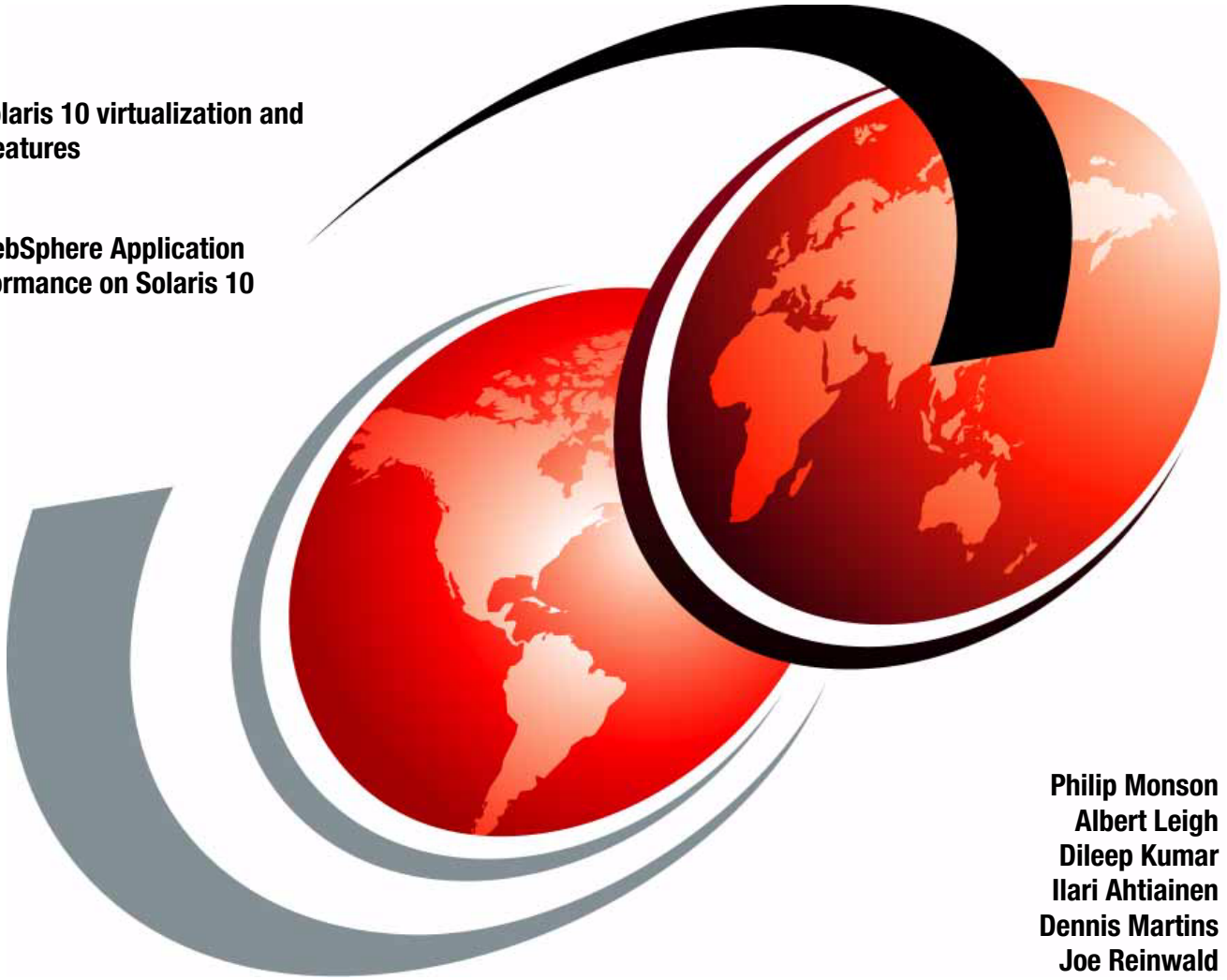
WebSphere. software

IBM

IBM WebSphere Application Server V6.1 on the Solaris 10 Operating System

- Learn detailed deployment and configuration strategies
- Leverage Solaris 10 virtualization and other new features
- Optimize WebSphere Application Server performance on Solaris 10



Philip Monson
Albert Leigh
Dileep Kumar
Ilari Ahtiainen
Dennis Martins
Joe Reinwald

Redbooks

ibm.com/redbooks

International Technical Support Organization

IBM WebSphere Application Server V6.1 on the Solaris 10 Operating System

March 2008

SG24-7584-00

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

First Edition (March 2008)

This edition applies to Version 6.1 of IBM WebSphere Application Server.

Contents

Notices

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

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

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

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

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

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

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

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

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

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which 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

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

AIX®
BladeCenter®
Cloudscape®
CICS®
Domino®
DB2 Universal Database™
DB2®
Hypervisor™
Informix®
IBM®
IMS™
iSeries®
Lotus®
Passport Advantage®
Rational®
Redbooks®
Redbooks (logo)  ®
System x™
Tivoli®
WebSphere®
z/OS®
zSeries®

The following terms are trademarks of other companies:

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

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

AMD, AMD Opteron, the AMD Arrow logo, and combinations thereof, are trademarks of Advanced Micro Devices, Inc.

CoolThreads, Enterprise JavaBeans, EJB, Java, Java HotSpot, JavaBeans, JDBC, JDK, JMX, JNI, JRE, JSP, JVM, J2EE, J2SE, NetBeans, N1, OpenSolaris, Solaris, Sun, Sun BluePrints, Sun Fire, Sun Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

Intel, Intel Xeon, 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

IBM® WebSphere® Application Server is a key building block of the IBM SOA reference architecture, providing a Java™ 2 Platform, Enterprise Edition (J2EE™) offering for assembling, deploying and managing applications. With standards-based messaging and support for the latest Web services standards, WebSphere Application Server enables you to reuse existing assets and helps you increase your return on existing investments.

This IBM Redbooks® publication highlights how WebSphere Application Server can be optimized with one of the many operating systems it supports: the Solaris™ 10 Operating System (OS).

The Solaris OS, developed by Sun™ Microsystems and now the foundation of the OpenSolaris™ open source project, complements the IBM WebSphere application environment by providing powerful, secure, and flexible infrastructure. Solaris is the leading operating environment to host demanding enterprise applications, including WebSphere Application Server, in today's challenging business environments. Solaris 10 is the most advanced and open UNIX®-based operating system that powers the most business-critical enterprise systems.

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, San Jose Center.

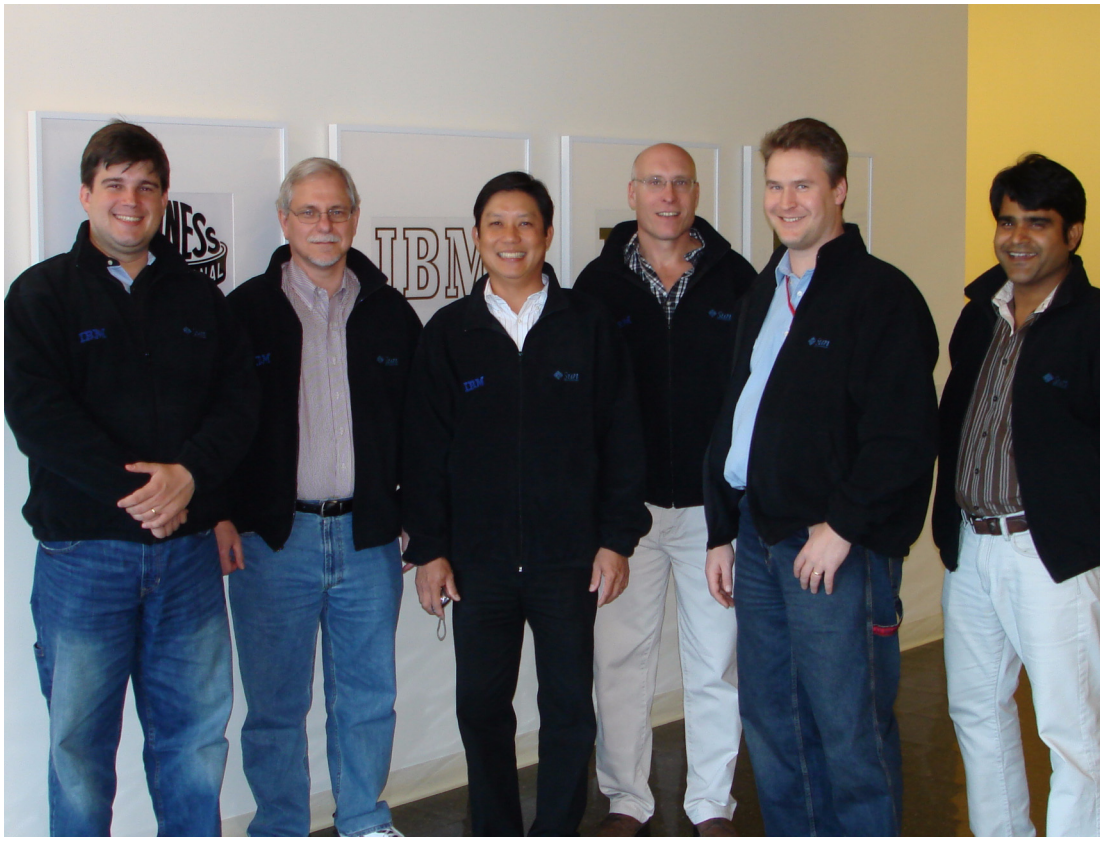


Figure 1 The team from left to right: Dennis Martins, Joe Reinwald, Albert Leigh, Phil Monson, Ilari Ahtiainen, and Dileep Kumar

Philip Monson is a Project Leader for the IBM ITSO. Phil has been with Lotus® / IBM for 17 years, joining the company when the early versions of Notes were rolled out for internal use. He has served in management, technical, and consulting roles in the IT, Sales, and Development organizations.

Albert Leigh is a Solution Architect in the Sun ISV Solution group at Sun Microsystems, Inc. Albert has been with Sun for ten years. He provides technical consultation worldwide in the development and deployment of customer solutions pertaining to enterprise Java application infrastructures, virtualization, and system performance for IBM WebSphere products on Solaris. He holds a M.S. degree in Computer Science from the University of Houston, Clear Lake. Prior to joining Sun, Albert worked on software development projects at NASA Johnson Space Center and has taught undergraduate courses at UHCL. He has written articles on Sun Web sites, technical journals, and also frequently blogs about WebSphere on Solaris at `http://blogs.sun.com/sunabl`.

Dileep Kumar is a Staff Engineer in the ISV Engineering Group at Sun Microsystems, Inc. He has over ten years of experience in the computer industry and now works on IBM WebSphere products on Solaris. His area of expertise includes Java and J2EE based system design and development, performance enhancement, and implementation. He holds a M.S. degree in Engineering Management from Santa Clara University, Santa Clara. Prior to joining Sun, Dileep worked on various software development projects at Netscape and various solution provider companies in India. Read Dileep's professional blog at `http://blogs.sun.com/dkumar`.

Ilari Ahtiainen is an IT Specialist who works closely with WebSphere Application Server and IBM HTTP Server products for IBM Finland. He has over nine years of IT experience, ranging from sales and support to designing and implementing complex J2EE infrastructures for customers. His specialties are WebSphere Application Server installations, WebSphere Application Server Plug-ins, and IHS (Web tier). He has a Bachelor of Science degree in Software Engineering from EVTEK University of Applied Sciences.

Dennis Martins is an IT Specialist working from Integrated Technology Delivery SSO, Brazil. He has 10 years of experience working in IT. He is an expert in IBM WebSphere Application Server and is certified in WebSphere Application Server V5 and V6. He has been working with WebSphere Application Server since V3.5 and mostly designs e-business solutions focused on using the WebSphere product family. His areas of expertise include the architecture, design, and development of J2EE applications, WebLogic Application Server, Solaris, AIX®, and Windows® platforms. Dennis holds a degree in Computer Science, a MBA in IT Strategic management, and a specialization in developing software degree with new technologies from Pontifical Catholic University, Rio de Janeiro.

Joe Reinwald is a Course Developer and Instructor for WebSphere Education, a team within the IBM Software Services for WebSphere organization. Over the past eight years, Joe has developed courses for WebSphere Application Server Administration, WebSphere Portal Administration, and WebSphere RFID Premises Server. Most recently, he has been the lead technical developer for both the WebSphere Application Server V6 and V6.1 Problem Determination courses. Joe has contributed to the development of several versions of the IBM Certification Tests for both WebSphere Application Server and WebSphere Portal System Administrators. In addition to development work, Joe has taught classes for WebSphere Application Server Administration, WebSphere Portal Server Administration, and WebSphere Application Server Problem Determination. Joe has delivered these classes to IBM customers and Business Partners, and has also enabled other instructors to teach the courses. Prior to joining IBM, Joe was a member of the Education Department at Transarc Corporation, which was acquired by IBM in 1999. Joe holds a BS degree in Mathematics and Physics from the University of Pittsburgh.

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

Carla Sadtler, Project Leader - WebSphere, IBM
International Technical Support Organization, Raleigh Center

William L. Gregory, WebSphere Web Components Development Manager, IBM Software Group

Richard Morgan, Global Alliance Manager - IBM, Sun Microsystems

Mike Nelson, Global Account Executive IBM SW Partner Sales, Sun Microsystems

Daniel Edwin, Technical Account Manager - IBM, ISV Engineering, Sun Microsystems

Angelo Rajadurai, Senior Staff Engineer, ISV Engineering, Sun Microsystems

Jeff Savit, Principal Engineer, Data Center Practice, Sun Microsystems

Bob Netherton, Principal Engineer, Data Center Practice, Sun Microsystems

Bhargava Yenduri, Staff Engineer, Solaris Software, Sun Microsystems

Larry Wake, Manager, Solaris Software, Sun Microsystems

Charlie Hunt, Staff Engineer, Java Engineering, Sun Microsystems

Tim Fors, Install Architect, IBM Canada

Art Jolin, Senior WebSphere Consultant - IBM

Vinicius de Melo Patrão, Web Hosting Support Team Leader, ITD - Global Delivery, IBM Brazil

Brian Doherty, Staff Engineer, Java Engineering, Sun Microsystems

Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will 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 will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Discover 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 books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

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

  ibm.com/redbooks

- Send your comments in an e-mail 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

1

Introduction to WebSphere Application Server V6.1 on Sun Solaris 10

IBM WebSphere products are architected to enable you to build business-critical applications for the Web. There are a wide range of WebSphere products that help you develop and serve Web applications. They are designed to make it easier for clients to build, deploy, and manage dynamic Web sites more productively.

In this chapter, we take a first look at WebSphere Application Server V6.1 and then, throughout the remainder of this IBM Redbooks publication, show how it can be deployed and optimized with one of the many operating systems it supports: the Solaris 10 Operating System (OS).

The Solaris OS, developed by Sun Microsystems and the foundation of the OpenSolaris open source project (see http://www.opensolaris.org), complements the IBM WebSphere application environment by providing powerful and flexible infrastructure. The Solaris OS is the leading operating environment to host demanding enterprise applications, including WebSphere, in today's challenging business environments. Solaris 10 is the most advanced and open UNIX-based operating system that powers the most business-critical enterprise systems. It delivers proven innovative features, security, performance, scalability, and reliability, availability, and serviceability (RAS).

Based upon open standards, the Solaris OS is architected for distributed network computing infrastructures, and has many features and functionalties to provide a highly reliable and scalable foundation for business computing. Solaris is supported on over 900 different types of x86 and SPARC systems from leading vendors, such as Sun, IBM, Fujitsu, HP, and Dell, including AMD64 and Intel® 64-based systems. Sun guarantees application binary compatibility for existing applications on forward releases, as well as source compatibility between SPARC and x86 through the Solaris Application Guarantee Program (its terms and conditions can be found at http://sun.com/solaris/guarantee).

With Solaris 10 and Sun revolutionary server technologies, Sun provides new ways to solve business problems, enable quicker time to market, and drive down operation costs in your enterprise.

1.1 WebSphere products overview

WebSphere is the IBM brand of software products designed to work together to help deliver dynamic e-business quickly. It provides solutions for connecting people, systems, and applications with internal and external resources. WebSphere is based on infrastructure software, or middleware, designed for dynamic e-business. It delivers a proven, secure, and reliable software portfolio that can provide an excellent return on investment.

The technology that powers WebSphere products is Java. Over the years, many software vendors have collaborated on a set of server-side application programming technologies that help build Web accessible, distributed, and platform-neutral applications. These technologies are collectively branded as the Java 2 Platform, Enterprise Edition (J2EE) platform. This contrasts with the Java 2 Standard Edition (J2SE™) platform, with which most clients are familiar. J2SE supports the development of client-side applications with rich graphical user interfaces (GUIs). The J2EE platform is built on top of the J2SE platform. J2EE consists of application technologies for defining business logic and accessing enterprise resources, such as databases, Enterprise Resource Planning (ERP) systems, messaging systems, e-mail servers, and so forth.

The potential value of J2EE to clients is tremendous. Among the benefits of J2EE are:

- An architecture-driven approach to application development helps reduce maintenance costs and allows for construction of an information technology (IT) infrastructure that can grow to accommodate new services.
- Application development is focused on unique business requirements and rules, such as security and transaction support. This improves productivity and shortens development cycles.
- Industry standard technologies allow clients to choose among platforms, development tools, and middleware to power their applications.
- Embedded support for Internet and Web technologies allows for a new breed of applications that can bring services and content to a wider range of customers, suppliers, and others, without creating the need for proprietary integration.

Another exciting opportunity for IT is Web services. Web services allow for the definition of functions or services within an enterprise that can be accessed using industry standard protocols that most businesses already use today, such as HTTP and XML. This allows for easy integration of both intra- and inter-business applications that can lead to increased productivity, expense reduction, and quicker time to market.

1.2 WebSphere Application Server

WebSphere Application Server provides the environment to run your Web-enabled e-business applications. An application server functions as *Web middleware* or a middle tier in a three-tier e-business environment. The first tier is the HTTP server that handles requests from the browser client. The third tier is the business database (for example, DB2® UDB for iSeries®) and the business logic (for example, traditional business applications, such as order processing). The middle tier is WebSphere Application Server, which provides a framework for a consistent and architected link between the HTTP requests and the business data and logic.

WebSphere Application Server is available on a wide range of platforms and in multiple packages to meet specific business needs. It also serves as the base for other WebSphere products, such as WebSphere Enterprise Service Bus and WebSphere Process Server, by providing the application server that is required to run these specialized applications.

Figure 1-1 illustrates a product overview of WebSphere Application Server.

