http://www-128.ibm.com/developerworks/java/library/j-ibmjava2/index.html>
- ▶ “Java Technology; IBM Style; Garbage Collection policies, Part 2”
<http://www-128.ibm.com/developerworks/java/library/j-ibmjava3/index.html>
- ▶ “Guide to Multiple Page Size Support on AIX 5L Version 5.3”
http://www-03.ibm.com/servers/aix/whitepapers/multiple_page.pdf
- ▶ *Transactional high availability and deployment considerations in WebSphere Application Server V6*
http://www-128.ibm.com/developerworks/websphere/techjournal/0504_beaven/0504_beaven.html

- ▶ *Setting up secure script execution between SSH clients and the HMC*
<http://publib.boulder.ibm.com/infocenter/eserver/v1r3s/topic/iphai/settingupsecurecriptexecutionsbetweensshclientsandthehmc.htm>
- ▶ *passAIX ssh client to pSeries HMC*
<http://www-03.ibm.com/support/techdocs/atsmastr.nsf/032f6e163324983085256b79007f5aec/f962418b4b10f21c86256dc6004abcfc?OpenDocument>
- ▶ WebSphere Application Server, Version 6.1 Information Center
<http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp>
- ▶ IBM Java Virtual Machine Infocenter documentation explaining JAVADUMPs
http://publib.boulder.ibm.com/infocenter/javasdk/v5r0/index.jsp?topic=/com.ibm.java.doc.diagnostics.50/diag/tools/javadump_tags_info.html
- ▶ IBM Systems Information Center
<http://publib.boulder.ibm.com/infocenter/systems/index.jsp>
- ▶ *Apache: The Definitive Guide*, O'Reilly, ISBN 0596002039
<http://www.oreilly.com/catalog/apache2/>

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

Numerics

32-bit memory management 193

A

abandonment rate 405
Active Directory Server 245
active-active performance 299
adapter_duplex 471
adapter_speed 471
adapter_type 471
Adding HMC Managed Partitions as Nodes 88
admin-authz.xml 158
Administrative Console 30
Administrative Security 428
Advanced Partition Operations 55
Advanced VIO setup 386
Agent Factory 273
AIO device 131
AIX generic installer 128
AIX startup and runtime settings 133
aix.systemlaunch.properties file 206
AIXTHERAD_COND_DEBUG 134
AIXTHREAD_MNRATIO 147, 211
AIXTHREAD_MUTEX_DEBUG 134
AIXTHREAD_RWLOCK_DEBUG 134
AIXTHREAD_SCOPE 134, 147, 211
allocation failure 306
Allocator 306
AllServiceableHardwareEvents 110
AllServiceableSwitchEvents 110
alog 133
Apache Axis 145
ApacheBench 118
Application Server Node name 431
Application Server Profile name 430
applicationLoginConfig 180
architecture 278
Asynchronous check removal 199
Asynchronous IO capabilities 128
AuditLogServiceableEvents 111
authConfig 180
authMechanisms 179
Availability 336

B

backupConfig.sh 94
backupos 83
Base Operating System Samples 132
benchmark 397
BigDecimal class 192
bindprocessor 416
Block hoisting 199
boot volume 75
BOOTSTRAP_ADDRESS 176–177
buffer sizes 166
Bulletin Board Factory 273

C

C++ 143
cable_type 471
cache block 306
cache statistics 204
cachespec.xml file 290
Capacity on Demand (CoD) 126
CAR 21
CDSBundleClassLoader 230
CEC 87
cell configuration file 244
Cell Manager 348
Cell name 431, 438
Cell Profile 430
CELL_DISCOVERY_ADDRESS 178
cell.xml 158
Central Electronics Complex 87
Charting 31
checkacquire 59
checking the aio device 131
checkrelease 59
chhwres 46, 61
chsysstate 46, 58
CICS® 146
CIP 21
Cisco LocalDirector 286
class sharing 304
Classloaders 190
cluster 4
cluster failover