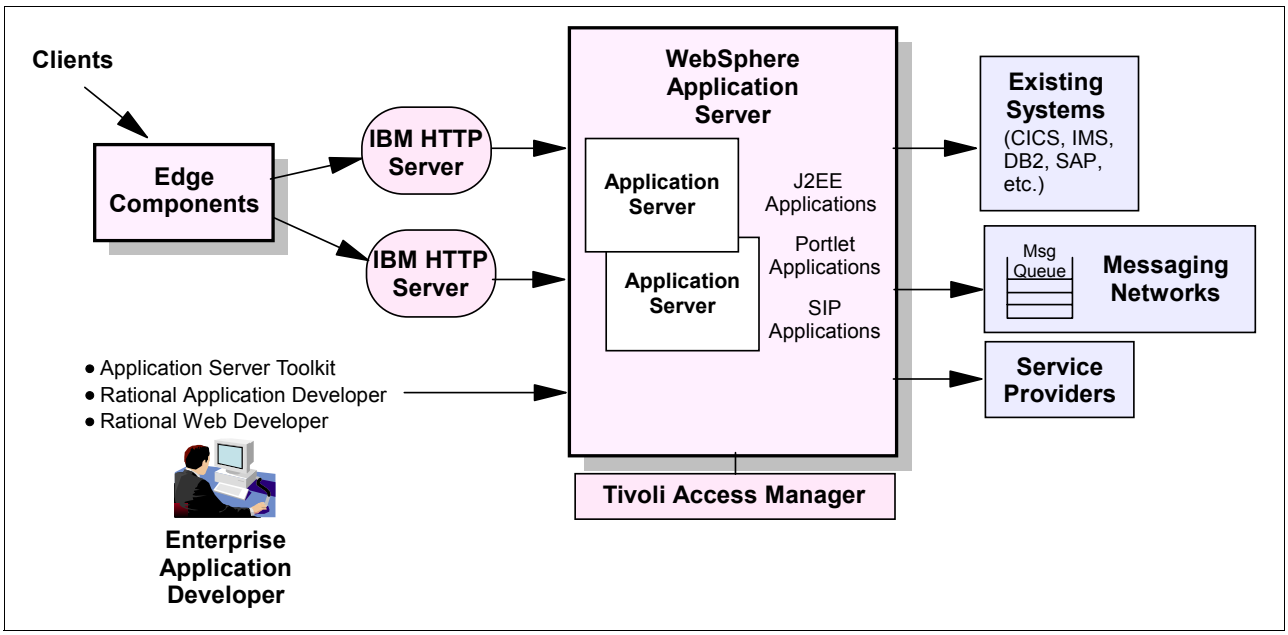


Figure 1-1 WebSphere Application Server product overview

The application server is the key component of WebSphere Application Server, providing the runtime environment for applications that conform to the J2EE 1.2, 1.3, and 1.4 specifications. Clients access these applications through standard interfaces and APIs. The applications, in turn, have access to a wide variety of external sources, such as existing systems, databases, Web services, and messaging resources that can be used to process the client requests. V6.1 extends the application server to allow it to run JSR 168 compliant portlets and Session Initiation Protocol (SIP) applications written to the JSR 116 specification.

With the Base and Express packages, you are limited to single application server environments. The Network Deployment package allows you to extend this environment to include multiple application servers that are administered from a single point of control and can be clustered to provide scalability and high availability environments.

WebSphere Application Server supports asynchronous messaging through the use of a JMS provider and its related messaging system. WebSphere Application Server includes a fully integrated JMS 1.1 provider called the default messaging provider. This messaging provider complements and extends WebSphere MQ and application server. It is suitable for messaging among application servers and for providing messaging capability between WebSphere Application Server and an existing WebSphere MQ backbone.

WebSphere Application Server provides authentication and authorization capabilities to secure administrative functions and applications. Your choice of user registries include the operating system user registry, an LDAP registry (for example, Tivoli® Directory Server), custom registries, file-based registries, or federated repositories. In addition to the default authentication and authorization capabilities, you have the option of using an external Java Authorization Contract for Containers (JACC) compliant authorization provider for application security. The IBM Tivoli Access Manager client embedded in WebSphere Application Server

is JACC-compliant and can be used to secure your WebSphere Application Server-managed resources. This client technology is designed to be used with the Tivoli Access Manager Server (shipped with Network Deployment).

WebSphere Application Server works with a Web server (such as the IBM HTTP Server) to route requests from browsers to the applications that run in WebSphere Application Server. Web server plug-ins are provided for installation with supported Web browsers. The plug-ins direct requests to the appropriate application server and perform workload management among servers in a cluster.

WebSphere Application Server Network Deployment includes the Caching Proxy and Load Balancer components of Edge Component for use in highly available, high volume environments. Using these components can reduce Web server congestion, increase content availability, and improve Web server performance.

1.3 Solaris overview

Sun Microsystem's Solaris Operating System is designed to deliver scalable performance, stability and security for the most demanding computing needs. With the largest application portfolio of any open operating system, it can be used throughout the entire enterprise, ranging from the Web tier and technical computer grids to the largest data warehouse and transaction processing applications.

Solaris can be deployed in diverse computing environments. Supported on over 900 Intel, AMD™, and SPARC platforms, it can be used on systems ranging from a single-CPU mobile or desktop computer to a enterprise-class server with hundreds of CPUs, yet exploit the price, performance, scale, or availability features of each platform. Solaris is designed to run efficiently on a single CPU desktop yet scale up to the largest Sun SPARC Enterprise Server with 64 dual core CPUs. Solaris is built from a single code base, even though it is built for different chip architectures, ensuring that system functionality is the same across different platforms.

This compatibility is a key Solaris feature. Solaris provides a high level of compatibility that lets customers leverage software investments over multiple generations of hardware and Solaris versions, without “binary breaks” that render existing production binaries unusable. Solaris provides an upwards-compatible application binary interface (ABI) engineered to ensure that binaries compiled for older versions of the operating system continue to work on later versions of that platform architecture. Sun provides a compatibility testing tool to detect application use of nonstandard APIs not belonging to the ABI, and guarantees that ABI-conforming applications will run on later versions of Solaris.

Compatibility is provided over time, so a binary compiled several years ago on an older UltraSPARC machine and an older version of Solaris will work on the latest SPARC-family processor and current Solaris, and over scale; applications running on a single-CPU SPARC workstation or low-end server will work on a 144-CPU Sun Fire™ E25K. Permitting customers to control when they upgrade their software portfolio, and minimizing the frequency with which they have to upgrade their applications, is an important benefit for cost savings and stability. Additionally, Solaris provides a common application programming interface (API) for both x86 and SPARC, so moving an application between the two instruction set architectures is typically no more than a re-compile. These considerations are directed towards compiled programs in languages like C or C++. Applications in Java run in the Java Virtual Machine, and are expected to run without change even across different processor families.

Solaris compatibility is not restricted to compatibility with itself. Sun Microsystems is a pioneer in open systems computing, from its outset based on the principle that a competitive computing marketplace based on open standards is the best way to generate quality and innovation, and reduce costs. Sun builds products in conformance with open standards, frequently leading standards efforts and developing innovations that are adopted by the entire computing world, such as NFS and Java. Sun also believes that vendors should compete for business based on superior implementations of public, stable interfaces, rather than proprietary lock-in. Solaris embodies this through its support of open standards, including X/Open, XPG3, XPG4, XPG4v2, POSIX, UNIX 98, UNIX 03 Product Standard, and POSIX threads.

Tip: From a Solaris console, issue the command **`man standards`** for further information about standards in Solaris.

At the same time, Sun continues to provide industry-leading operating systems innovations in Solaris. Solaris is designed in a modular fashion that permits innovation without invalidating existing parts of the OS. Operating systems are a key component of system infrastructure. Sun invests in research and development to invent new capabilities that add new value to the Solaris OS and help solve customer problems. Just a few of the innovative features added to Solaris 10 are:

- Solaris Containers: A form of light-weight virtualization, known as Solaris Zone, that creates private virtual environments without the impact seen in traditional virtual machine implementations, permitting hundreds or even thousands of virtual environments on the same Solaris instance.
- DTrace: A breakthrough tool for real-time system performance and problem diagnosis, which has made it possible for customers and Sun engineers to understand the dynamic behavior of their systems, netting dramatic improvements in debugging and performance improvement.
- Solaris ZFS: A new file system that provides superior data integrity, simpler management, capacity, and performance compared to traditional file systems.
- Configurable privileges and role-based access control (RBAC): Lets the systems manager control and audit access to system privileges, without the “all or nothing” privilege model of traditional UNIX.
- Predictive Self Healing: The Solaris Fault Management Architecture and Service Management Facility work together to add robustness to Solaris environments by providing the ability to proactively detect and mitigate hardware and software failures. These features provide the reliability, availability, and serviceability needed for mission-critical applications.

This is just a small subset of the innovations in Solaris, and Sun continues to do research to create more functionality.

At the same time, Sun continues to preserve the ABI that provides upwards compatibility. This makes Solaris the ideal robust, stable environment platform needed for production computing, while providing the innovation and added value needed for current and future business needs.

1.4 WebSphere packaging

Because varying e-business application scenarios require different levels of application server capabilities, WebSphere Application Server is available in multiple packaging options. Although they share a common foundation, each provides unique benefits to meet the needs of applications and the infrastructure that supports them. At least one WebSphere Application Server product fulfills the requirements of any particular project and its supporting infrastructure. As your business grows, the WebSphere Application Server family provides a migration path to more complex configurations.

WebSphere Application Server - Express V6.0

The Express package is geared to those who need to get started quickly with e-business. It is specifically targeted at medium-sized businesses or departments of a large corporation, and is focused on providing ease of use and ease of application development. It contains full J2EE 1.4 support, but is limited to a single-server environment.

WebSphere Application Server - Express is unique from the other packages in that it is bundled with an application development tool. Although there are WebSphere Studio and Rational® Developer products designed to support each WebSphere Application Server package, normally they are ordered independently of the server. WebSphere Application Server - Express includes the Rational Web Developer application development tool. It provides a development environment geared toward Web developers and includes support for most J2EE 1.4 features with the exception of Enterprise JavaBeans™ (EJB™) and J2EE Connector Architecture (JCA) development environments. However, keep in mind that WebSphere Application Server - Express V6 does contain full support for EJB and JCA, so you can deploy applications that use these technologies.

WebSphere Application Server V6.1

The WebSphere Application Server package is the next level of server infrastructure in the WebSphere Application Server family. Though the WebSphere Application Server is functionally equivalent to that shipped with Express, this package differs slightly in packaging and licensing.

This package includes two tools for application development and assembly:

- The Application Server Toolkit, which has been expanded in V6.1 to include a full set of development tools. The toolkit is suitable for J2EE 1.4 application development as well as the assembly and deployment of J2EE applications. It also supports Java 5 development.

  In addition, the toolkit provides tools for the development, assembly, and deployment of JSR 116 SIP and JSR 168 portlet applications.
- This package also includes a trial version of Rational Application Developer, which supports the development, assembly, and deployment of J2EE 1.4 applications.

To avoid confusion with the Express package in this IBM Redbooks publication, we refer to this as the Base package.

WebSphere Application Server Network Deployment V6.1

WebSphere Application Server Network Deployment is an even higher level of server infrastructure in the WebSphere Application Server family. It extends the WebSphere Application Server base package to include clustering capabilities, Edge components, and high availability for distributed configurations. These features become more important at larger enterprises, where applications tend to service a larger client base, and more elaborate performance and availability requirements are in place.

Application servers in a cluster can reside on the same or multiple machines. A Web server plug-in installed in the Web server can distribute work among clustered application servers. In turn, Web containers running servlets and Java Server Pages (JSPs) can distribute requests for EJBs among EJB containers in a cluster.

The addition of Edge components provides high performance and high availability features. For example:

- The Caching Proxy intercepts data requests from a client, retrieves the requested information from the application servers, and delivers that content back to the client. It stores cacheable content in a local cache before delivering it to the client. Subsequent requests for the same content are served from the local cache, which is much faster and reduces the network and application server load.
- The Load Balancer provides horizontal scalability by dispatching HTTP requests among several, identically configured Web server or application server nodes.

Packaging summary

Table 1-1 shows the features included with each WebSphere Application Server packaging option available for Solaris.

Table 1-1 WebSphere Application Server packaging for Solaris

Features included	Express V6.0[1]	Base V6.1	Network Deployment V6.1
WebSphere Application Server	Yes	Yes	Yes
Deployment manager	No	No	Yes
Web server plug-ins	Yes	Yes	Yes
IBM HTTP Server	Yes	Yes	Yes
Application Client (not available on Linux® for zSeries®)	Yes	Yes	Yes
Application Server Toolkit	Yes	Yes	Yes
DataDirect Technologies JDBC™ Drivers for WebSphere Application Server	Yes	Yes	Yes
Rational Development tools	Rational Web Developer (single use license)	Rational Application Developer Trial	Rational Application Developer Trial
Database	IBM DB2 Universal Database™ Express V8.2	IBM DB2 Universal Database Express V8.2 (development use only)	IBM DB2 UDB Enterprise Server Edition V8.2 for WebSphere Application Server Network Deployment
Production ready applications	IBM Business Solutions	No	No

Features included	Express V6.0[1]	Base V6.1	Network Deployment V6.1
Tivoli Directory Server for WebSphere Application Server (LDAP server)	No	No	Yes
Tivoli Access Manager Servers for WebSphere Application Server	No	No	Yes
Edge Components	No	No	Yes
1. Express is limited to a maximum of two CPUs.			

Note: Not all features are available on all platforms. See the System Requirements Web page for each WebSphere Application Server package for more information.

1.5 Solaris packaging

Solaris 10 media is available in one package, containing:

- One DVD for SPARC systems
- One DVD for x86 systems (including those with 64-bit AMD64 or Intel 64 processors)
- One DVD with the Sun Studio development tool suite

Note: This packaging may be different (for example, without the SPARC DVD) when IBM distributes with their own Solaris media kit for x86-based IBM System x™ servers and BladeCenter® servers.

You can also download Solaris DVD images at no charge from http://sun.com/solaris.

Sun also offers a kit with additional disks, including Sun Java™ System products. The Solaris 10 OS includes licenses to run some of these products. For example, Sun Java System Directory Server is the industry's most widely deployed, general-purpose, LDAP-based directory server. Solaris 10 includes a license for 200,000 directory entries. For more details, look at:

http://sun.com/software/solaris/what_you_get.jsp

1.6 WebSphere Application Server supported platforms and software

The following tables illustrate the platforms, software, and versions that WebSphere Application Server V6.1 supports at the time of the writing of this document. For the most up-to-date operating system levels and requirements, refer to the WebSphere Application Server system requirements at:

http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html

1.6.1 Web servers

The following Web servers are supported by WebSphere Application Server V6.1 on all available platforms:

- Apache HTTP Server 2.0.54
- IBM HTTP Server for WebSphere Application Server V6.0.2
- IBM HTTP Server for WebSphere Application Server V6.1
- Internet Information Services 5.0
- Internet Information Services 6.0
- Lotus Domino® Enterprise Server V6.5.4 or V7.0
- Sun Java System Web Server 6.0 SP9
- Sun Java System Web Server 6.1 SP3
- Sun Java System Web Server 7.0

1.6.2 Database servers

Table 1-2 shows the database servers that WebSphere Application Server V6.1 supports.

Table 1-2 Supported database servers and versions

Databases	Versions
IBM DB2	DB2 for iSeries V5.2, V5.3, or V5.4 DB2 for z/OS® V7 or V8 DB2 Enterprise Server Edition V8.2 FP4 DB2 Express V8.2 FP4 DB2 Workgroup Server Edition V8.2 FP4
Cloudscape®	Cloudscape 10.1
Oracle®	Oracle 9i Standard/Enterprise Release 2 - 9.2.0.7 Oracle 10g Standard/Enterprise Release 1 - 10.1.0.4 Oracle 10g Standard/Enterprise Release 2 - 10.2.0.1 or 10.2.0.2
Sybase	Sybase Adaptive Server Enterprise 12.5.2 or 15.0
Microsoft® SQL Server®	Microsoft SQL Server Enterprise 2000 SP4 Microsoft SQL Server Enterprise 2005
Informix®	Informix Dynamic Server 9.4C7W1 or 10.00C4
IMS™	IMS V8 or V9
WebSphere Information Integrator	WebSphere Information Integrator V8.2 FP4

1.6.3 Directory servers

Table 1-3 shows the LDAP servers that WebSphere Application Server V6.1 supports.

Table 1-3 Supported Directory Servers and versions

Directory Server	Versions
IBM Tivoli Directory Server	5.2 and 6.0
z/OS Security Server	1.6 and 1.7
z/OS.e Security Server	1.6 and 1.7
Lotus Domino Enterprise Server	6.5.4 and 7.0
Sun Java System Directory Server	5.1 SP4 and 5.2
Sun Java System Directory Server Enterprise Edition	6.0, 6.1 and 6.2
Windows Active Directory®	2003 and 2000
Novell eDirectory	8.7.3 and 8.8

1.7 Solaris supported platforms and WebSphere

The Solaris Operating System is supported on over 900 SPARC and x86/x64 systems from leading vendors such as Sun, IBM, Fujitsu, Dell, and HP. The detailed list of these supported platforms is available on the Sun Web site called “Solaris OS: Hardware Compatibility Lists” at http://sun.com/bigadmin/hcl.

Your system should have the following minimum hardware requirements to support WebSphere Application Server:

- SPARC, AMD Opteron™ or Intel EM64T, or compatible processor at 1 GHz or faster.
- 1 GB of physical memory.
- DVD-ROM drive.
- The disk space requirements defined in 2.1.3, “Procedure for preparing the operating system” on page 17.

As of this writing, Sun servers include:

- High-end servers
  - Sun SPARC Enterprise M8000/M9000
  - Sun Fire E20K/E25K servers
- Mid-range
  - Sun SPARC Enterprise M4000/M5000
  - Sun Fire v490/V890/E2900/E4900/E6900 servers
- Entry-level Servers (up to 4CPU)
  - Sun SPARC Enterprise servers based on UltraSPARC T1 and T2 processors
  - Sun Fire X servers based on AMD Opteron or Intel Xeon® processors
  - Sun Fire V servers based on UltraSPARC IIIi processor
  - Sun Blade Servers based on SPARC, AMD Opteron and Intel Xeon processors

You can get the up-to-date information about these Sun servers at http://sun.com/servers.

IBM servers supported with Solaris 10 are:

- IBM System x
- IBM BladeCenter

For more information, refer to *Implementing Sun Solaris on IBM BladeCenter Servers*, found at:

http://www.redbooks.ibm.com/abstracts/redp4259.html

1.7.1 The Solaris Operating System

Table 1-4 shows the supported operating systems and versions for WebSphere Application Server V6.1.

Table 1-4 Supported Solaris Operating Systems and versions

Platform	Versions
SPARC	► Solaris 9 with the latest patch cluster ► Solaris 10 with the latest patch cluster
x64	► Solaris 10 with the latest patch cluster

WebSphere Application Server is available with 32-bit or 64-bit JVM™ on SPARC systems while 64-bit JVM is available on x64 (AMD64 and Intel 64).

1.7.2 The Java technology for WebSphere on Solaris

WebSphere Application Server is built on Java technology. It is primarily based on the Java 2 Enterprise Edition (J2EE) that also comprises the Java 2 Standard Edition (J2SE) (also known as the Java Development Kit (JDK™)). From a historical perspective, look at Table 1-5 for the Java technology versions that various WebSphere Application Server versions rely on.

Table 1-5 WebSphere Application Server and the Java Platform Versions

WebSphere Application Server	J2EE	J2SE (JDK)	Comment
V6.1	1.4	5	Available with 64-bit JVM.
V6.0.2	1.4	1.4.2	Available for Solaris on x64 systems.
V5.1.1	1.3	1.4.2	WebSphere Application Server v5.x is supported on Solaris 10 in Global Zone only.
V5.1	1.3	1.4.1	
V5.0	1.3	1.3.1	

Note that the JDK update and version in each of these WebSphere Application Server versions can be upgraded when IBM releases updates called “Fix Pack” and “Service Release” (SR).

For more details on the JDK version shipped with WebSphere Application Server and Fix Packs, go to the following Web site:

http://www.ibm.com/support/docview.wss?uid=swg27005002

WebSphere Application Server for the Solaris OS is bundled with Sun's JDK with the exception of some modifications made by IBM. The JDK also provides the Java Runtime Environment (JRE™) that includes the Java Virtual Machine (JVM). The Sun JDK bundled with WebSphere Application Server slightly differs from the standard Sun JDK that can be publicly downloaded from http://java.sun.com. The differences are in the following areas:

- Object Request Broker (ORB) libraries
- XML Processing libraries
- Security Framework

For example, executing the **java -version** command from the installed directory of WebSphere Application Server v6.1 produces the following output:

```
Java(TM) 2 Runtime Environment, Standard Edition (IBM build 1.5.0_06-erdist-2006
0404 20060511)
Java HotSpot(TM) Server VM (build 1.5.0_06-erdist-20060404, mixed mode)
IBM Java ORB build orb50-20060511a (SR2)
XML build XSLT4J Java 2.7.4
XML build IBM JAXP 1.3.5
XML build XML4J 4.4.5
```

It is important to be aware of this information because it is very useful in managing performance of the WebSphere Application Server environment with the appropriate JVM arguments and selecting the proper garbage collectors for an example. We discuss the details about performance management, including Sun JVM, in Chapter 9, "WebSphere Application Server performance tuning on Solaris 10" on page 313.

2

Configuring a Solaris environment for WebSphere Application Server

This chapter describes how to prepare Solaris systems for the installation of IBM WebSphere Application Server Version 6.1. IBM specifies the detailed lists of system requirements for WebSphere Application Server V6.1 at the following Web site:

http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg27007651

In this chapter, the following topics are described:

- How to grant permissions to a non-root installer
- Exporting the location of supported browsers
- How to satisfy disk space requirements
- Setting Solaris kernel parameters to support WebSphere Application Server
- How to verify operating system prerequisites

2.1 Preparing Solaris systems for WebSphere Application Server installation

This section provides key considerations you need to have in order to prepare the Solaris operating environment for WebSphere Application Server. Detailed information about a Solaris installation is available at:

http://docs.sun.com/app/docs/coll/1236.1

2.1.1 System prerequisites for WebSphere Application Server V6.1 on Solaris

You must first check the following IBM Web sites to determine the detailed system requirements for WebSphere Application Server V6.1 for Solaris:

- SPARC system requirements:

  http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg27007662

- x64 System requirements:

  http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg27008314

2.1.2 Solaris OS installation requirements

Attention: The Solaris OS installation process is beyond the scope of this book; however, detailed procedures are available at http://docs.sun.com/app/docs/coll/1236.1

The Solaris OS lets you install a subset of the full operating system, by specifying which Solaris "Software Groups" to install. This makes it possible to reduce the disk space or time needed to do an install. The Solaris software groups are collections of Solaris packages. The current software groups in Solaris 10 today include:

- Entire Distribution plus OEM support
- Entire Distribution
- Developer System Support
- End User System Support
- Core System Support
- Reduced Networking Core System Support

“Reduced Networking Core System Support” contains the minimal number of packages and requires the least amount of disk space while “Entire Distribution Plus OEM Support” contains all available packages for the Solaris OS requiring the most amount of disk space. Detailed descriptions of the Solaris Software Group and recommended space are available in the publication *Solaris 10 Installation Guide: Basic Installations*, found at:

http://docs.sun.com/app/docs/doc/817-0544

Figure 2-1 on page 17 shows an example screen for choosing the necessary “Software Groups” during the Solaris OS installation.

```
Select Solaris Software Group to install:

[ ]  Entire Distribution plus OEM support ....... 5641.00 MB
[X]  Entire Distribution ........................ 5593.00 MB
[ ]  Developer System Support ................... 5468.00 MB
[ ]  End User System Support .................... 4452.00 MB
[ ]  Core System Support .......................  958.00 MB
[ ]  Reduced Networking Core System Support .....  916.00 MB
```

Figure 2-1 Selecting Solaris Software Group during the Solaris OS installation

Each software group includes support for different functions and hardware drivers:

- For an initial Solaris installation, select “Entire Distribution” as the minimum set of Software Group to support WebSphere Application Server V6.1.
- If you have a Solaris system pre-installed with the Software Group less than “Entire Distribution”, upgrade the Solaris installation to this group.

Once the Solaris OS has been properly installed and configured, you can boot up the Solaris system and prepare the operating system to be ready for WebSphere Application Server installation and configuration.

2.1.3 Procedure for preparing the operating system

Preparing the operating system involves such tasks as allocating disk space and installing patches to the operating system. IBM tests WebSphere Application Server products on each operating system platform. Such tests verify whether an operating system change is required for WebSphere Application Server products to run correctly. Without the required changes, WebSphere Application Server products will not run as expected. To run these tests, do the following steps:

1. Log on to the operating system.

   You can log on as root or as a non-root installer. Details on which user to select are discussed in 2.3, “Installation and runtime user of WebSphere Application Server” on page 25.

   Select a umask that allows the owner to read and write to the files, and allows others to access them according to the prevailing system policy. For root, a umask of 022 is recommended. For non-root users, a umask of 002 or 022 can be used, depending on whether the users share the group. To verify the umask setting, issue the following command:

   ```
   umask
   ```

   To set the umask setting to 022, issue the following command:

   ```
   umask 022
   ```

2. *Optional*: Set the location of the supported Web browser.

   Attention: If you are using *Solaris 10 8/07* or later, the Mozilla *Firefox* browser is located in the default location /usr/bin/firefox.

   If your Web browser is not in your command path, you should set the location of it using the command that identifies the actual location of the browser. For example, if the Mozilla package is in the /usr/sfw/bin/mozilla directory, use the following command:

   ```
   BROWSER=/usr/sfw/bin/mozilla
   export BROWSER
   ```

3. *Optional*: In the previous step, the browser requires that you run it in a graphical window. If your Sun server does not have a graphical display terminal, you can either set your display to a remote X display terminal or put the X server capability on the Sun server that your are working with using products like the VNC Server (http://www.realvnc.com).

   To display a remote X display terminal, run the following commands:

   ```
   DISPLAY=your_X_display_terminal:ID
   export DISPLAY
   ```

4. Stop any other WebSphere Application Server-related Java processes on the machine where you are installing the product.
5. Stop any Web server process such as the IBM HTTP Server.
6. Provide adequate disk space:

   The Network Deployment product requires the following disk space:

   – 730 MB for the app_server_root directory before creating profiles

     The installation root directory includes the core product files. This size does not include space for profiles or applications. Profiles require 40 MB of temp space in addition to the sizes shown. Profiles have the following space requirements:

     - 30 MB for the Deployment manager profile

       This size does not include space for Sample applications that you might install. The size also does not include space for applications that you might deploy.

     - 200 MB for an Application Server profile with the Sample applications

       This size does not include space for applications that you might develop and install.

     - 10 MB for an unfederated custom profile

       This size does not include space for applications that you might develop and install. The requirement does include space for the node agent. However, you must federate a custom profile to create an operational managed node.

       After federating a custom profile, the resulting managed node contains a functional node agent only. Use the deployment manager to create server processes on the managed node.

   – 100 MB for the /tmp directory

     The temporary directory is the working directory for the installation program.

     If the /tmp directory does not have enough free space, the installation program stops the installation and displays a message such as `Prerequisite checking has failed. Not enough space.`

– 1030 MB total requirement

  This amount is the total space requirement when installing the product from the disk and when not installing service. Installing profiles requires more space.

The following space is required for the IBM HTTP Server product:

– 300 MB for the web_server_root directory

  The IBM Global Security Kit (GSKit) requires this space.

  On SPARC platforms, the following space is required for the IBM WebSphere Application Server Clients:

  • 150 MB for the app_client_root directory

    The amount of space required to install the application clients is actually less than 150 MB. The amount of space depends on the clients that you install as features.

The following space is required for the Update Installer:

– 200 MB for the /opt/IBM/WebSphere/UpdateInstaller directory

Attention: The installation wizard for each component displays required space on the confirmation panel before you install the product files and selected features. The installation wizard also warns you if you do not have enough space to install the product.

If you plan to migrate applications and the configuration from a previous version, verify that the application objects have enough disk space. As a rough guideline, plan for space equal to 110 percent of the size of the application objects:

- For Version 4.0.x: The size of enterprise archive (EAR) files
- For Version 5.0.x: The size of EAR files

7. IBM recommended parameters for the Solaris OS are shown in Example 2-1. These kernel parameters have been part of WebSphere Application Server since Version 5 for embedded messaging based on the WebSphere MQ product. With Service Integration Bus (SIB) for Java messaging, WebSphere Application Server V6.1 departed from WebSphere Application Server V5. SIB does not require modifications of these recommended parameters in /etc/system except for rlim_fd_cur.

Example 2-1 IBM recommended parameters for Solaris

```
set shmsys:shminfo_shmmax = 4294967295
set shmsys:shminfo_shmseg = 1024
set shmsys:shminfo_shmmni = 1024
set semsys:seminfo_semaem = 16384
set semsys:seminfo_semmni = 1024
set semsys:seminfo_semmap = 1026
set semsys:seminfo_semmns = 16384
set semsys:seminfo_semmsl = 100
set semsys:seminfo_semopm = 100
set semsys:seminfo_semmnu = 2048
set semsys:seminfo_semume = 256
set msgsys:msginfo_msgmap = 1026
set msgsys:msginfo_msgmax = 65535
set rlim_fd_cur=1024
```

Solaris 10 provides a new mechanism to set these parameters through resource control, which we discuss in detail in “Solaris kernel parameters” on page 22. There are two things you should note in Solaris 10:

- Many default values are larger than before and some kernel parameters are obsolete.
- You can manage these parameters through the new resource management framework with /etc/project. By using /etc/project instead of /etc/system, you no longer need to reboot the system.

There are several ways you can set these kernel parameters, as shown in Table 2-1.

Table 2-1 Setting kernel parameters

Effect	Description
Current Session	Set the necessary parameter(s) in your current shell session before you execute your application, such as WebSphere Application Server. This will work for the current session only and will not persist for the next session. The relevant command is: `# prctl -n process.max-file-descriptor -r -v 1024 $$`
System Wide	Set the necessary parameter(s) for system wide in the /etc/project. You may not want to do this as you are increasing the default value(s) for the entire system. The relevant command is: `# projmod -sK 'process.max-file-descriptor=(privileged,1024,deny)' system`
Root User	Set the necessary parameter(s) for the root user in the /etc/project. This is for all processes owned by the root user. The relevant command is: `# projmod -sK 'process.max-file-descriptor=(privileged,1024,deny)' user.root`

Non-Root User	Set the necessary parameter(s) for the WebSphere Application Server user, who runs the server processes, in the /etc/project. This is a best practices because you are changing the default value(s) specifically to the WebSphere Application Server user based upon the IBM recommendation for the WebSphere processes. The relevant commands are:
	`# projadd user.wasuser1`
	`# projmod -sK 'process.max-file-descriptor=(privileged,1024,deny)' user.wasuser1`

If you use Solaris Containers, you can also set the kernel parameters at the zone configuration level, as explained in “Resource control for Solaris Zones” on page 148.

8. Verify that the prerequisites and corequisites are at the required release levels.

   Although the installation wizard checks for prerequisite operating system patches with the prereqChecker application, you should review the prerequisites on the supported hardware and software Web site at `http://www-1.ibm.com/support/docview.wss?rs=&uid=swg27006921` and ensure that your system meets them prior to launching the installation wizard.

   Refer to the documentation for non-IBM prerequisite and corequisite products to learn how to migrate to their supported versions.

The above list of steps results in preparing the operating system for installing the product. After preparing the operating system for installation, you can install the WebSphere Application Server product.

2.1.4 Tunable parameters in Solaris 10

In the Solaris 10 OS release, many kernel tunable parameters, including System V IPC facilities, are either automatically configured or can be controlled by resource controls. Facilities that can be shared among processes are memory, message queues, and semaphores.

Resource controls allow IPC settings to be made on a per-project (for example, accounting group) or per user basis on the local system or in a name service environment. In previous Solaris releases, IPC facilities were controlled by kernel tunable parameters. You had to modify the /etc/system file and reboot the system to change the default values for these facilities. For further information, refer to http://docs.sun.com/app/docs/doc/806-7009.

Because the IPC facilities are now controlled by resource controls, their configurations can be modified while the system is running. Many applications that previously required system tuning to function properly may now run without any tuning because of increased default values and the automatic allocation of resources.

Most new values of the default kernel parameters are typically sufficient and a good baseline to support WebSphere Application Server. In Solaris 10, many kernel values have been increased to accommodate an application's increased demand for more system resources.

For installation and verification tests, you can verify the values of these tunable parameters (installed as "root" user) and achieve successful installation and execution of WebSphere Application Server.

Prior to Solaris 10, the **sysdef** command was used to provide the IPC Module related settings:

```
bash-# sysdef -i
```

But, when this command is executed on Solaris 10 system, the output shows that these modules do not have system-wide limits. Example 2-2 shows a portion of the output from **sysdef -i** command on Solaris 10.

Example 2-2 Output from the sysdef -i command

```
*
* Process Resource Limit Tunables (Current:Maximum)
*
0x0000000000000100:0x0000000000010000 file descriptors
*
* Streams Tunables
*
9 maximum number of pushes allowed (NSTRPUSH)
65536 maximum stream message size (STRMSGSZ)
1024 max size of ctl part of message (STRCTLSZ)
*
* IPC Messages
*
* The IPC Messages module no longer has system-wide limits.
* Please see the "Solaris Tunable Parameters Reference Manual" for
* information about how the old limits map to resource controls and
* the prctl(1) and getrctl(2) manual pages for information about
* observing the new limits.
**
* IPC Semaphores
*
* The IPC Semaphores module no longer has system-wide limits.
* Please see the "Solaris Tunable Parameters Reference Manual" for
* information about how the old limits map to resource controls and
* the prctl(1) and getrctl(2) manual pages for information about
* observing the new limits.
**
```

```
* IPC Shared Memory
*
* The IPC Shared Memory module no longer has system-wide limits.
* Please see the "Solaris Tunable Parameters Reference Manual" for
* information about how the old limits map to resource controls and
* the prctl(1) and getrctl(2) manual pages for information about
* observing the new limits.
```

To obtain the IPC and other settings for the current shell environment where the WebSphere Application Server is to be installed, use the following command:

```
bash-3.00# prctl $$
```

The **prctl** command with the $$ option lists all the System V related IPC and file descriptor settings for the current shell that will be applied to any process started within that shell. To make any changes, change the settings, close the current shell, and log back in to get the new settings into effect.

You can use the **id** command to get the current project id of the root user:

```
bash-# id -p
uid=0(root) gid=0(root) projid=1(user.root)
```

Solaris kernel parameters

If you must modify the Solaris kernel parameters for other WebSphere related components that are to be co-located with WebSphere Application Server, such as WebSphere MQ, we provide you with additional information in this section and the examples here are for the root user.

Table 2-2 lists the comparison between the IBM recommended parameters and the new tunable kernel parameters, such as SYS V IPC, in the Solaris 10 for WebSphere Application Server installation. It shows the differences between the original /etc/system file and the Solaris 10 /etc/project file. It also lists the new default values and obsoleted parameters.

Table 2-2 Solaris 10 tunable parameters for WebSphere Application Server

IBM recommended /etc/system settings for WebSphere Application Server	New resource control parameters in Solaris 10	New default value
set shmsys:shminfo_shmmax = 4294967295	project.max-shm-memory	1/4 of physical memory
set shmsys:shminfo_shmseg = 1024	Obsolete	
set shmsys:shminfo_shmmni = 1024	project.max-shm-ids	128
set semsys:seminfo_semaem = 16384	Obsolete	
set semsys:seminfo_semmni = 1024	project.max-sem-ids	128
set semsys:seminfo_semmap = 1026	Obsolete	
set semsys:seminfo_semmns = 16384	Obsolete	
set semsys:seminfo_semmsl = 100	process.max-sem-nsems	512
set semsys:seminfo_semopm = 100	process.max-sem-ops	512
set semsys:seminfo_semmnu = 2048	Obsolete	
set semsys:seminfo_semume = 256	Obsolete	

IBM recommended /etc/system settings for WebSphere Application Server	New resource control parameters in Solaris 10	New default value
set msgsys:msginfo_msgmap = 1026	Obsolete	
set msgsys:msginfo_msgmax = 65535	Obsolete	
set rlim_fd_cur=1024	process.max-file-descriptor	256

As you go through the IBM recommended /etc/system settings in Solaris 10, ignore the obsoleted parameters; thus, you only need to modify the following parameters:

```
project.max-shm-memory
project.max-shm-ids
project.max-sem-ids
process.max-sem-nsems
process.max-sem-ops
process.max-file-descriptor
```

If a parameter's current value is less than the recommended threshold, update the current value with the recommended value. By using the **projmod** command, the settings are stored in the /etc/project file. The commands to modify the resource control parameters for the root user are shown in Example 2-3.

Example 2-3 Commands to modify the resource control parameters

```
bash-3.00# projmod -s -K 'project.max-shm-memory=(privileged,4gb,deny)' user.root
bash-3.00# projmod -s -K 'project.max-shm-ids=(privileged,1024,deny)' user.root
bash-3.00# projmod -s -K 'project.max-sem-ids=(privileged,1024,deny)' user.root
bash-3.00# projmod -s -K 'process.max-sem-nsems=(privileged,512,deny)' user.root
bash-3.00# projmod -s -K 'process.max-sem-ops=(privileged,512,deny)' user.root
bash-3.00# projmod -s -K 'process.max-file-descriptor=(privileged,1024,deny)' \
              user.root
```

You can now examine the new tunable parameter settings in the /etc/project file. Example 2-4 shows the new parameter settings for the root user. These project setting modifications should be performed on each zone where WebSphere is being installed. See Chapter 6 5.1.3, “Solaris Containers” on page 118 for a discussion of Solaris zones.

Example 2-4 Contents of the /etc/project file

```
bash-3.00# cat /etc/project
system:0::::
user.root:1::::
   process.max-file-descriptor=(privileged,1024,deny);
   process.max-sem-ops=(privileged,512,deny);
   process.max-sem-nsems=(privileged,512,deny);
   project.max-sem-ids=(privileged,1024,deny);
   project.max-shm-ids=(privileged,1024,deny);
   project.max-shm-memory=(privileged,4294967296,deny)
noproject:2::::
default:3::::
group.staff:10::::
```

You can also make these changes using the **prctl** command, but the settings *will not* persist through a system reboot:

```
bash-3.00# prctl -n project.max-shm-memory -r -v 4gb -i project 1
```

To make the change effective, you must log out and then log in, or simply you can start a new shell window that will have the new settings in effect. Use the **prctl** command to examine the new settings:

```
bash-3.00# prctl $$
```

In Solaris 10, you can enable logging using **rctladm** so the system will notify you when you are running out of these resources. For example, if a user wants to be notified by the system when it is running out of process.max-file-descriptor, then the user can issue the following command from the shell prompt:

```
bash-3.00# rctladm -e syslog process.max-file-descriptor
```

Modify process.max-file-descriptor in the /etc/rctladm.conf file from `process.max-file-descriptor=none` to `process.max-file-descriptor=syslog=notice`.

If the system happens to run out of file descriptors, it will be reported in the /var/adm/messages file, as shown in Example 2-5.

Example 2-5 Resource notifications in the /var/adm/messages file

```
Sep 27 13:57:04 host-02 genunix: [ID 883052 kern.notice] privileged rctl
process.max-file-descriptor (value 5) exceeded by process 16797
Sep 30 17:43:19 host-02 genunix: [ID 883052 kern.notice] basic rctl
process.max-file-descriptor (value 256) exceeded by process 10535
```

This can also be achieved by directly updating the /etc/rctladm.conf file. To get notification about all the resources, you can change all the lines shown in Example 2-5 for the file descriptor and then examine the /var/adm/messages file for notifications.

You can set or get limitations on the system resources available to the current shell and its descendents (for example, file-descriptors limits) using the **ulimit** command.

To retrieve the current value, run the following command:

```
bash-3.00# ulimit -a
```

To set a new value, run the following command:

```
bash-3.00# ulimit -n <new value>
```

2.2 Network configuration for WebSphere Application Server communication

Many of the Sun servers today have more than one network device available. It is important for the system administrator to decide ahead of time whether to use shared or dedicated network interfaces for WebSphere Application Server nodes being deployed. Each network device has its own one or more IP addresses, netmask, and other attributes. Details can be found in the *Solaris 10 System Administration Guide: IP Services*, found at:

http://docs.sun.com/app/docs/doc/816-4554

Configuring network interfaces

For WebSphere Application Server to be able to utilize a network interface, it must be plumbed and brought up properly by the Solaris system administrator. Assume you want to bring up a network interface called e1000g1, which you can do by running the following command:

```
# ifconfig -a
# ifconfig e1000g1 up
# ifconfig e1000g1 ip_address net_mask plumb
```

Network interfaces can also be multiplexed as multiple logical devices by running the following command:

```
# ifconfig e1000g1:1 ip_address net_mask plumb
```

Check the status of sockets (ports)

You can query the active ports on the system to ensure that there are no port conflicts after WebSphere Application Server is installed by running:

```
netstat -a
```

TCP tuning parameters

TCP driver parameters are used for network communications and can be dynamically set with the commands shown in Example 2-6.