- IP-based 336
 - non-IP based 336
- Cluster System Manager 86
- Cluster VLAN 86
- cluster.xml 158
- Clustering 289
 - for availability 336
 - horizontal 363
 - isolation 363
 - on micropartitions 362
 - Options 362
 - options 337
 - vertical 363
 - WebSphere 337
 - WebSphere Application Server 361
- Clustering for scalability and failover 12
- CoD 126
- Cold block outlining 199
- Co-locating WebSphere Applications 369
- com.ibm.cds_1.0.0.jar 155
- com.ibm.cds_1.0.0.jar file 202
- com.ibm.jdt.core_6.1.0.jar 155
- com.ibm.ws.runtime_6.1.0.jar 155, 234
- commands
 - backupios 83
 - bindprocessor 416
 - chhwres 61
 - csn2nimnodes 90
 - csmbbackup 92
 - csmssetupnim 90
 - csnstat 91
 - db2 400
 - db2cmd 399
 - definnode 89
 - drmgr 59
 - dsh 394
 - geninstall 128
 - getadapters 89
 - lparstat 416
 - lsaudrec 111, 121
 - lscondition 107, 120
 - lsdev 80
 - lshmc 45
 - lshwres 66
 - lslpp 128
 - lspv 74
 - lsresponse 107
 - lssyscfg 48
 - mirrorios 84
 - mkcondition 113
 - mkresponse 113
 - mksensor 119
 - mktcpip 79
 - mkvg 75
 - rpower 88–89
 - setupCmdLine.sh 249
 - startcondresp 115, 120
 - startManager.sh 402
 - startNode.sh 402
 - systemid 88
 - wmlassign 370
 - wsadmin.sh 401
 - xiplm 365
- Common sub-expression elimination 199
- Compaction phase 307
- Compiler Optimization 200
- compilers 199
- Complex remote command execution 58
- concurrent users 405
- configuration archive 21
- ConsoleMethod 88
- Container Startup 263
- Copy propagation 199
- CORBA IIOP 146
- coregroup.xml 158
- coregroups 162
- CPU Shares 366
- CPU utilization 406, 417
- CPUoD 348
- cryptography 178
- CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESSES 177
- CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESSES 177
- CSM 86, 386, 423
 - performance monitoring 124
- CSM / RSCT Monitoring 122
- CSM Adapter Definition File 423
- CSM Master 87
- CSM WebSM access 104
- CSM_ADAPTERS_STANZA_FILE 471
- csn2nimnodes 90
- csmbbackup 92
- csmssetupnim 90, 101
- csnstat 91
- customized installation package 21

D

- Data Replication Service 17
- DATA_REPLICATION 174
- database 399
- database connectivity 400
- Database persistence 18
- Database Tier 299
- DB2 387
- db2 shell 400
- DB2 universal JDBC Type 4 driver 399
- DB2/UDB V9.1 399
- db2cmd 399
- db2inst1 399
- dcp 93
- DCS 292
- DCS_UNICAST_ADDRESS 177, 295
- DDL 156
- Dead tree removal 199
- deadlock 226
- Default Ports 432, 439
- Default Profile 430, 437
- default profiles
 - AppSrv01 155
 - Dmgr01 155
- DefaultCoreGroup 162
- definenode 89
- delta 368
- DEP_EXEMPT 197
- Deployment Manager 336, 388
- Deployment Manager Node name 431, 438
- Deployment Manager Profile name 430
- deploytool 152
- derby 152
- determining 311
- Device slot Numbers 70
- disaster
 - planning 345
- Disaster Recovery 342
- Disk resource 374
- Distributed Consistency Services 292
- Distributed Copy 93
- Distributed Replication Services 273
- Distributed Shell 92
- Distributing workloads 16
- DLPAR processes resource dependencies 59
- DLPAR Support 210
- dmgr 5
- DMgr profile 151
- DR 342

- drmgr 5, 59
- DRS 17
- DRS_CLIENT_ADDRESS 177
- dsh 92, 394
- dynacache 289
- DynacacheESI 290
- Dynamic clusters 27
- Dynamic LPAR 125
- Dynamic LPAR assignments 64
- Dynamic partition profiles 64
- Dynamic Provisioning 3
- dynamic recompilation 198
- Dynamic Resource Allocation 4
- Dynamically changing resources 61

E

- EAR 21
- Easy Startup 96
- Eclipse 141
- Eclipse registry 258
- Eclipse/OSGI 239
- Edge Side Include Caching 290
- EJB 145
- EJB caching 398
- EJB Query Language 146
- EJB requests
 - distribution 16
- ejbdeploy 398
- Embedded Messaging Publish and Subscribe 147
- endPoint 175
- endPointName 167
- Enterprise application archive 21
- Enterprise Java Beans 145
- Entitled capacity delta 367
- entitlement 367
- Entity Bean 145
- Equinox 239
- ERRM 110
- Escape analysis 199
- Event-response resource manager 110
- Execution 191
- exname option 391
- Exporting physical storage 72
- Exporting VIO disk mapping 74
- Extended Deployment (XD) considerations 24
- Extended manageability 30
- Extensions 375
- EZNIM 96

F

- fail-over 375
- failover 336
- Fast Response Cache 286
- feature.xml 153
- Federated Repositories option 179
- File store considerations 19
- filesystem 391
- Fix Pack 423, 455
 - installation 397
- fix pack 397
- FRCA kernel extension 285
- fractrl 285, 289
- Full Speed Debug 199
- fusion fast interconnect 299

G

- Garbage Collection 209
 - Compaction Phase 307
 - Mark phase 307
 - Sweep phase 307
- garbage collection 304
 - additional runtime options 313
- Garbage Collection policy
 - choosing 305
- Garbage Collector 306
- GatherSSHHostKeys 110
- GC policy 305
 - Generational Concurrent 310
 - Optimal Average Pause 310
 - Subpool 313
- gencon 310
- Generational Concurrent 305
 - runtime options 312
- Generational Concurrent GC policy 310
 - tuning 311
- geninstall 128
- getadapters 89
- granularity 368
- Gridscale 299
- Group Manager, 273
- GSSAPI 238

H

- HA 335
- HACMP 298, 336, 348
- HADR 299
- HAManager 148, 292, 336

- HAManager runtime control 296
- hamanagerservice.xml 158
- Heap 306
- heap 194
- Heap sizing 304, 316
- heartbeat 293
- High Availability 335
- High Availability Deployments 276
- HMC 364
 - command line interface 44
 - setup and access 38
- horizontal cluster 363
- Horizontal scaling 13, 338
- Host name 431, 437
- Hot Code Replace 199
- hot methods 198
- hscroot 41
- HTTP 144
- HTTP requests
 - distribution 15
- httpd.conf file 285
- HyperVisor 66, 232

I

- I/O mechanisms 150
- IBM J9 JVM 149
- IBM Java 5 Diagnostics Guide 307
- IBM Java Virtual Machine 303
- IBM_JAVACOREDIR 215
- IETF RFC 2617 238
- IHS 285
- IMS 147
- Incremental Install 426
- inittab 285
- Inlining 199
- Install Location 427
- install_gateway 471
- installableApps 153
- installation 385
- installation response file 424
- installedApps 153
- installed-channels.xml 158
- installedConnectors 153
- installedFilters 153
- installing AIX and CSM on cluster nodes 99
- Installing Update 397
- interface_type 471
- Inventory Scout 39

INVOKEVIRTUAL 192
iocp device 130
iocp fileset 128
IP-based cluster failover 336
IPlatformRunnable interface 255
Isolation 363
 in High Availability 339

J

J2C 147
J2EE 143–144
J9 Java Virtual Machine Architecture 189
J9 JVM 149
JAAS 238
jacl 156
jacl scripts 398
Java
 defined 143
Java Authentication and Authorization Service 180
Java Connector Architecture 147
Java heap 306
Java memory management 306
Java Naming and Directory Interface 147
Java Native Interface 191
Java Server Pages 145
Java singleton components 148
Java Virtual Machine 303
 overview 189
Java Virtual Machine settings page 308
JAVA_DUMP_OPTS 215
java.lang.management 210
java.security.provider standard 178
Java2 Extended Edition, 144
JAVACORE 215
JAVADUMP 215
javasharedresources 206
javasharedresources file 201
JAX-RPC 398
JAX-WS 145
JCA 147
JDBC 146, 456
JFAP 236
JIT 305
JMS 146
JMS Messaging Engine 147
JMS resource 456
JMS/MQ message 292

JNDI 147
JNI 191
JNI Native Shared Object Libraries 243
JSF-Portlet bridge 154
JSP 144
Just-In-time (JIT) Compiler
 Optimization and Compilation 197
Just-In-Time compiler 304
JVM 303
 tuning 304
 using the large object area 315
JVM tuning 22, 303
JVM tuning capabilities 304
Jython 154