Example 2-6 Commands to set TCP parameters

```
# ndd -set /dev/tcp tcp_conn_req_max_q 16384
# ndd -set /dev/tcp tcp_conn_req_max_q0 16384
# ndd -set /dev/tcp tcp_max_buf 4194304
# ndd -set /dev/tcp tcp_cwnd_max 2097152
# ndd -set /dev/tcp tcp_recv_hiwat 400000
# ndd -set /dev/tcp tcp_xmit_hiwat 400000
```

2.3 Installation and runtime user of WebSphere Application Server

WebSphere Application Server V6.1 can be installed by a root or non-root user. The installation user owns the files and directories of WebSphere Application Server once the installation is completed successfully. We describe the differences between the two users in the next two sections. Then, we describe the runtime user as a related topic.

2.3.1 Installation with the root privileges

When WebSphere Application Server V6.1 is installed by the root user, two IBM WebSphere Application Server's package information entries are added to the Solaris package registry. The package names are WSBAA61 and WSBAA61LI. You can query to see if WebSphere Application Server V6.1 has been installed by root with the **pkginfo** command, as shown in Example 2-7. For more information about Solaris software package and management, see http://docs.sun.com/app/docs/doc/817-1985.

Example 2-7 Solaris package query for WebSphere Application Server V6.1

```
bash# pkginfo -l WSBAA61 WSBAA61LI
   PKGINST:  WSBAA61
      NAME:  IBM WebSphere Application Server
  CATEGORY:  application
      ARCH:  sparc
   VERSION:  6.1.0.0.DSP=6.1.0.0
   BASEDIR:  /opt/IBM/WebSphere/AppServer
    VENDOR:  IBM
      DESC:  IBM WebSphere Application Server V6.1
    STATUS:  completely installed

   PKGINST:  WSBAA61LI
      NAME:  LAP Component
  CATEGORY:  application
      ARCH:  sparc
   VERSION:  6.1.0.0.DSP=6.1.0.0
   BASEDIR:  /opt/IBM/WebSphere/AppServer
    VENDOR:
      DESC:  LAP
    STATUS:  completely installed
```

2.3.2 Installation with non-root privileges

WebSphere Application Server V6.1 can be installed by a non-root user. The WebSphere Application Server package information described in 2.3.1, "Installation with the root privileges" on page 26 will not be registered to the Solaris package registry. It is possible to have port conflicts when other WebSphere Application Server users may already be using the ports you are attempting to use.

Before you can begin installation, preparation tasks must be performed on the underlying Solaris system by creating the necessary *user id*, *group id*, and its *home directory* for successful installation. Example 2-8 shows how to create the home directory, add a group and user ID, project entry, and file descriptor limit per IBM recommendation for the WebSphere Application Server user.

Example 2-8 Preparing a non-root user for WebSphere Application Server installation

```
# mkdir -p /ex1/home/wasuser1
# groupadd wasgrp1
# useradd -c "WAS User" -d "/ex1/home/wasuser1" \
      -g wasgrp1 -s /usr/bin/bash wasuser1
# chown -R wasuser1:wasgrp1 /ex1/home/wasuser1
# passwd wasuser1
```

There are other limitations associated with non-root installers that you should be aware of. For details, read the IBM WebSphere Application Server Information Center document found at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/rins_nonroot.html

Important: For the non-root user to be able to perform successful profile creation in the root installed environment, you must first grant write permission of files and directories to this user. Refer to:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/tpro_nonrootpro.html

2.3.3 The runtime user of WebSphere Application Server processes

The runtime user is the user that actually invokes the WebSphere Application Server commands to configure, manage, and execute the WebSphere Application Server runtime environment. This user could be the root user or a non-root user. You must first properly set up the user, as shown in either Example 2-9 or Example 2-10 for the respective user.

Example 2-9 Preparing a non-root user for WebSphere Application Server execution

```
# projadd user.wasuser1
# projmod -sK 'process.max-file-descriptor=(privileged,1024,deny)' user.wasuser1
```

Example 2-10 Preparing the root user for WebSphere Application Server execution

```
# projmod -sK 'process.max-file-descriptor=(privileged,1024,deny)' user.root
```

The user also must have full read and write privileges in the WebSphere Application Server profile and log directories. The detailed instructions of how to limit the runtime privileges is explained in 5.6, “Process Rights Management” on page 157.

2.4 Default product locations when the root user installs the product

The root user is capable of registering shared products and installing into system-owned directories. The following default directories shown in Table 2-3 are system-owned directories. These file paths are default locations. You can install the product and other components in any directory where you have write access. You can create profiles in any valid directory where you have write access. Multiple installations of WebSphere Application Server Network Deployment products or components, of course, require multiple locations.

Table 2-3 Default product file system locations for root installer

Variable	Description	File system location
app_server_root	The install_root for WebSphere Application Server	/opt/IBM/WebSphere/AppServer
profile_root	The install_root for WebSphere Application Server profiles	/opt/IBM/WebSphere/AppServer/profiles/profile_name

Variable	Description	File system location
plugins_root	The install_root for the IHS plug-in	/opt/IBM/HTTPServer/Plugins
web_server_root	The install_root for the IHS	/opt/IBM/HTTPServer
gskit_root	The install_root for IBM Global Security Kit	/opt/ibm/gsk7
app_client_root	The install_root for WebSphere Application client	/opt/IBM/WebSphere/AppClient (J2EE Application client only)
updi_root	The install_root for Update Installer for WebSphere Software	/opt/IBM/WebSphere/UpdateInstaller
cip_app_server_root	Root directories for a customized installation package (CIP)	/opt/IBM/WebSphere/AppServer/cip/cip_uid

Note: The cip_uid variable is the CIP unique ID generated during creation of the build definition file. You can override the generated value in the Build definition wizard. Use a unique value to allow multiple CIPs to install on the system.

2.5 Default product locations when a non-root user installs the product

The non-root user is not capable of registering shared products and installing into system-owned directories. The following default directories shown in Table 2-4 are user-owned directories in the home directory of the non-root installer as opposed to being globally shared resources that are available to all users.

Table 2-4 Default product file system locations for non-root installer

Variable	Description	File system location
app_server_root	The install_root for WebSphere Application Server	/*user_home*/IBM/WebSphere/AppServer
profile_root	The install_root for WebSphere Application Server profiles	/*user_home*/IBM/WebSphere/AppServer/profiles/*profile_name*
plugins_root	The install_root for the IHS plug-in	/*user_home*/IBM/HTTPServer/Plugins
web_server_root	The install_root for the IHS	/*user_home*/IBM/HTTPServer
gskit_root	The install_root for IBM Global Security Kit	/*user_home*/ibm/gsk7
app_client_root	The install_root for WebSphere Application client	/*user_home*/IBM/WebSphere/AppClient (J2EE Application client only)
updi_root	The install_root for Update Installer for WebSphere Software	/*user_home*/IBM/WebSphere/UpdateInstaller

Variable	Description	File system location
cip_app_server_root	Root directories for a customized installation package (CIP)	/*user_home*/IBM/WebSphere/AppServer/cip/cip_uid

Note: The cip_uid variable is the CIP unique ID generated during creation of the build definition file. You can override the generated value in the Build definition wizard. Use a unique value to allow multiple CIPs to install on the system.

3

Installation of WebSphere Application Server for Solaris 10

In this chapter, you will be guided through the procedure to install the WebSphere Application Server basic installation on Solaris 10. You will be provided the guidance to prepare, install, and verify the installation of your WebSphere Application Server basic installation.

3.1 Installing a stand-alone WebSphere Application Server

In this section, we install WebSphere Application Server with a stand-alone application server profile in various ways. We demonstrate two approaches to performing the installation of WebSphere Application Server V6.1 on Solaris 10. You will be guided through the default installation procedure.

Important: To install the product successfully, we recommend that system configurations on Solaris be performed according to 2.1.3, “Procedure for preparing the operating system” on page 17. Before you proceed, you should verify that your Solaris system is set up correctly.

The topology and installation information is summarized in Table 3-1.

Table 3-1 Installation information

Installation option	Value
install sample applications	no
was_root	/opt/IBM/WebSphere/AppServer
server profile	stand-alone (Application Server)
enable administrative security	yes

For more information about the planning and designing of a WebSphere Application Server V6 installation, go to:

http://www.redbooks.ibm.com/abstracts/redp4305.html?Open

3.1.1 Graphical user interface installation

Do the following steps:

1. Log in to your Solaris system as root.
2. If you have not set the operating system parameters (umask or BROWSER, for example), refer to 2.1.3, “Procedure for preparing the operating system” on page 17 for further information.
3. Go to the directory where you unpacked the product’s source files or mount your CD-ROM or DVD-ROM drive.
4. Start the installation.
   - You can start the installation of the product with the launchpad script:

     ```
     bash#./launchpad.sh
     ```

     and choosing **WebSphere Application Server Network Deployment installation** from the left side of the launchpad’s welcome window, as shown in Figure 3-1 on page 33.

Tip: If you have problems starting launchpad, verify that you have a browser installed into your system and you have exported it. Locate your browser binaries and export it and start over. For example:

```
bash# export BROWSER=/usr/sfw/bin/mozilla
```

Figure 3-1 Launchpad: Welcome window

– Alternatively, you can start the installation wizard directly with the **`install`** command:

```
bash#./WAS/install
```

Tip: If you have problems starting the installation wizard due to Java Runtime Environment (JRE) error, you can force the installation wizard to use its own JRE by using the following command:

```
./WAS/install -is:javahome /PATH_TO_MEDIA/WAS/java/jre
```

5. Click **Launch the Installation wizard for the WebSphere Application Server Network Deployment** link in the launchpad application (Figure 3-2) to start the installation wizard.

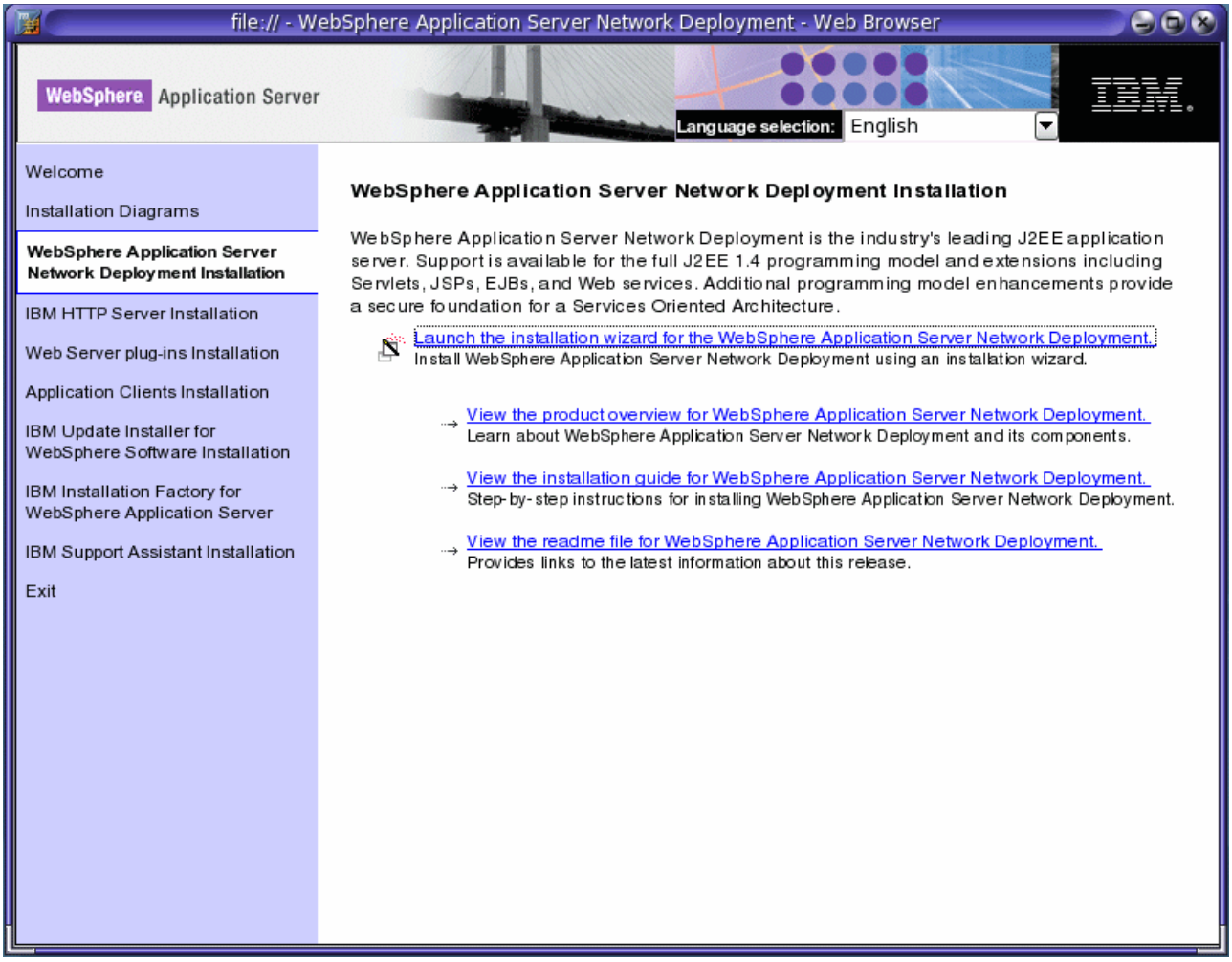


Figure 3-2 Launchpad: Start installation of WebSphere Application Server

6. The Installation wizard starts and you should see the Welcome window of the installation wizard, as shown in Figure 3-3 on page 35.

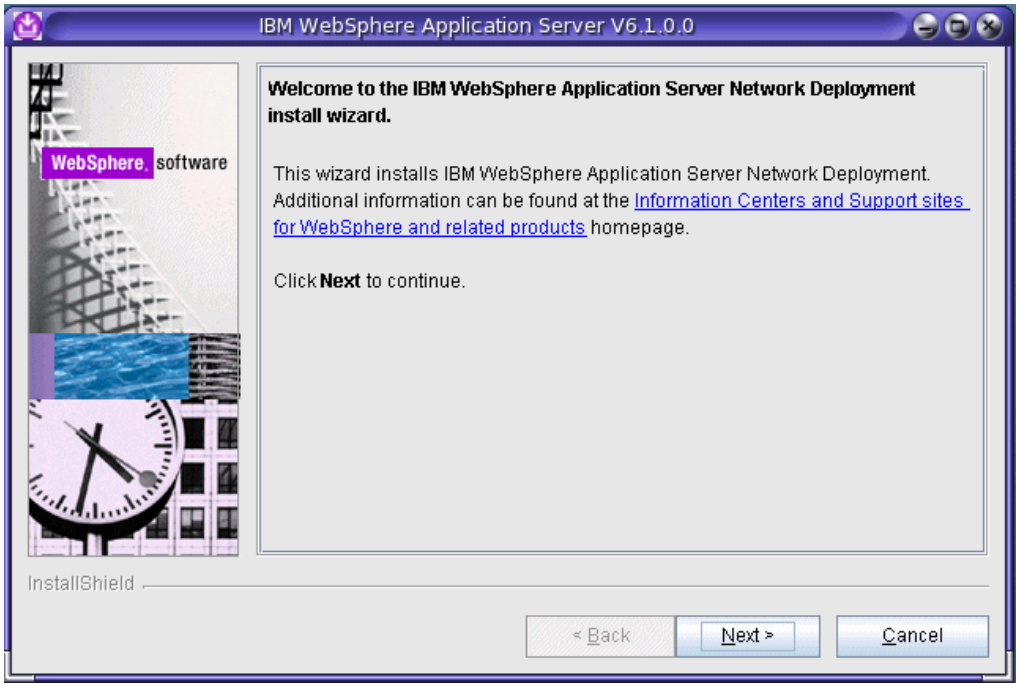


Figure 3-3 Installation wizard: Welcome panel

7. Click **Next**, and the window shown in Figure 3-4 should appear.

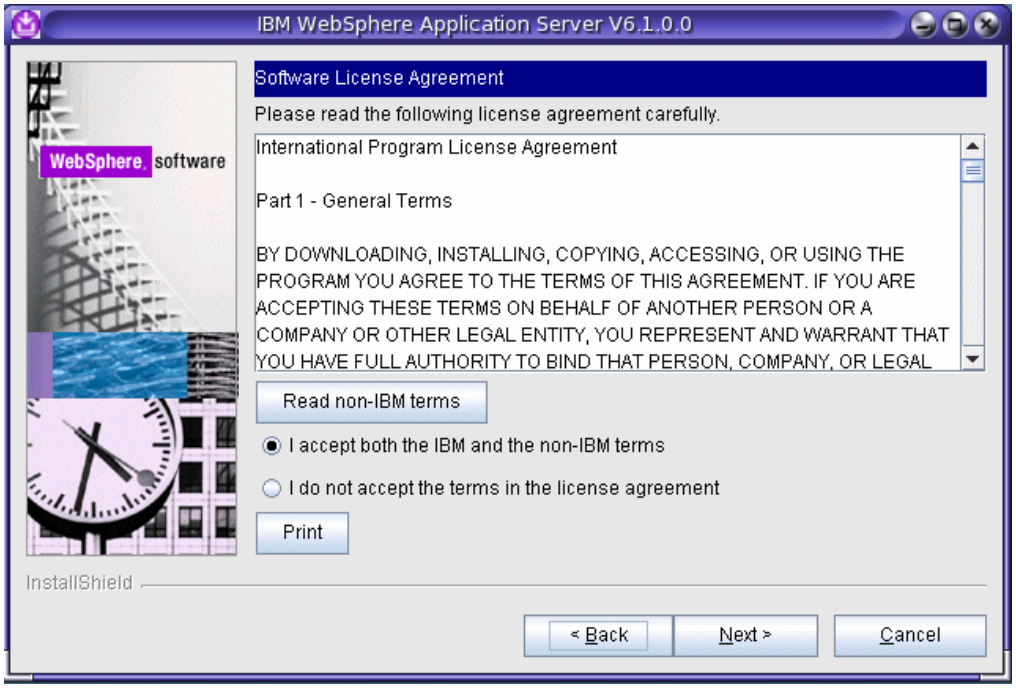


Figure 3-4 Installation wizard: License agreement panel

8. Read and accept the IBM and third-party license agreement by selecting the **I accept both the IBM and the non-IBM terms** radio button shown in Figure 3-4 and then click **Next** to continue the installation.

After accepting the licensing terms, the installation verifies your system. The system must meet the prerequisites for installation to continue. If you receive an error message indicating your system does not meet the prerequisites, cancel the installation, make the required corrections, and restart the installation.