K

Kerberos 99, 174
key.p12 163
keyManagers 186
keySetGroups 186
keySets 186
keyStores 185

L

large object area 314
 sizing 315
large page sizes 304
latency 376
Lazy application start 27
LDR_CNTRL 194
libgetClasses.so 243
libibmaio.so 236, 243
libibmaiodbg.so 236, 243
libj9gc23.so 213
libj9jit23.so 213
libj9thr23.so 213
libNativeFile.so 243
libpmiJvmpiProfiler.so 243
libpmiJvmtiProfiler.so 244
libraries.xml 158
libSelector.so 236, 244
libUnixRegistryImpl.so 244
libWs60ProcessManagement.so 244
Lightweight Third Party Authentication 163
Limits 371
LinuxService Check 434
LinuxService User Name 434
LMB 368

- LOA 314
- load average 366
- Load balanced clustered data center 287
- logical memory block 368
- Logical Partition creation 53
- Logical Partitions
 - High Availability 339
- logical volume 70
- Loop unroller 199
- Loop versioning 199
- LPAR event handling
 - script-based 56
- lparstat 416
- lsaudrec 121
- lsaudrecl 111
- lscondition 107, 120
- lsdev 80
- lshmc 45
- lshwres 45, 66
- lspp 128
- lspv 74
- lsresponse 107
- lssensor 119
- lssyscfg 46, 48, 56
- ltpa.jceks 163

M

- MAC_address 471
- machine_type 471
- mallocoptions 210
- malloctype 210
- managed system operation 46
- Managed System profile 49
- Management VLAN 86
- managementScopes 184
- manageprofiles.sh 155
- Managing physical storage 72
- MANIFEST.MF file 234, 253
- Mapping virtual devices 80
- Mark phase 307
- Mark Sweep Compact 307
- maxima 371
- Memory 51
- Memory Allocation Control 210
- Memory Management 192
- Memory minima 372
- Memory resource 374
- Memory shares 366

- Memory-to-memory session replication 17
- Message Driven Beans 146
- messageBrokerDomainName 174
- Messaging Publish and Subscribe 147
- metro-cluster clustering 292
- Micro-Partitioning 50, 125
- micropartitions
 - clustering 337
- migration 347
- minima 371
- Minimum entitlement 367
- mirrorios 84
- mkcondition 113
- mkresponse 113
- mksensor 119
- mktcpip 79
- mkvg 75
- monitoring 112
- mount 391
- MQ 298
- MSC 307
- multibroker.xml 159

N

- Nagle 282
- namestore.xml 159
- naming-authz.xml 159
- NAS 336
- netaddr 471
- Network Buffer Cache 286
- Network Deployment 151
- Network Installation Manager 96
- Networked File Systems 336
- Networking on the VIO Server 77
- NFSRoot 391
- NFSv4 156, 292, 391
 - setup 387
- NIM 69, 96, 387
- NIM client-request processing
 - tuning 97
- NIM installation 387
- nimesis daemon 97
- nimmksysb 393
- node 4
- Node Agent 336
- Node groups 25
- Node Manager 348
- Node Port File 432

- Node Starting Port 432
- NODE_DISCOVERY_ADDRESS 177
- NODE_IPV6_MULTICAST_DISCOVERY_ADDRESS 177
- NODE_MULTICAST_DISCOVERY_ADDRESS 177
- NodeChanged 110
- NodeFullInstallComplete 110
- nodegroups 162
- NodeManaged 110
- nodes.xml 159
- non-default options 393
- Non-IP cluster failover 336
- notify intervals 366–367
- nursery 310
- nursery space
 - sizing 312

O

- Oak 143
- ObjectGrid 34
- Omit Action 436
- OnTap operating system 292
- Open Service Gateway Initiative 240
- OpenSSH 45, 99
- OpenSSL 99
- operating system
 - system installation 387
- optavgpause 310
- Optimal Average Pause GC policy 310
- Optimal store placement 199
- Optimal Throughput 305
- optimizer 200
- optionalLibraries 154
- optthruput 308
- Oracle 146
- orb
 - ObjectRequestBroker service 165
- ORB_LISTENER_ADDRESS 176, 178
- org.eclipse.jdt.core_3.1.2.jar 155
- org.eclipse.osgi_3.1.2.jar 155
- OSGI 141, 240
- OSGi framework 149
- osgiConsole.sh 261
- outermost element 178
- OutOfMemoryError 314

P

- Page sizes 209
- page steal 367
- Partial redundancy elimination 199
- Partition
 - definitions 391
- Partition Load Management 364
 - parameters table 365
- Partition Load Manager 362
- Partition operations 49
- partition start up and shut down
 - automating 58
- partitioning facility 32
- performance monitoring with CSM 124
- Performance schedule
 - summary report 414
- Performance testing 405
- perftuners.xml 159
- Physical I/O 52
- PingTrade.ksh 118
- pinnable memory 59
- PKI provider 185
- Planned events 32
- Planning for disaster 345
- PLM 362
 - Tuneables 365
- PLM capabilities 364
- PLM Configuration 365
- plugin-cfg.xml 155, 159, 398
- plugin-cfg.xml file 285
- plug-ins 152
- PME 153
- pmi-config.xml 159
- pmirm.xml 159
- policies
 - Resource management 365
- policy_file 365
- Port File 432, 438
- Portlet Container 142
- postrelease 59
- preacquire 59
- prerelease 59
- printAllStats 204
- processor maximum 372
- Processors 51
- Profile Creation Selection 428
- Profile name 437
- Profile path 430, 437
- profiles 151

- programming model extensions 153
- Provisioning 84
- provisioning
 - hardware level 125
- Provisionnode script 91
- Public VLAN 87
- Publish 147

Q

- Queuing and clustering considerations 9

R

- RAC 299
- RAMClass 201
- RAS 196
- Rational Performance Tester 385, 389
- rc.was 134
- rconsoleUpdateResponse 110
- Real Application Clustering 299
- recovery 346
- Redbooks Web site 477
 - Contact us xiv
- Redundancy 341
- redundant configuration 342
- Redundant monitor elimination 199
- Reliable Multicast Messaging 276
- Reliable Scalable Cluster Technology 106
- remembered set 313
- remote command execution 56
- request rate 405
- resident set size 209
- resilience 273
- Resource management policies 365
- Resource Monitor & Control RSCT subsystem 106
- Resource Monitoring and Control 38
- Resource sets 372
- resources.xml 160
- resources-cei.xml 159
- resources-pme.xml 160
- resources-pme502.xml 160
- response condition 120
- response file 423, 455
- Response Time 418
- RFC 2246 238
- RFC 2617 238
- Rich Client Platform 252
- RMC 38, 106
- RMM 276

- ROMClass 191, 201
- rpower 88–89
- RSCT 106
- RSS 209
- runAsGroup 174
- runAsUser 174
- RunCFMToNode 110
- Runtime map 31
- Runtime Startup 244
- Runtime topology 30
- RuntimeBundleActivator 253–254

S

- sample files
 - CSM Adapter Definition File 423
 - Trade 6.1 Installation script
 - trade.jacl 423
 - Update Installer
 - response.txt 423
 - WebSphere
 - Responsefile.nd.txt 423
 - WebSphere V6.1 Fix Pack 2
 - fp2.response.txt 423
- Sample Websphere installation 389
- SAN 72, 336
- SAP 147
- sar2war_tool 156
- SAS_SSL_SERVERAUTH_LISTENER_ADDRESS 177
- sb_max 137
- Scalability considerations 8
- scavenger 312
- scenario 385
- Scenario testing 415
- SCSI adapter 52
- SEA adapter device 77
- Security Options 429
- security.xml 160, 178
- Sensor command 118
- server 5
- Server virtual adapter 76
- SERVER_LOG_ROOT 165
- server.xml 155, 158, 160, 163
- serverEntries 175
- serverindex.xml 158, 161, 175
- serverType 175
- Service Agent 39
- Service Focal Point 39