9. After confirming that the prerequisites have been met, click **Next** (Figure 3-5).

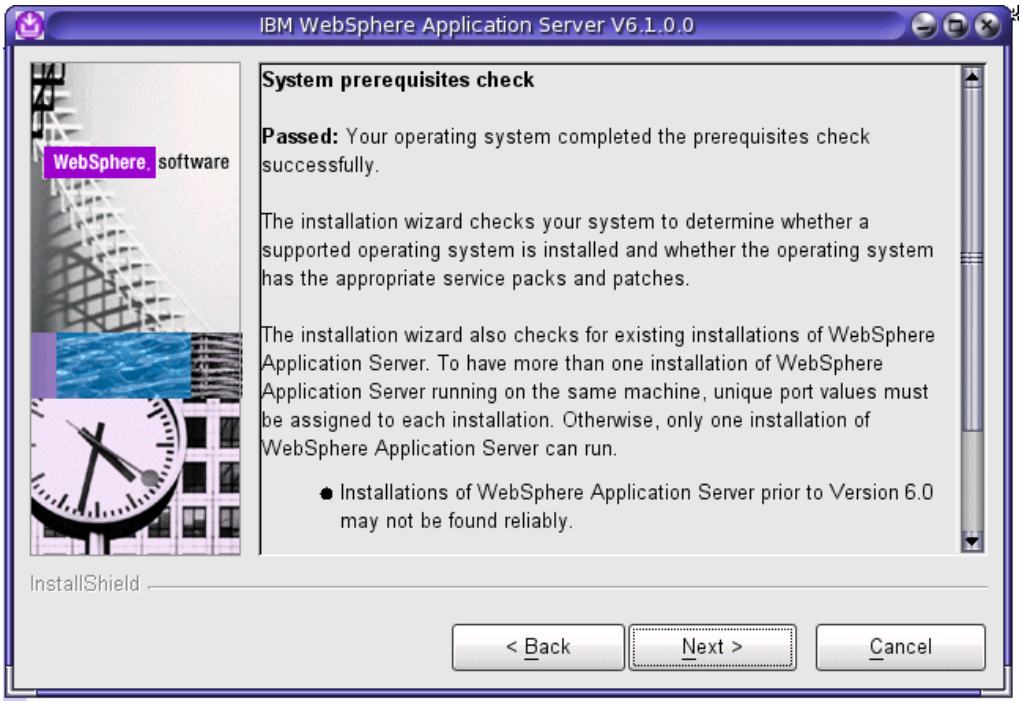


Figure 3-5 Installation wizard: System Prerequisites check

The installation wizard checks for a previous installation at the same product level. The wizard looks for an existing Version 6.1 installation. If a previous installation is detected, the wizard shows an Existing installation panel, where you can:

- Add features to the existing installation.
- Perform a new installation to another directory.
- Perform an upgrade of a trial installation or Express installation to the full product.

The following steps assume that you do not have an existing installation that needs to be upgraded or updated with additional features. For more information about upgrading or migration issues, refer to the *WebSphere Application Server V6 Migration Guide*. SG24-6369 and the WebSphere Application Server Information Center.

10. Click **Next** without the **Install the sample applications** check box selected (Figure 3-6 on page 37).

Tip: For better performance, we recommend that you do not install Sample applications in your production environment. By omitting the Samples, you can improve application server startup time by 60 percent and save 15 percent of disk space on each application server installation. You can also save up to 30 percent of processing footprint (based on a maximum heap size of 256 MB).

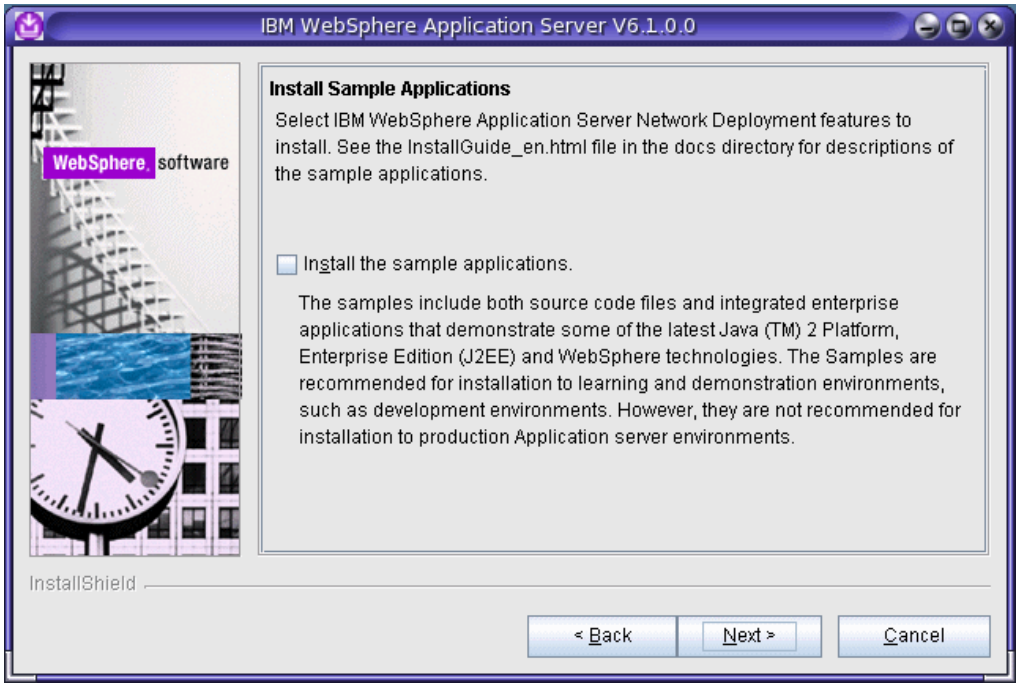


Figure 3-6 Installation wizard: Installation of sample applications

11. Specify the destination of the installation root directory to the field using the information in Table 3-1 on page 32 (Figure 3-7 on page 38). Click **Next**.

Attention:

- Leaving the destination of the installation root field empty prevents you from continuing.
- Do not use symbolic links as the destination directory because they are not supported.
- Spaces are not supported in the name of the destination installation directory in the Sun Solaris environment.

The installer checks for the required space before calling the installation wizard. If you do not have enough disk space, cancel the installation program and acquire more disk space. The disk space required for the base WebSphere Application Server product is:

- 930 MB for the destination of installation root directory
- 100 MB for the /tmp directory

If both of these directories are in the same file system, the total amount of free space needed in that file system is 1030 MB. For full disk space requirements, refer to 2.1.3, “Procedure for preparing the operating system” on page 17 for more information.

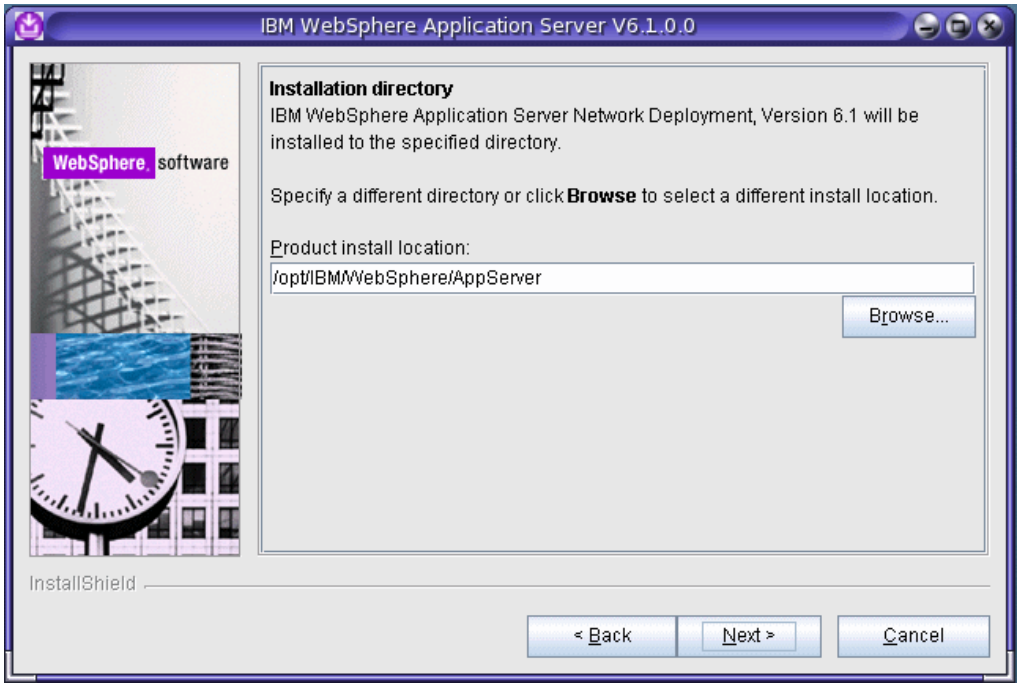


Figure 3-7 Installation wizard: Installation directory panel

12. To create a stand-alone application server profile, choose the **Application Server** environment, as shown in Figure 3-8 and click **Next**.

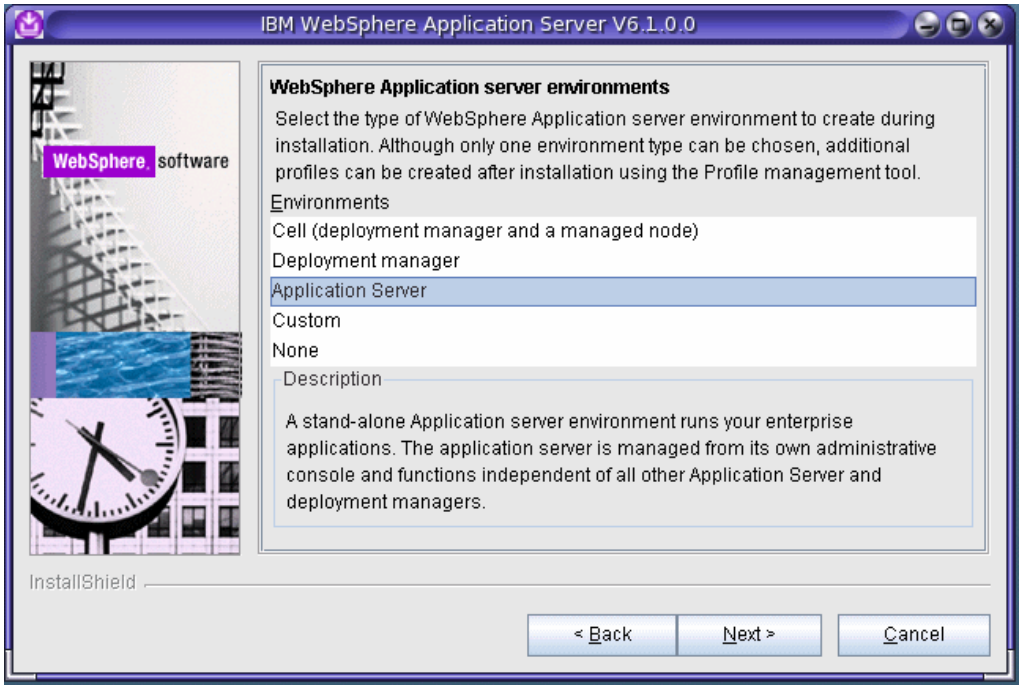


Figure 3-8 Installation wizard: Application server environment selection

13. Optional: If you want to enable security for administration console user, select the **Enable administrative security** radio button (Figure 3-9). Fill in the following fields:
    - Username
    - Password
    - Confirm Password

    Remember that the Password and Confirm Password fields must contain the exact same strings. We recommend that you do not use special characters in passwords because of the possibility of different encoding locales.

Attention: If you enable security at this point, you will get a file based user repository. If you are planning to use another type of user repository, you should enable administrative security after the installation or profile creation is complete. See 8.1.1, “Enabling security” on page 269 for more information about enabling administrative security.

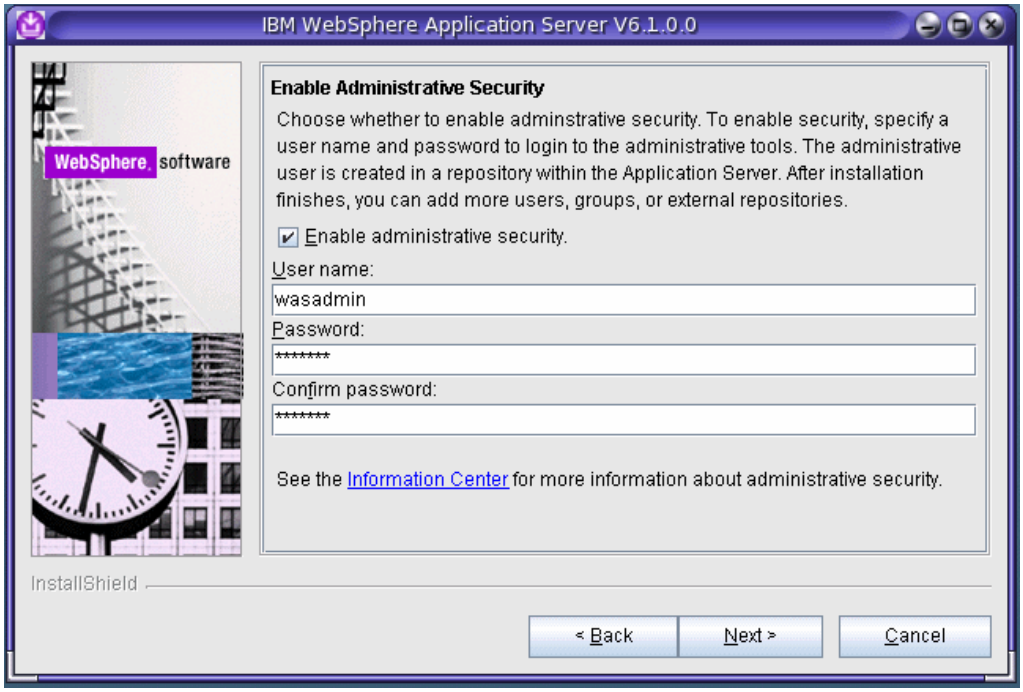


Figure 3-9 Installation wizard: Administrative security window

Tip: In environments where you plan to have multiple stand-alone application servers, the security policy of each application server profile is independent of the others. Changes to the security policies in one application server are *not* synchronized with other profiles.

Remember the user name and password you type here. You cannot log onto the application server’s administrative console without it. You cannot stop the Application Server from the command line or use the Application Server at all unless you have the user name and password.

14. Review the summary in Figure 3-10. If you have to change something, click **Back**; otherwise, click **Next**.

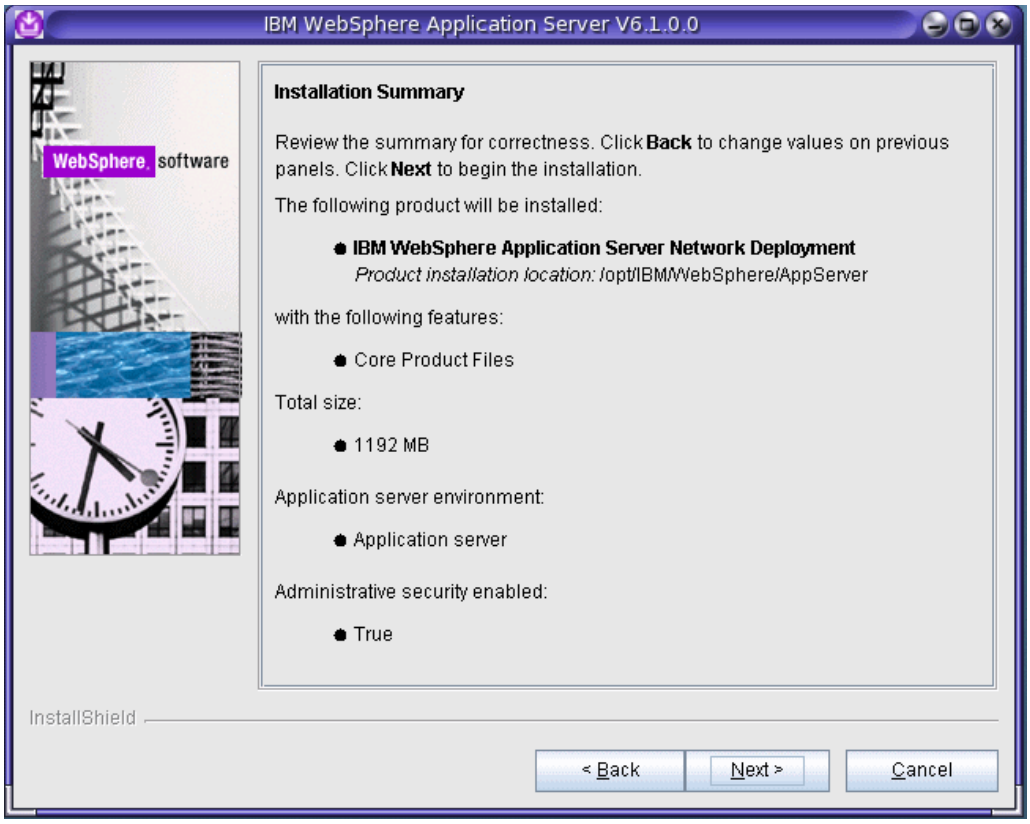


Figure 3-10 Installation wizard: Installation summary

The Installation wizard creates the uninstaller program and then displays a progress window that shows which components are being installed. This may take a little time. At the end of the installation, the wizard displays the Installation completion window (Figure 3-11 on page 41).

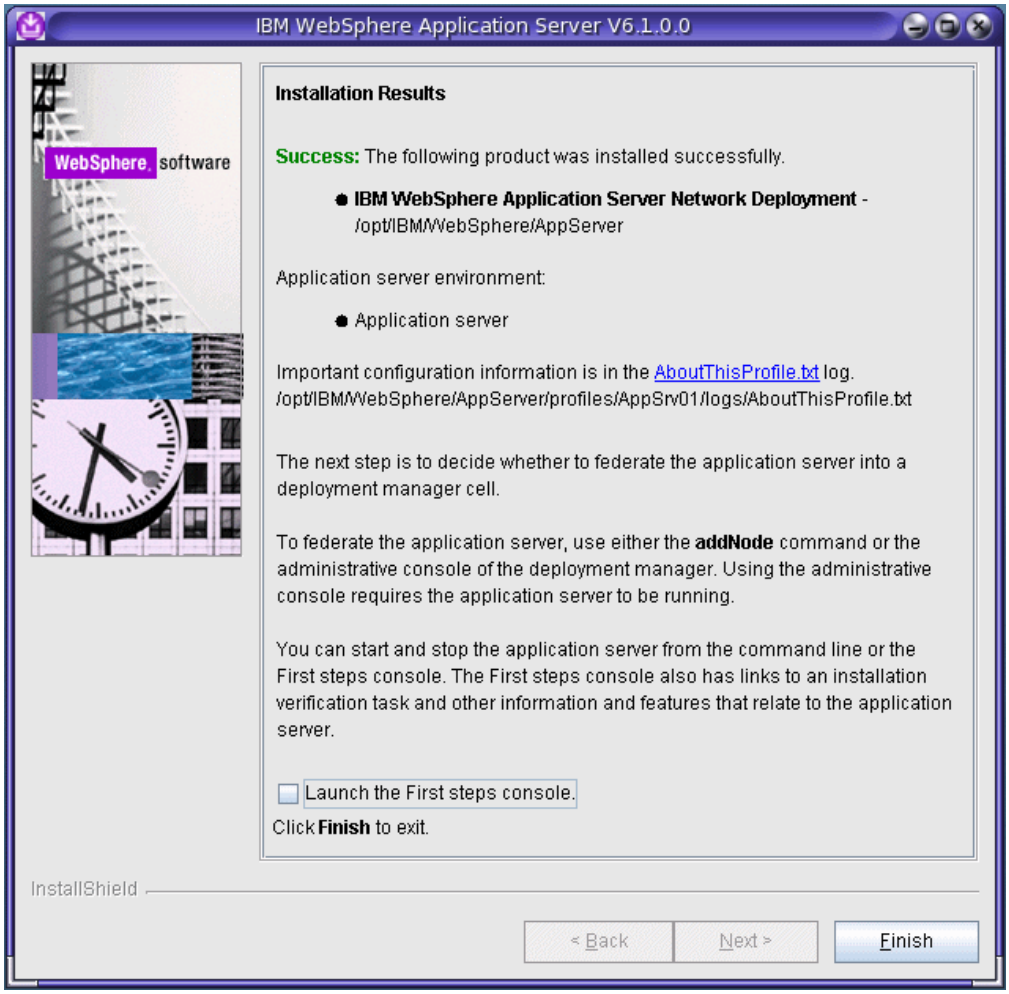


Figure 3-11 Installation wizard: Installation completed

15. Click the **Exit** link on the launchpad's left hand side as shown in Figure 3-12 on page 42 to exit the launchpad application.

Optional: Install the application clients.

You do not need to install the Application Client unless an application that you are deploying was designed to run as a client application.

For details, refer to the WebSphere Application Server Information Center at `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp`, and look for scripting: Installing Application Client for WebSphere Application Server.

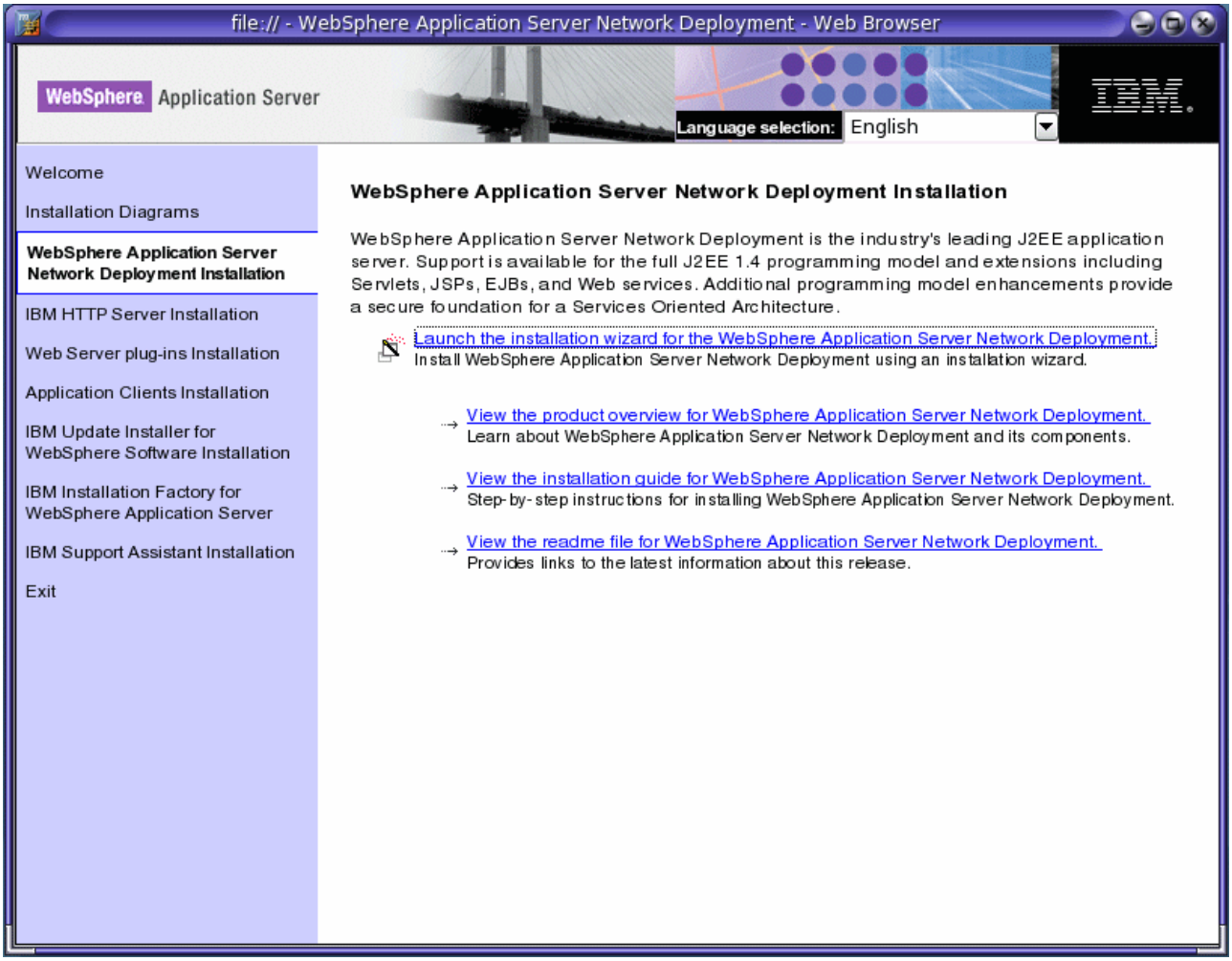


Figure 3-12 Launchpad: Exit launchpad application

16. Verify the installation by examining the completion window and the log.txt located in the <was_root>/logs/install directory.

    You should see the `INSTCONFSUCCESS` entry at the end of the log.txt file after a successful installation.

    You can also select the **Launch the First steps console** check box in the installation wizard's completion window shown Figure 3-11 on page 41 before clicking the **Finish** button. For additional details about using the First steps console, refer to 10.3.1, "Installation problem determination" on page 418.

If problems occur, consult the following applicable logs in Table 3-2 on page 43. You can also refer to *WebSphere Application Server V6.1 installation problem determination*, REDP-4305, found at:

`http://www.redbooks.ibm.com/abstracts/redp4305.html?Open`

Table 3-2 Installation and profile creation logs for WebSphere Application Server

Log	Content	Indicators
app_server_root/logs/install/log.txt	Logs all installation events.	Return codes: 0: Success. 1: Failure. 2: Partial Success.
app_server_root/logs/manageprofiles/*profile_name*_create.log	► Traces all events that occur during the creation of the named profile. ► Created when using the Profile Management tool or the **manageprofiles** command.	INSTCONFFAIL: Total profile creation failure. INSTCONFSUCCESS: Successful profile creation. INSTCONFPARTIALSUCCESS: Profile creation errors occurred but the profile is still functional. Additional information identifies the errors.
app_server_root/logs/manageprofiles/*profile_name*_delete.log	► Traces all events that occur during the creation of the named profile. ► Created when using the Profile Management tool or the **manageprofiles** command.	INSTCONFFAIL Total profile creation failure. INSTCONFSUCCESS Successful profile creation. INSTCONFPARTIALSUCCESS Profile creation errors occurred but the profile is still functional. Additional information identifies the errors.
app_server_root/logs/install/installconfig.log.gz	► Logs the activities of ANT configuration scripts that run at the end of the installation procedure. ► Gzip file.	Configuration action failed: Unsuccessful ANT script configuration. Configuration action succeeded: Successful ANT script configuration.

This procedure results in the installation wizard installing WebSphere Application Server into the installation root directory. The installation wizard creates a profile named AppSrv01 that provides the runtime environment for the server1 application server. Further configuration is not necessary at this time. However, you can create additional stand-alone application servers with the Profile Management tool. For more information, refer to 4.2.2, “Creating profiles with Profile Management Tool” on page 60.

3.1.2 Silent installation

Installing WebSphere Application Server using silent installation refers to using installation options without user interaction. A silent installation uses the installation wizard to install the product in silent mode, without the graphical user interface.

Instead of displaying an installation wizard interface, the silent installation causes the installation program to read all of your responses from a file that you provide. To specify non-default options during the installation, you must use a customized response file. To install silently, you must accept the license agreement in the agreement option.

To perform the silent installation, do the following steps:

1. Log on to your Solaris system as root.
2. Go to the directory where you unpacked the products source files.

3. Copy the responsefile.nd.txt as, for example, responsefile.txt:

   ```
   bash# cp responsefile.nd.txt responsefile.txt
   ```

4. Open responsefile.txt with your favorite text editor and edit:

   a. Set property `-OPT silentInstallLicenseAcceptance` to “true”.

   b. Uncomment `-OPT installLocation=”/opt/IBM/WebSphere/AppServer”` and set it to the correct value if you are installing to a different location.

   c. Comment out the Windows install Location option so you do not have a duplicate entry for installLocation option. If you have a duplicate entry on this option, the installation will fail.

   d. Change option `-OPT profileType` to “standAlone”.

   e. If you enable administrative security:

      i. Set a value for the option `-OPT PROF_adminUserName`.

      ii. Set a value for the option `-OPT PROF_adminPassword`.

5. Save your response file.

6. Issue the command **`./install -options “responsefile.txt” -silent`** and press Enter.

   The installation wizard creates an uninstallation program and then installs all product binaries. When installation is complete, the program returns to the command-line prompt, but you will not see any messages indicating either a successful or unsuccessful installation.

7. When the installation is complete, you should check the log.txt file in the <was_root>/logs/install directory and look for the `INSTCONFSUCCESS` indicator at the end of the log file.

If problems occur, consult the applicable logs in Table 3-2 on page 43. Correct the problem and restart the installation from Step 6.

Example 3-1 shows a sample response file.

Example 3-1 Example response file in short

```
-OPT silentInstallLicenseAcceptance="true"
-OPT allowNonRootSilentInstall="true"
-OPT installType="installNew"
-OPT feature="noFeature"
-OPT installLocation="/opt/RB/RB-WAS-Install"
-OPT profileType="standAlone"
-OPT PROF_enableAdminSecurity="false"

# Stand-Alone Profile info
-OPT PROF_profileName=RBProfile
-OPT PROF_profilePath="/opt/RB/RB-Profile"
-OPT PROF_isDefault="true"
-OPT PROF_hostName=app4
-OPT PROF_nodeName=app4Node
-OPT PROF_cellName=app4Cell
-OPT PROF_startingPort=12000
-OPT PROF_nodeStartingPort=13000
-OPT PROF_validatePorts="true"
```

Tip: For more information about silent installation profile creation, refer to “Silent installation considerations” on page 449.

3.1.3 Custom Installation Packages

The detailed procedures for creating Custom Installation Packages (CIP) are described in “Custom Installation Packages (CIPs)” on page 95.

3.2 Installation problem determination

Detailed information about installation problem determination is described in 10.3.1, “Installation problem determination” on page 418.

3.3 Uninstallation of WebSphere Application Server

The **uninstall** command calls the uninstaller program that is created during installation. The uninstaller program is customized for each product installation with specific disk locations and routines for removing installed features.

The uninstaller program removes registry entries, uninstalls the product, and removes all related features. The uninstaller program does not remove log files in the installation root directory.

The following procedure uninstalls the IBM WebSphere Application Server product from Solaris:

1. Log in to your Solaris system as root.
2. Stop all application server processes in all profiles on the machine, for example:

```
bash# cd /opt/IBM/WebSphere/AppServer/profiles/AppSrv01/bin
bash# ./stopServer.sh server1
```

Attention: Remember, if you have enabled administrative security on the profile, you have to add two additional parameters to the **./stopServer.sh** command:

```
bash# ./stopServer.sh -username <admin_username> -password <admin_password>
```

3. After all application server processes are stopped, change the directory to the installation root’s uninstall directory, for example:

```
bash# cd /opt/IBM/WebSphere/AppServer/uninstall
```

4. Issue the **uninstall** command

```
bash# ./uninstall
```

5. You will see the Welcome window for the uninstallation wizard (Figure 3-13).

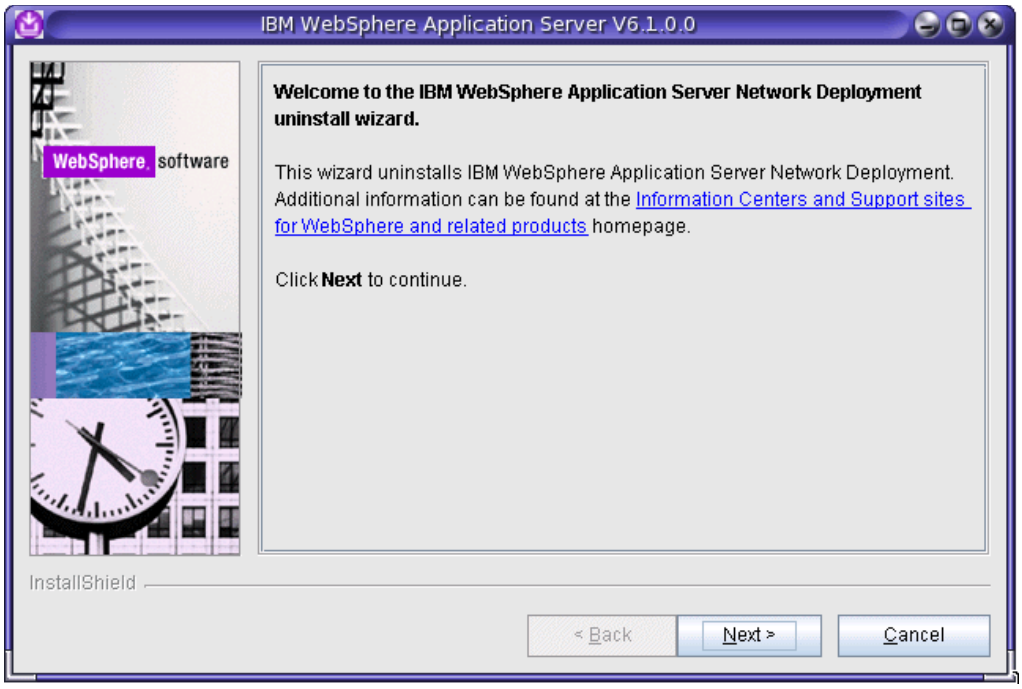


Figure 3-13 Uninstallation wizard: Welcome window

Tip: You can also issue the **`uninstall`** command with a silent parameter to use the wizard without the graphical user interface:

- `bash#`**`./uninstall -silent`** (This is the default behavior, and it removes the product and all profiles.)

  OR

- `bash#`**`./uninstall -silent -OPT removeProfilesOnUninstall="false"`** (Leaves all created profiles intact during the uninstallation process.)

6. Click the **Next** button in the Welcome window shown in Figure 3-13.
7. Click the **Remove all profiles** check box (Figure 3-14 on page 47) if you want to do a complete uninstallation and click **Next**.

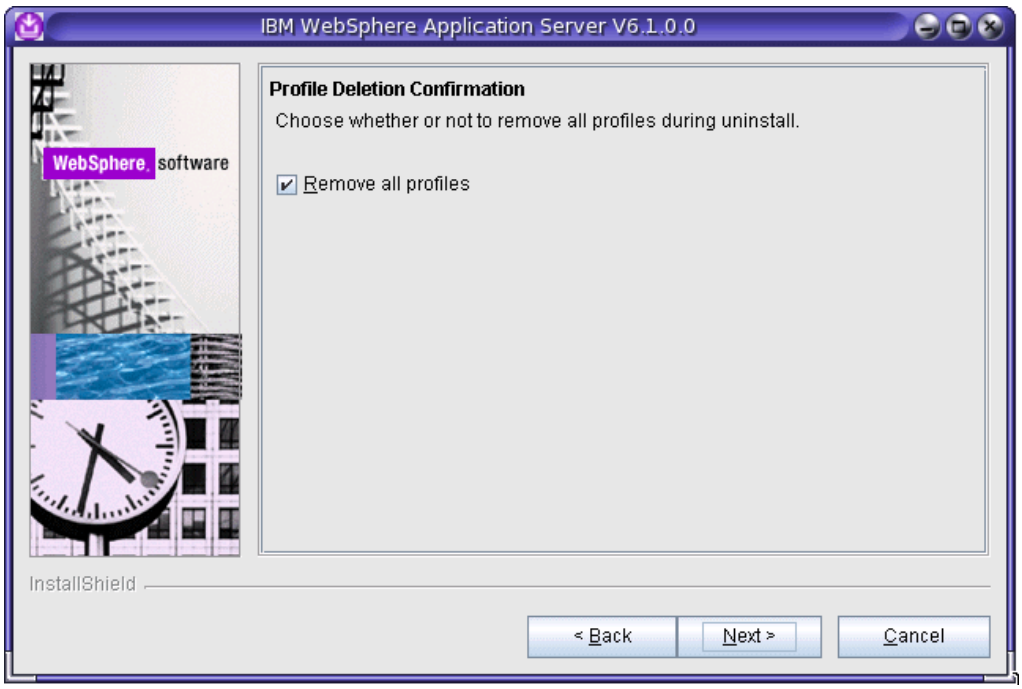


Figure 3-14 Uninstallation wizard: Profile deletion window

Note: When using the wizard, the window allows you to choose whether or not the uninstaller deletes all profiles before it deletes the core product files. By default, all profiles will be deleted, but this option can be deselected on this window (Figure 3-14).

8. Verify the correctness of uninstallation information in the summary window (Figure 3-15). If you have something to correct, click **Back**. If you are ready to continue the uninstallation, click **Next**.

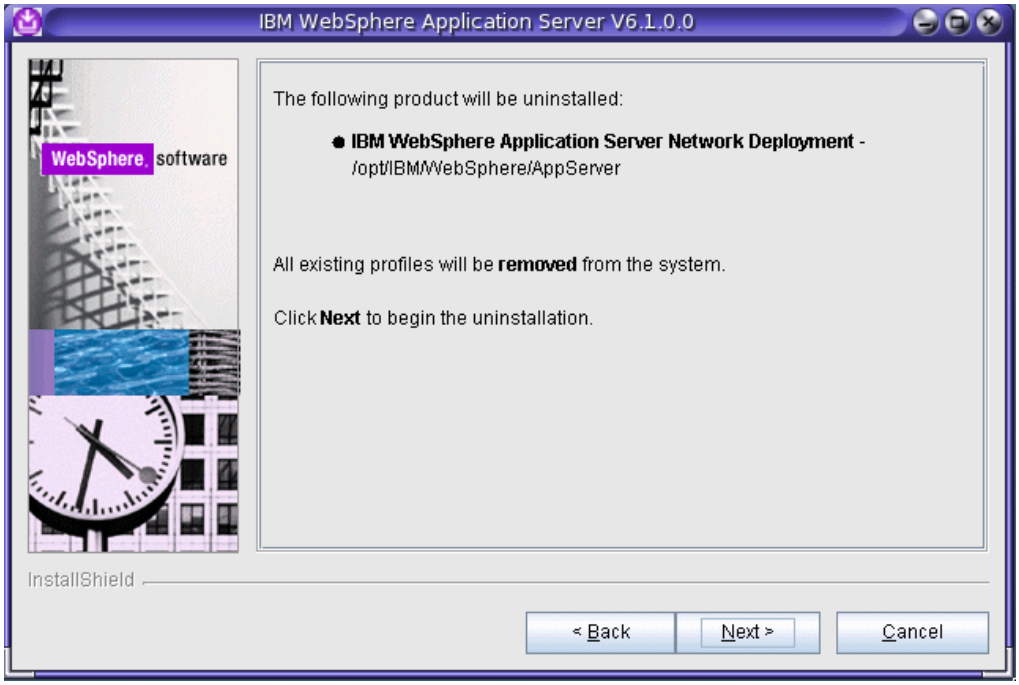


Figure 3-15 Uninstallation wizard: Uninstallation summary window

Attention: You can always click the **Cancel** button to cancel the uninstallation. Be aware that this is the last chance to cancel the uninstallation. After you click the **Next** button shown in Figure 3-15, the uninstallation of WebSphere Application Server starts.

9. After the uninstallation is over, you see the uninstallation completion window (Figure 3-16 on page 49).

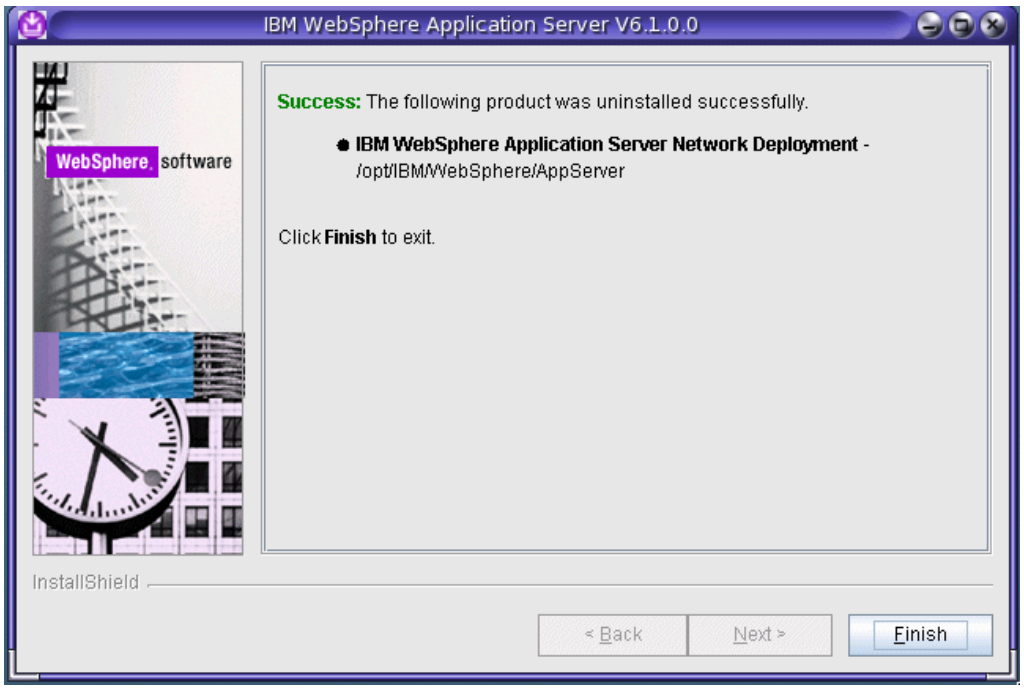


Figure 3-16 Uninstallation wizard: Uninstallation completed

10. Click **Finish** to close the window and the uninstallation program.

Review the log.txt file that is located in the/opt/IBM/WebSphere/AppServer/logs/uninstall directory. The log file records file system or other unusual errors. Look for the `INSTCONFSUCCESS` indicator of successful uninstallation.

If you plan to reinstall, manually uninstall any remaining product files. Remove all artifacts of the product so you can install it into the same installation root directory.

Uninstallation problems

If the WebSphere Application Server is running on your Solaris system and you try to uninstall it, you will see the error message shown in Figure 3-17.

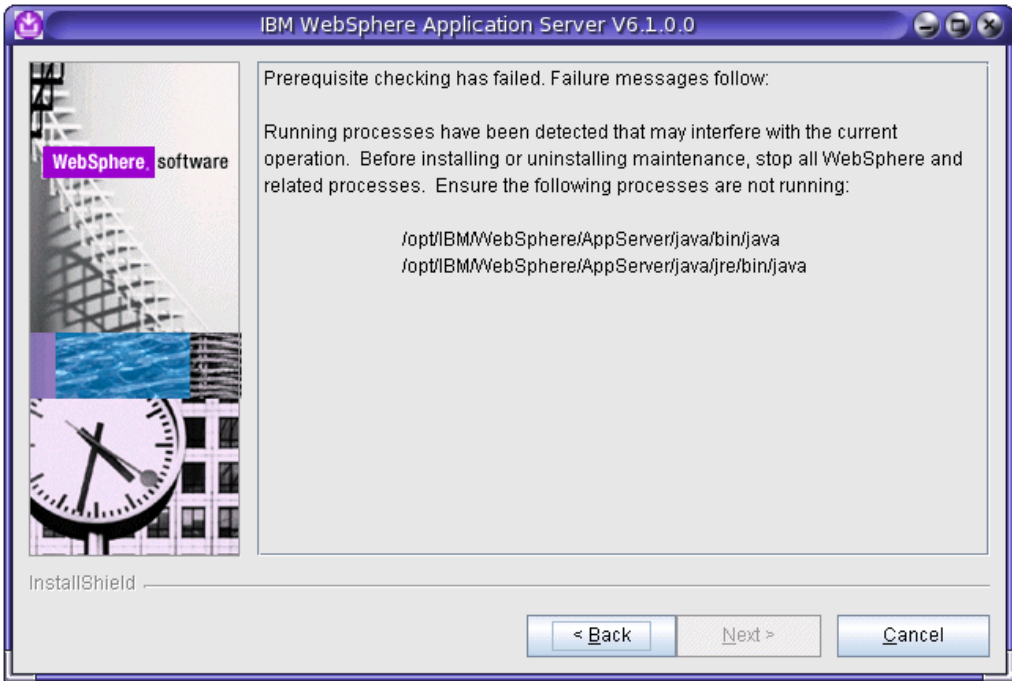


Figure 3-17 Uninstallation wizard: Uninstallation failure due to existing WebSphere Application Server processes

- Click **Cancel** in Figure 3-17 and then click **Yes** in Figure 3-18 to exit the uninstallation wizard.

Figure 3-18 Exit uninstallation wizard window

- After you have exited the uninstallation wizard, stop your currently running Application Server instances and restart the uninstallation from Step 3.

4

Installation of WebSphere Application Server Network Deployment for Solaris 10

This chapter describes the installation of WebSphere Application Server V6.1 Network Deployment in a Sun Solaris 10 environment.

In this chapter, the following topics are discussed:

- Installing WebSphere Application Server V6.1 Network Deployment
- Profile Management Tool
- Installation Factory

4.1 Installing WebSphere Application Server V6.1 Network Deployment

The installation process for WebSphere Application Server V6.1 uses InstallShield Multi-platform (ISPM). It is a full installation and not an upgrade-style installation from previous versions.

All packages in WebSphere Application Server V6.1 continue to offer graphical and silent installation options and also use common logging and tracing for consistency across packages.

The Version 6.1 installer provides a streamlined installation process that makes it faster and easier to set up a complete environment. A Profile can now be created directly by the installer, rather than requiring you to launch the profile management tool after installation. You even have the option to create a deployment manager and an already-federated node in a single step during the installation.

Note: See the *WebSphere Application Server V6 Migration Guide*, SG24-6369 for information about migration from previous versions.

4.1.1 Graphical user interface installation

The graphical user interface installation for WebSphere Application Server network deployment is the same as explained in 3.1.1, “Graphical user interface installation” on page 32.

4.1.2 Silent installation

The silent installation uses a response file similar to that of previous releases, but now uses a single response file for both installation and profile creation. All options that are available during profile creation can be accessed during a 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.

Do the following steps:

1. Log on as root. For information about non-root installation, see Chapter 5, “Configuration of WebSphere Application Server in an advanced Solaris 10 Environment” on page 115.

   In addition, verify that the umask setting is 022. To verify the umask setting, issue the following command:

   ```
   umask
   ```

   To set the umask setting to 022, issue the following command:

   ```
   umask 022
   ```

2. Copy the response file as myoptionsfile to your disk drive and customize it.

   The name of the original file is responsefile.nd.txt.

You must also create a profile for the Network Deployment product to create an operational environment. You can create a deployment manager profile, an application server profile, or a custom profile that becomes a managed node when you add the node into a deployment manager cell.

Do not add an options line to any of the profile creation response files that include the -silent parameter.

The -silent parameter is not required. If it exists in any of the files, the file cannot create a profile during a silent product installation. If you use the Profile creation wizard in silent mode, the additional parameter does not affect the creation of a profile. However, when the silent installation of the Network Deployment product attempts to call a silent profile response file, the parameter prevents the creation of the profile.

Example 4-1 Example of responsefile.nd.txt

```
##############################################################################
## V6.1 InstallShield Options File
#
# Wizard name: Install
# Wizard source: setup.jar
##############################################################################
## License Acceptance
-OPT silentInstallLicenseAcceptance="true"
##############################################################################
## Install a New Copy
-OPT installType="installNew"
##############################################################################
## Install Location
## Solaris or Linux Default Install Location:
-OPT installLocation="/opt/RB/RB-WAS-Install"
# If you are installing as non-root user #
##-OPT installLocation="<user's home>/opt/RB/RB-WAS-Install"
##############################################################################
## Profile Creation Selection
#
-OPT profileType="deploymentManager"
##############################################################################
# Administrative Security
-OPT PROF_enableAdminSecurity="true"
##############################################################################
## Security Options
-OPT PROF_adminUserName=wasadmin
##-OPT PROF_adminPassword=password
##############################################################################
# # Deployment Manager Profile name
-OPT PROF_dmgrProfileName=itsoDmgrPrf01
##############################################################################
# Profile path
-OPT PROF_profilePath="/opt/RB/RB-Profile"
##############################################################################
# Host name
-OPT PROF_hostName=app1
##############################################################################
```

3. Issue the proper command to use your custom response file. For example, issue the following command:

   ```
   mnt_cdrom/WAS/install -options /tmp/WAS/myoptionsfile.txt -silent
   ```

4. When the installation is complete, you should check the log.txt file in the /opt/IBM/WebSphere/AppServer/logs directory and see the `INSTCONFSUCCESS` indicator at the end of log file.

If problems occur, consult the following applicable logs in Table 4-1.

Table 4-1 Installation logs for WebSphere Application Server

Log	Content	Indicators
<was_home> / logs / install/log.txt	Logs all installation events.	0: Success 1: Failure 2: Partial Success
<was_home> / logs / manageprofiles/ <*profile_name*>_create.log	► Traces all events that occur during the creation of the named profile. ► Created when using the Profile Management tool or the **`manageprofiles`** command.	INSTCONFFAIL: Total profile creation failure. INSTCONFSUCCESS: Successful profile creation. INSTCONFPARTIALSUCCESS: Profile creation errors occurred but the profile is still functional. Additional information identifies the errors.
<was_home> / logs /manageprofiles/ <*profile_name*>_delete.log	► Traces all events that occur during the deletion of the named profile. ► Created when using the Profile Management tool or the **`manageprofiles`** command.	INSTCONFFAIL: Total profile deletion failure. INSTCONFSUCCESS: Successful profile deletion. INSTCONFPARTIALSUCCESS: Profile deletion errors occurred but the profile is still functional. Additional information identifies the errors.
<was_home> /logs /pmt.log	Logs all profile creation events that occur when using the Profile Management tool.	INSTCONFFAIL: Total profile creation failure. INSTCONFSUCCESS: Successful profile creation. INSTCONFPARTIALSUCCESS: Profile creation errors occurred but the profile is still functional. Additional information identifies the errors.
<was_home>/logs/install/ installconfig.log.gz	Logs the activities of ANT configuration scripts that run at the end of the installation procedure.	Configuration action failed: Unsuccessful ANT script configuration. Configuration action succeeded: Successful ANT script configuration.

4.2 Profiles

The purpose of this section is to help you build your initial WebSphere Application Server environment after you have installed the product.

WebSphere Application Server V6.1 files are divided into two categories:

- Product Files: Product files include the application binaries needed to run the Application Server.
- User Files: User files contain information used by the Application Server. User files contain defined variables, resources, and log files.

A profile is a collection of these files, creating a WebSphere Application Server runtime environment. When combined, the core product files (or the installed binaries) and the configuration files (or a profile) make up a fully functional WebSphere Application Server installation.

The WebSphere Application Server installation process simply lays down a set of core product files required for the runtime processes. After installation, you need to create one or more *profiles* that define the runtime to have a functional system. The core product files are shared among the runtime components defined by these profiles.

With the Network Deployment package, you have the option of defining multiple application servers with central management capabilities, as summarized in Figure 4-1 on page 56. The administration domain is the cell, consisting of one or more nodes. Each node contains one or more application servers and a node agent that provides an administration point managed by the deployment manager.

The deployment manager can be located on the same machine as one or more of the application servers. This would be a common topology for single machine development and testing environments. In most production topologies, we recommend that the deployment manager be placed on a separate dedicated machine.

The basis for this runtime environment starts with the deployment manager that provides the administration interface for the cell. As you would expect, the deployment manager is defined by a deployment manager profile.

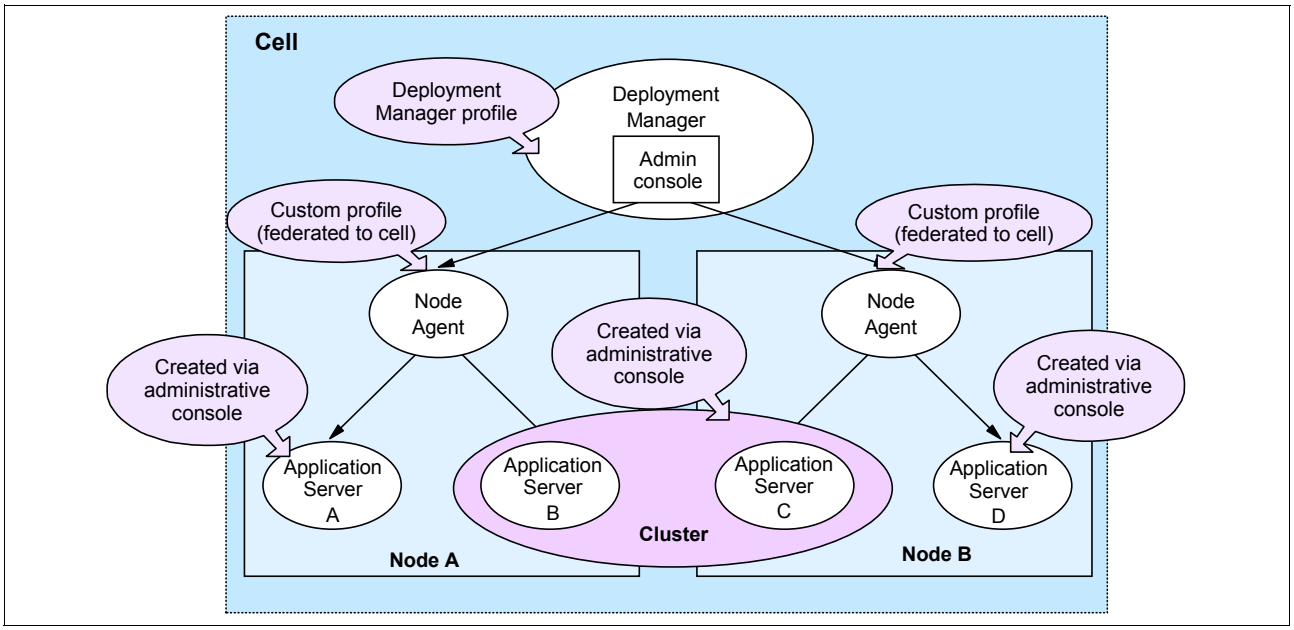


Figure 4-1 System management topology: Network Deployment

Nodes can be added to the cell in one of two ways:

- You can create an application server profile, then federate it to the cell. When a node is added to a cell, a node agent is created on the node and configuration files for the node are added to the master configuration repository for the cell. The deployment manager then assumes responsibility for the configuration of all servers on the node.

  Note that the server name for a federated application server is always going to be "server1".

- You can define a custom profile to create an empty node for federation to the cell. After federation, you can further configure the node by creating application servers and clusters from the deployment manager administrative console. If you are using a naming convention for servers, this is the best option.

Note: You can also create stand-alone application servers with the Network Deployment package, though you would most likely do so with the intent of federating that server into a cell for central management.

Types of profiles

We mentioned the types of profiles available for defining the runtime. In the following sections, we take a closer look at these profiles.

Application server profile

The *application server profile* defines a single stand-alone application server. Using this profile gives you an application server that can run stand-alone, or unmanaged. The environment will have the following characteristics:

- The profile consists of one cell, one node, and one server. The cell and node are not relevant in terms of administration, but you see them when you administer the server through the administrative console scopes.
- The name of the application server is "*server1*".
- The application samples are installed on the server (optional).

- The server has a dedicated administrative console.

The primary use for this type of profile is:

- To build a stand-alone server in a Base or Express installation.
- To build a stand-alone server in a Network Deployment installation that is not managed by the deployment manager (a test machine, for example).
- To build a server in a distributed server environment to be federated and managed by the deployment manager. If you are new to WebSphere Application Server and want a quick way of getting an application server complete with samples, this is a good option. When you federate this node, the default cell becomes obsolete and the node is added to the deployment manager cell. The server name remains “server1” and the administrative console is removed from the application server.

Deployment manager profile

The deployment manager profile defines a deployment manager in a distributed server environment. Although you could conceivably have the Network Deployment package and run only stand-alone servers, this would bypass the primary advantages of Network Deployment, which is workload management, failover, and central administration.

In a Network Deployment environment, you should create one deployment manager profile. This gives you:

- A cell for the administrative domain
- A node for the deployment manager
- A deployment manager with an administrative console
- No application servers

Once you have the deployment manager, you can:

- Federate nodes built either from existing application server profiles or custom profiles.
- Create new application servers and clusters on the nodes from the administrative console.

Custom profile

A custom profile is an empty node, intended for federation to a deployment manager. This type of profile is used when you are building a distributed server environment. Use a custom profile in the following way:

1. Create a deployment manager profile.
2. Create one custom profile on each node on which you will run application servers.
3. Federate each custom profile to the deployment manager, either during the custom profile creation process or later by using the **addNode** command.
4. Create new application servers and clusters on the nodes from the administrative console.

Cell profile

Cell profile (new): This new option allows you to quickly set up a distributed server environment on a single system.

A cell profile is actually a combination of two profiles: a deployment manager profile and an application server profile. The application server profile is federated to the cell. The deployment manager and application server reside on the same system. This type of profile lets you get a quick start with a distributed server environment and is especially useful for test environments that typically have all nodes on one test system.

Directory structure

If you have worked with previous versions of WebSphere Application Server, you will notice a difference in the directory structure. First, all packages (Base, Express, and Network Deployment) specify the same default root directory during installation. In this IBM Redbooks publication, we refer to the root directory as the *<was_home>* directory.

In addition to the traditional directories under the *<was_home>* directory (bin, config, installedapps, and so on), you now have a profiles directory containing a subdirectory for each profile you create and allow to use the default home location. The directory structure for each profile resembles the primary structure. In other words, there is a bin, config, installedApps, and other directories required for a unique runtime under each profile.

However, profiles can be stored in any folder, so we suggest storing them in a more accessible structure (by default, there are at least six levels). We refer to the root of each profile directory (by default *<was_home>/profiles/profile_name*) as *<profile_home>*. Example 4-2 assumes that a profile named itsoAppServer01 exists.

Why do we emphasize this point? If you enter commands while in the *<was_home>/bin* directory, they are executed against the runtime defined by the default profile. The default profile is determined by the following items:

- The profile was defined as the default profile when you created it. The last profile specified as the default takes precedence. You can also use the **manageprofiles** command to specify the default profile.
- If you have not specified the default profile, it will be the first profile you created.

To make sure command-line operations are executed for the correct runtime, you need to do one of two things:

- Specify the -profileName option when using a command and execute the command from the *<was_home>/bin* directory.
- Execute the command from its *<profile_home>*/bin directory.

Example 4-2 Profile directory

```
was_home/profiles/itsoAppSrv01/bin
was_home/profiles/itsoAppSrv01/config
was_home/profiles/itsoAppSrv01/configuration
was_home/profiles/itsoAppSrv01/etc
was_home/profiles/itsoAppSrv01/firststeps
was_home/profiles/itsoAppSrv01/installableApps
was_home/profiles/itsoAppSrv01/installedApps
was_home/profiles/itsoAppSrv01/installedConnectors
was_home/profiles/itsoAppSrv01/installedFilters
was_home/profiles/itsoAppSrv01/logs
was_home/profiles/itsoAppSrv01/properties
was_home/profiles/itsoAppSrv01/samples
was_home/profiles/itsoAppSrv01/temp
was_home/profiles/itsoAppSrv01/wstemp
```

4.2.1 Building a system using profiles

During the planning cycle, a topology was selected for the WebSphere Application Server environment. There are many topologies to choose from, each with its own unique features.

However, when we discuss using profiles to build a WebSphere Application Server environment, we are focusing on the WebSphere Application Server processes. Regardless of the topology you select, there are really only two primary situations to consider when deciding which profiles you need to create:

- You plan to create one or more stand-alone application servers. We will refer to this as a *stand-alone server environment*.
- You plan to create a deployment manager and one or more nodes with application servers. We refer to the application servers in this environment as managed servers. These nodes can coexist or reside on different machines. We refer to this as a *distributed server environment*.

The following topics will give the basic steps for each. You can extend this to suit your own environment.

Stand-alone server environment

If you are creating a stand-alone application server, install your choice of Base, Express, or Network Deployment on the system.

An application server profile is created during the installation of Express and Base. With Network Deployment, you have the option of creating a profile of any type, including an application server profile.

Create an application server profile on that system. Since you have an application server automatically with the Base and Express installation, you only need to do this if you want an additional stand-alone server environment.

Distributed server environment

There are two options for building this environment. The option you select depends on your circumstance. If you are building a new production environment from scratch, we would recommend method 1. Either method is fine for a development or test environment.

Note: When defining multiple deployment managers or application servers on a single machine or LPAR, you need to ensure that the ports and names you select for each are unique. For more information about ports, see *Planning and Designing for WebSphere Application Server V6.1,* SG24-7305.

Method 1

This method assumes that you do not have a stand-alone application server to federate, but instead will create application servers from the deployment manager. This gives you a little more control over the characteristics of the application servers during creation, including the server name (all application servers created with the application server profile are named server1). You can also create an application server, customize it, and then use it as a template for future application servers you create. If you are using clustering, you can create the cluster and its application servers as one administrative process.

When you create an application server this way, you do not automatically get the sample applications, but can install them later if you want.

The process to follow for this method is:

1. Install Network Deployment on a server. If this is a multiple-machine install with the deployment manager on one machine and application servers on one or more separate machines, install the product on each machine.
2. Create a deployment manager profile on the deployment manager machine and start the deployment manager.
3. Create and federate a custom profile on the application server machine and start the node. You can federate the node to the cell as part of the profile creation process, or you can elect to do it manually as a second step.
4. Verify that the node agent is started. It should be started automatically as part of the federation process.
5. Open the deployment manager's administrative console, then create application servers or clusters on the custom profile node from the administrative console.

Method 2

This method assumes you will federate an application server profile to the cell. With the application server profile, you have an existing application server (server1) and might have applications installed, including the sample applications and any user applications you have installed. Do the following steps:

1. Install Network Deployment on the server. If this is a multiple machine install (deployment manager on one and application servers on one or more separate machines), install the product on each machine.
2. Create a deployment manager profile on the deployment manager machine and start the deployment manager.
3. Create an application server profile on the application server machine and start the application server.
4. Open the deployment manager's administrative console and add the node defined by the application server profile to the cell.
5. This deletes the application server cell, and federates the node to the deployment manager cell. If you want to keep applications that have been installed on the server, be sure to specify this when you federate the node by checking the **Include applications** box in the administrative console. Alternatively, use the -includeapps option with the **`addNode`** command if you are federating the node from the command line.

The new node agent is started automatically by the federation process, but you need to start the application server manually.

4.2.2 Creating profiles with Profile Management Tool

This section shows how to create profiles using the Profile Management Tool.

Important: Note that the Profile Management Tool is not available on 64-bit platforms. With this platform, **`manageprofiles.sh`** can be used instead. For details of the **`manageprofiles`** command, see 4.3.1, "Using the manageprofiles command" on page 90

The Profile Management Tool replaces the Profile Creation Tool from previous versions. It is based on the Eclipse Rich Client Platform, as opposed to the previous Profile Creation Tool, which was based on InstallShield Multi Platform.

The Profile Management Tool provides:

- Multiple profile templates, including the ability to create a cell in a single step
- Feature selection during profile creation
- Option to enable administrative security during profile creation
- Port conflict resolution enhancements
- Option to create a Web server definition

The Profile Management Tool provides multiple profile templates, including the new cell template, which has the ability to create a cell in a single step. It allows you to select a variety of features during the profile creation, including the ability to enable administrative security out of the box. It also offers enhanced port conflict resolution functionally and allows you to optionally create a Web server definition at profile creation time.

The first steps are the same, regardless of the type of profile you will create. You can start the Profile Management Tool in the following way:

1. Use the platform-specific command in the *<was_home>/bin/ProfileManagement* directory. For example, for Solaris, use **pmt.sh**.
2. Use the box (directly after installation) from the install wizard to launch the Profile Management Tool.
3. When you start the wizard, the first window you see is the Welcome window. Click **Next** to select the type of profile you will create, as shown in Figure 4-2.

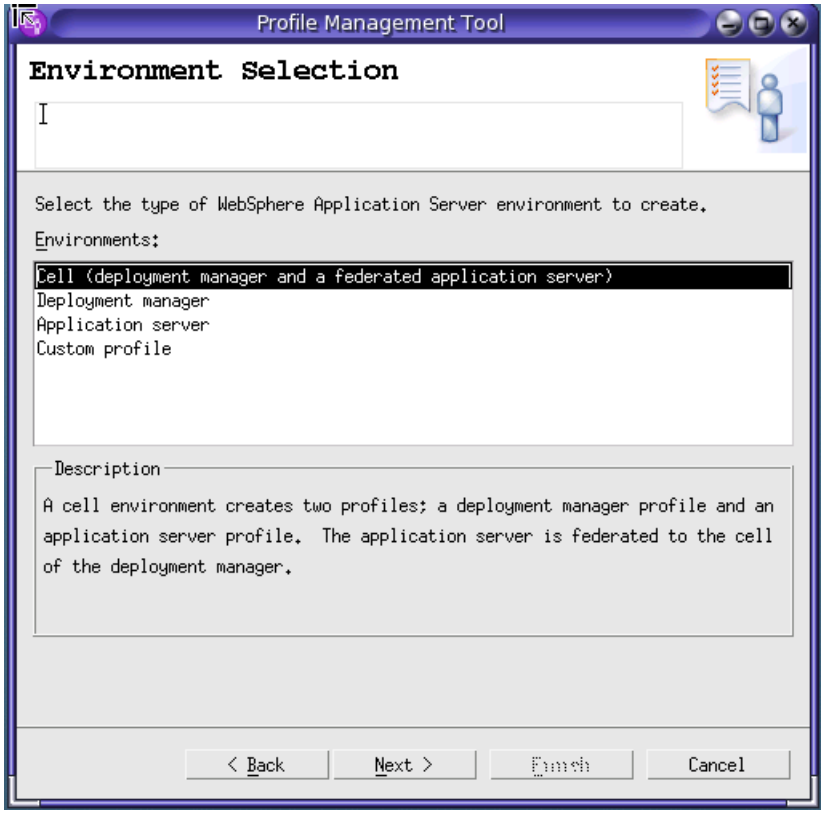


Figure 4-2 Creating a profile: Profile type selection

The rest of the wizard varies, depending on the type of profile you are creating. The steps to create each type of profile are discussed more in the following sections.

Default profiles: As you create profiles, you will have the option of specifying a default profile. This is the profile that commands are executed against if you execute them from the *<was_home>*/bin directory and you do not specify the -profileName argument. The default profile is the first profile that you create, unless you subsequently specify another profile as the default. To see this option, you must take the Advanced path through the Profile Management Tool.

At the end of the Profile Management Tool, you have the opportunity to start the First steps interface. This interface helps you start the deployment manager or application server and has other useful links, such as opening the administrative console, migration help, starting the Profile Management Tool, and installation verification.

Each profile you create has its own First steps program located here:

<profile_home>/firststeps/firststeps.sh

If you choose not to start the First steps program at the completion of the wizard, you can start it later from this location.

You will always have two options when using the Profile Management Tool to create a profile. The “Typical” path will determine a set of default values to use for most settings without giving you the option to modify them. The “Advanced” path lets you specify values for each option.

Attention: The Profile Management Tool is mostly used for creating profiles and cannot perform functions like delete, listProfiles, or getName. Only the **`manageprofiles`** command can perform these functions. For more details, refer to 4.3, “Managing profiles” on page 90

4.2.3 Creating a deployment manager profile

Table 4-2 shows a summary of the options you have for creating a deployment manager. The table shows the options and results you will see depending on which path (typical or advanced) you take.

Table 4-2 Deployment manager profile options

Typical settings	Advanced options
The administrative console is deployed by default.	You can choose whether to deploy the administrative console. We recommend that you do so.
The profile name is Dmgr*xx* by default, where *xx* is 01 for the first deployment manager profile and increments for each one created. The profile is stored in *<was_home>*/profiles/Dmgr*xx*.	You can specify the profile name and its location.
The cell name is *<hostname>*Cell*xx*. The node name is *<hostname>*CellManager*xx*. The host name is pre-filled with your system’s host name.	You can specify the node, host, and cell names.
You can enable administrative security (yes or no). If you select yes, you will be asked to specify a user name and password that will be given administrative authority.	
TCP/IP ports will default to a set of ports not used by any profiles in this WebSphere installation instance.	You can use the recommended ports (unique to the installation), use the basic defaults, or select port numbers manually.

The following steps outline the process of creating a deployment manager:

1. Start the Profile Management Tool and click **Next** on the Welcome page.
2. Select the **Deployment manager profile** option. Click **Next**.
3. Select whether to take the typical settings or to go through the advanced windows. The options you see next depend on the path you take:
   - If **Typical** is selected, then you will only see one more option (to enable security).
   - If **Advanced** is selected, you will continue with the following steps.
4. Select whether to deploy the administrative console application. This is recommended, but if you choose not to, you can install it after profile creation.
5. Enter a unique name for the profile or accept the default. The profile name will become the directory name for the profile files (see Figure 4-3). Click the box if you want this to be the default profile for receiving commands. Select the location for the profile and click **Next**.

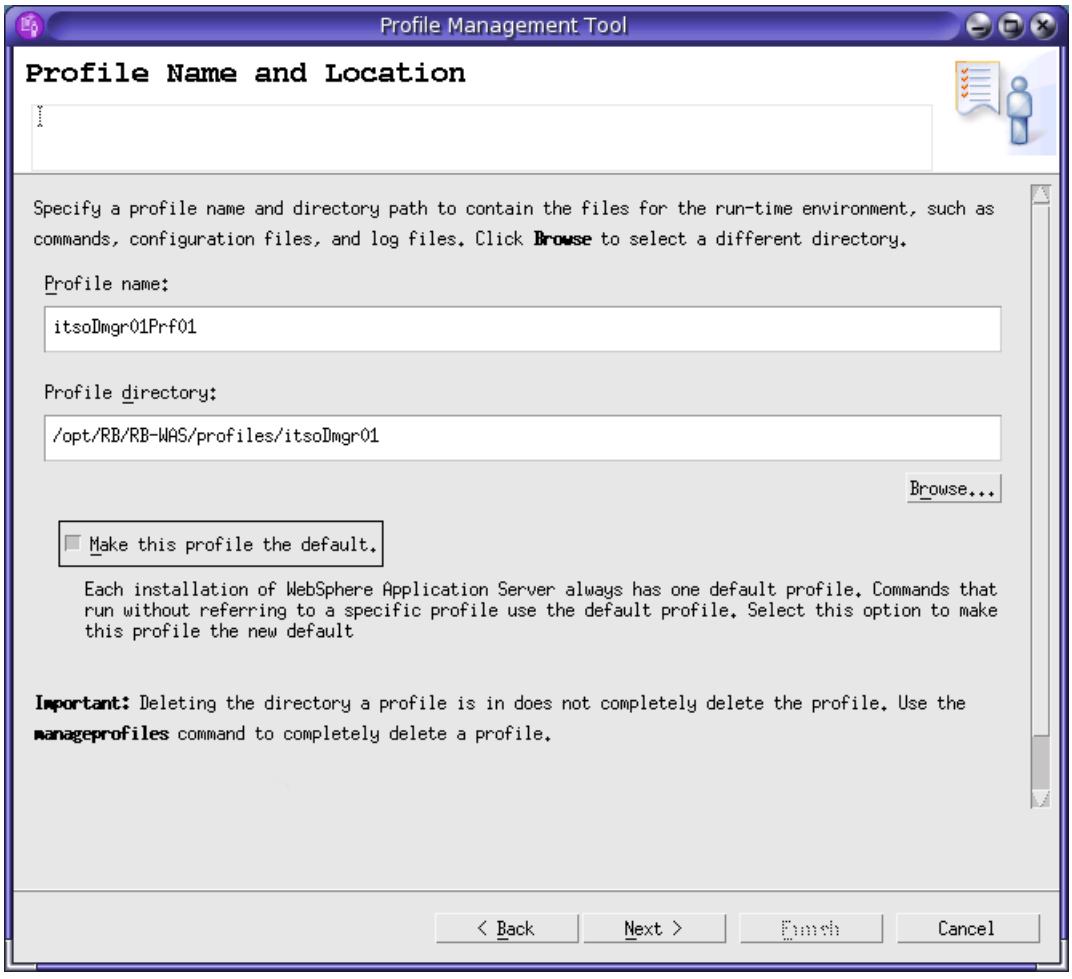


Figure 4-3 Creating a deployment manager profile: Enter name and location

6. Enter the node, host, and cell names. These names will have default values based on the host name of your system. The wizard recognizes if there are existing cells and nodes in the installation and takes this into account when creating the default names, as shown in Figure 4-4.

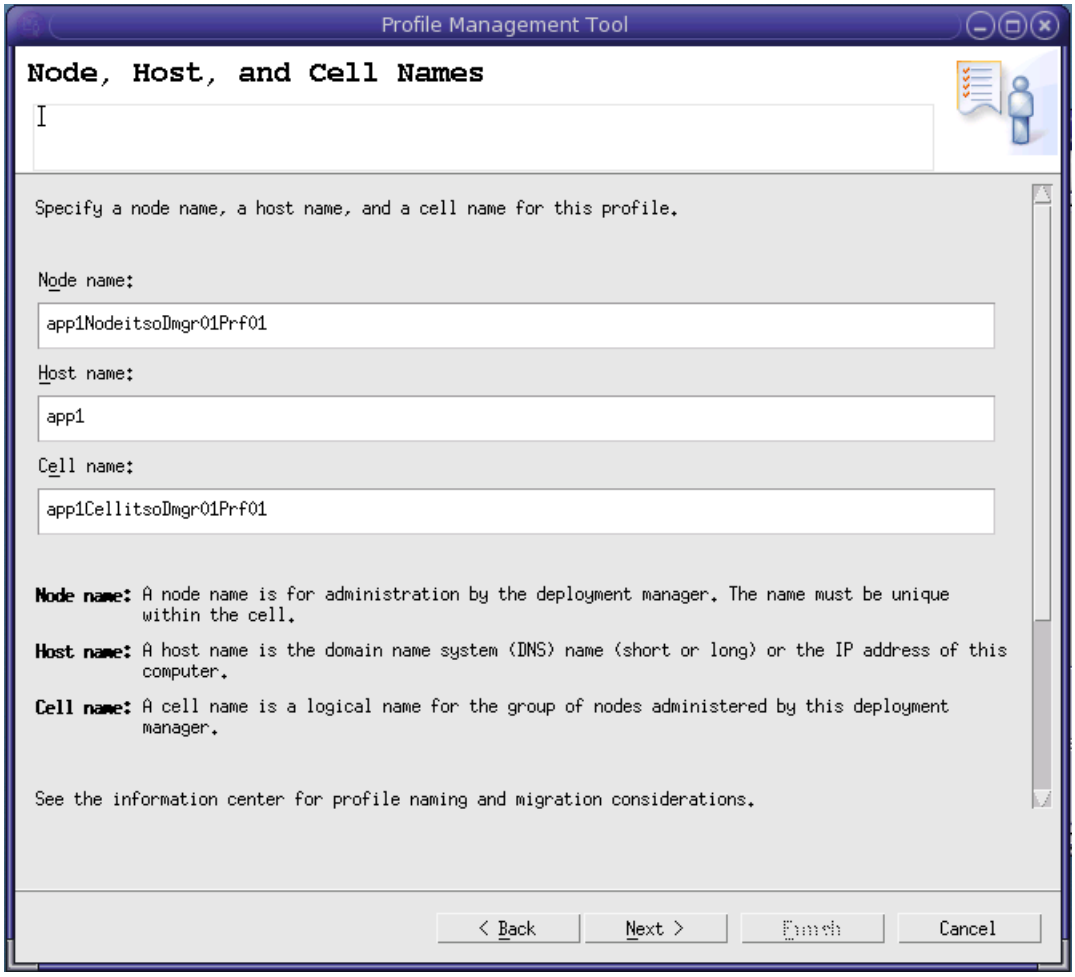


Figure 4-4 Creating a deployment manager profile: Enter cell, host, and node names

Click **Next**.

7. Choose whether to enable administrative security. If you enable security here, you will be asked for a user ID and password that will be added to a file-based user registry with the Administrative role. See Figure 4-5.

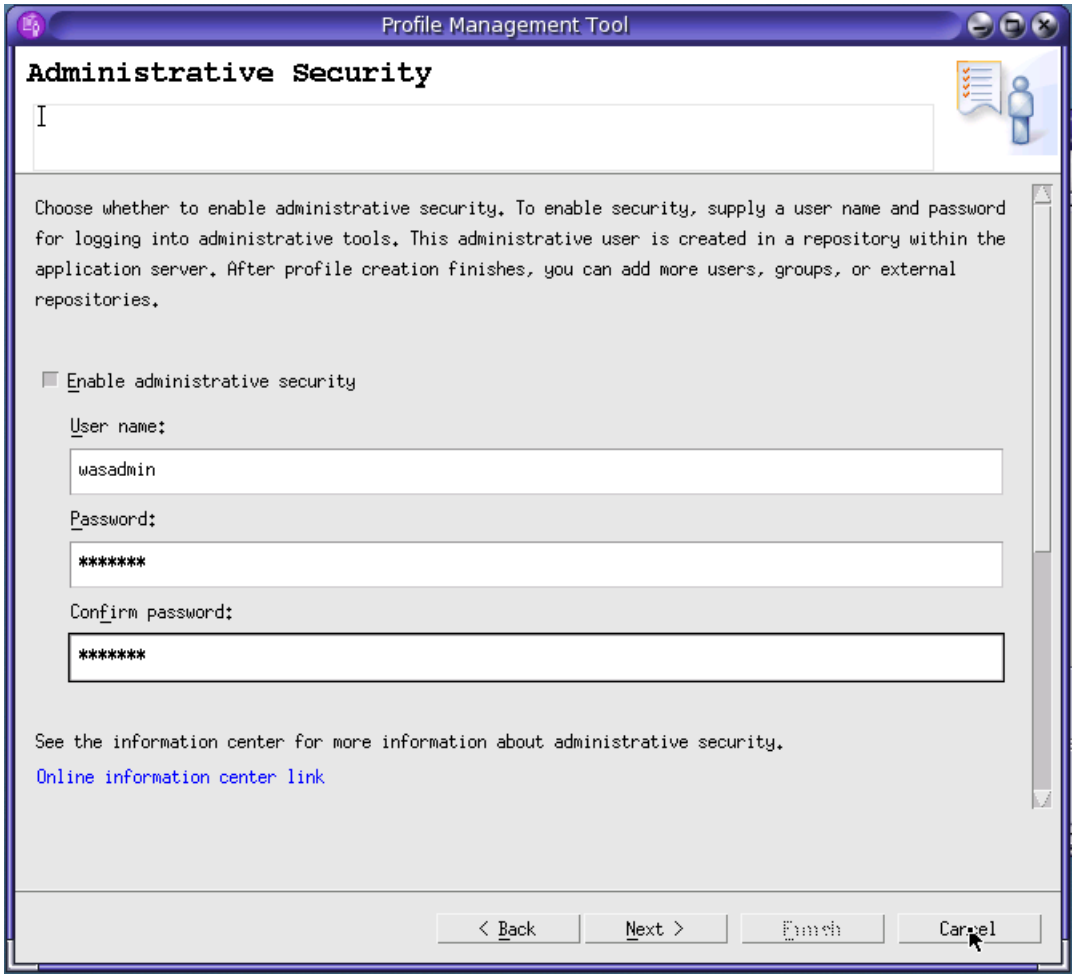


Figure 4-5 Creating a deployment manager profile: Security

Click **Next**.

Note: This user does not have to be defined on the OS. Remember the user name and password you type here. You cannot log onto application server administrative console without it.

8. The wizard presents a list of TCP/IP ports for use by the deployment manager. If you already have existing profiles on the system, this is taken into account when the wizard selects the port assignments. However, you should verify that these ports will be unique on the system. See Figure 4-6.