Service Integration Bus 142
 service workload management 373
 Servlet Caching 289
 Session Bean 145
 Session Initiation Protocol 142
 Session persistence 17
 session replication
 Memory-to-memory 17
 sessionDRSPersistence 174
 sessionPersistenceMode 174
 setting up 112
 setupCmdLine.sh 249
 SetupSSHAndRunCFM 111
 Shared classes 196
 shared classes 305
 Shared processor 362
 Shares 370
 SIB_ENDPOINT_ADDRESS 177
 SIB_ENDPOINT_SECURE_ADDRESS 178
 SIB_MQ_ENDPOINT_ADDRESS 178
 SIB_MQ_ENDPOINT_SECURE_ADDRESS 178
 sib-service.xml 161
 SIBus 398
 SIGQUIT 215
 silent install 424, 452
 Silent installation mode 393
 Simple WebSphere Authentication Mechanism 183
 Simplifier 199
 single point of failure 19
 singleton 148
 SIP 156
 SIP container 149
 SIP servlet archive (sar) 156
 SIP_DEFAULTHOST 177
 SIP_DEFAULTHOST_SECURE 177
 Slot numbers 70
 SMT 232
 SOA 315
 SOAP 398
 SOAP_CONNECTOR_ADDRESS 177
 Sockets 128
 specialEndPoints 175
 SPINLOOPTIME 134
 SPNEGO 174
 SPNEGO TAI 174
 SPOF 19
 SQL authorization ID 400
 SQL Server 146
 SSL connection 41
 SSL/TLS 182
 sslConfigGroups 188
 Stack Based 191
 Stack Execution Disable 197
 Standalone server 392
 standby 343
 startcondresp 115, 120
 Starting Port 432, 438
 startManager.sh 402
 startNode.sh 402
 startup and runtime settings
 AIX 133
 startup.jar 251
 Stop-The-World collector 307
 StreamRedirect 171
 Struts 145, 154
 subclasses 370
 subnet_mask 471
 Subpool 305
 Subpool GC policy 313
 Subscribe 147
 Superclasses 370
 Sweep phase 307
 Switch analysis 199
 Symmetric Multi-Threading 232
 synchronization 226
 systemapps.xml 161
 systemid 88

T
 Table.ddl 399
 TAM 180
 tcp_keepidle 135
 tcp_keepinit 135
 tcp_keepintvl 135
 tcp_mssdflt 137
 tcp_recvspace 137
 tcp_sendspace 137
 tcp_timewait 135
 TCP/IP buffer size 136
 tcp/ip network settings 135
 tenure space
 sizing 311
 tenure space size 311
 tenured 310
 thread local heap 306
 Thread Management 211
 threadPool 166

- threadpoolmanager 170
- Tiers 371
- TimeBomb 252
- TLH 306
- tmsStorage 156
- tmx4j 152
- Tomcat 145
- Trade 6.1
 - installation and configuration 397
- Trade 6.1 Installation script
 - trade.jacl 423
- trade.jacl 456
 - customization 401
- Trade61 387
- tradeTech.pdf 398
- Transaction Manager 293
- transaction recovery 19
- transactions 19
- transport channels 166
- trust.p12 163
- trustAssociation 179
- trustManagers 186
- Tuneables 365
- tuning 304
 - Java virtual maching 303
- Tuning Format 4

U

- UDDI 153
- UDDI registr 156
- uncapped partitions 362
- Unexpected workload changes 373
- UNIVERSAL_JDBC_DRIVER_PATH 402
- Unplanned events 32
- Update Installer 452
 - response.txt 423
- URLClassLoader 201
- userRegistries 179–180
- utilization 367

V

- Validate Ports 433, 439
- Value propagation 199
- variables.xml 161
- verbosegc 311
- vertical cluster 363
- Vertical scaling 12
- Vertical stacking 27

- VIO disk mapping 74
- VIO Server
 - backup and availability 83
 - installing 69
- Virtual adapter slot 77
- Virtual I/O 52
- Virtual I/O Server 65
 - installing using NIM 100
- Virtual I/O Server configuration 65
- virtual memory size 209
- virtual target device 70
- virtualhosts.xml 161
- volume group 70
- VSCSI 71

W

- WAS component startup 264
- WAS runtime startup 258
- WAS_Prod_High 65
- WAS_Prod_Med 65
- WASRegistry 151
- WASView plugin 260
- WC_adminhost 177
- WC_adminhost_secure 177
- WC_defaulthost 177
- WC_defaulthost_secure 177
- WCChannelLink 266
- Web SM 415
- webAuthAttrs 184
- WebServer Check 435
- WebServer Hostname 435
- WebServer Install Path 436
- WebServer Name 435
- WebServer OS 435
- WebServer Plugin Path 436
- WebServer Port 436
- WebServer Type 435
- WebSM 39, 87
 - HMC access 39
- WebSphere
 - Responsefile.nd.txt 423
- WebSphere Application Server
 - Clustering 361
 - installing 393
- WebSphere Application Server ND 162
- WebSphere Applications
 - Co-locating 369
- WebSphere Applicaton Server on AIX

- disk layout 151
- WebSphere Clustering 337
- WebSphere Edge Server 398
- WebSphere Extended Deployment 24
- WebSphere eXtreme Scale 32
- WebSphere MQ Server 147
- WebSphere Partitioning Facility 32
- WebSphere Partitions
 - installing using NIM 101
- WebSphere V6.1 Fix Pack 2
 - fp2.response.txt 423
- WebSphere Virtual Enterprise 24
- WebSphere XD
 - Health policy 29
 - Service policy 29
- WEMPS 147
- which_fileset 132
- Windows XP 389
- WinService Account Type 433
- WinService Check 433
- WinService Password 434
- WinService Startup Type 434
- WinService User Name 433
- WLM 4, 15, 362, 368
 - Automation support 374
 - Changes to startServer.sh 375
 - configuration 369
 - disk resource 374
 - Extensions 375
 - Fail-over 375
 - Limits 371
 - Memory resource 374
 - Resource sets 372
 - Shares 370
 - subclasses 370
 - Tiers 371
 - Unexpected workload changes 373
- wmlassign 370
- Workload management 15
- Workload Management feature 368
- Workload Manager 362
- WPF 32
- wsadmin.sh 401
- wsCertificateExpirationMonitor 188
- WSDL 398
- WSLauncher class 252, 255
- wsNotifications 188
- WSPreLauncher code 251
- wsSchedules 188

- WS-Security 152
- ws-security.xml 161
- WsServerLauncher class 206
- wstemp 157

X

- X509 185
- XA distributed transaction management 156
- XAResource Coordinator 292
- XCOFF header 197
- Xconcurrentbackground 314
- Xconcurrentlevel 314
- Xconmeter 316
- XD 4, 24
- Xdisableexcessivegc 314
- Xdisableexplicitgc 314
- Xgcpolicy 308, 310
- Xgcthreads 314
- Xloainitial 315
- Xloamaximum 316
- Xlp64K 209
- xlplm 365
- Xnopartialcompactgc 314
- Xnosigcatch 211
- Xpartialcompactgc 314
- Xscmx 202
- Xshareclasses 202
- Xsigcatch 211



Redbooks

Running IBM WebSphere Application Server on System p and AIX: Optimizaton and Best Practices

(1.0" spine)
0.875" <-> 1.498"
460 <-> 788 pages



Running IBM WebSphere Application Server on System p and AIX: Optimization and Best Practices

**System p and AIX
configuration
strategies for
WebSphere
Application
Server**

**How JVM runtime
and WebSphere
Application
Server interact
with AIX**

**Implementation
scenarios**

This IBM Redbooks publication describes how to run the IBM Java Virtual Machine for AIX and WebSphere Application Server V6.1 on IBM System p and the AIX 5L Operating Systems. In terms of provisioning, tuning and maintenance, it consolidates information from all of these areas into a single resource and explains how you can implement, tune, and utilize the unique features of the IBM POWER Systems platform, AIX, and WebSphere Application Server together for maximum optimization. The book is intended for UNIX system administrators, Java developers, infrastructure designers, J2EE architects, project managers, performance testers and anyone who runs WebSphere Application Server on System p and AIX. It may contain some information which you already know, and other information that is new to you, depending on your background. AIX system administrators may be expert in configuring logical partitions and advanced virtualization, but may gain an understanding from this book about how WebSphere deployment teams may be able to exploit the features of IBM POWER Systems™ and AIX. WebSphere infrastructure architects may already know exactly how they want their redundant systems to work, but might learn how AIX teams can provide two or three physical servers that provide all of the different levels of application services necessary for the entire application lifecycle environment.

**INTERNATIONAL
TECHNICAL
SUPPORT
ORGANIZATION**

**BUILDING TECHNICAL
INFORMATION BASED ON
PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks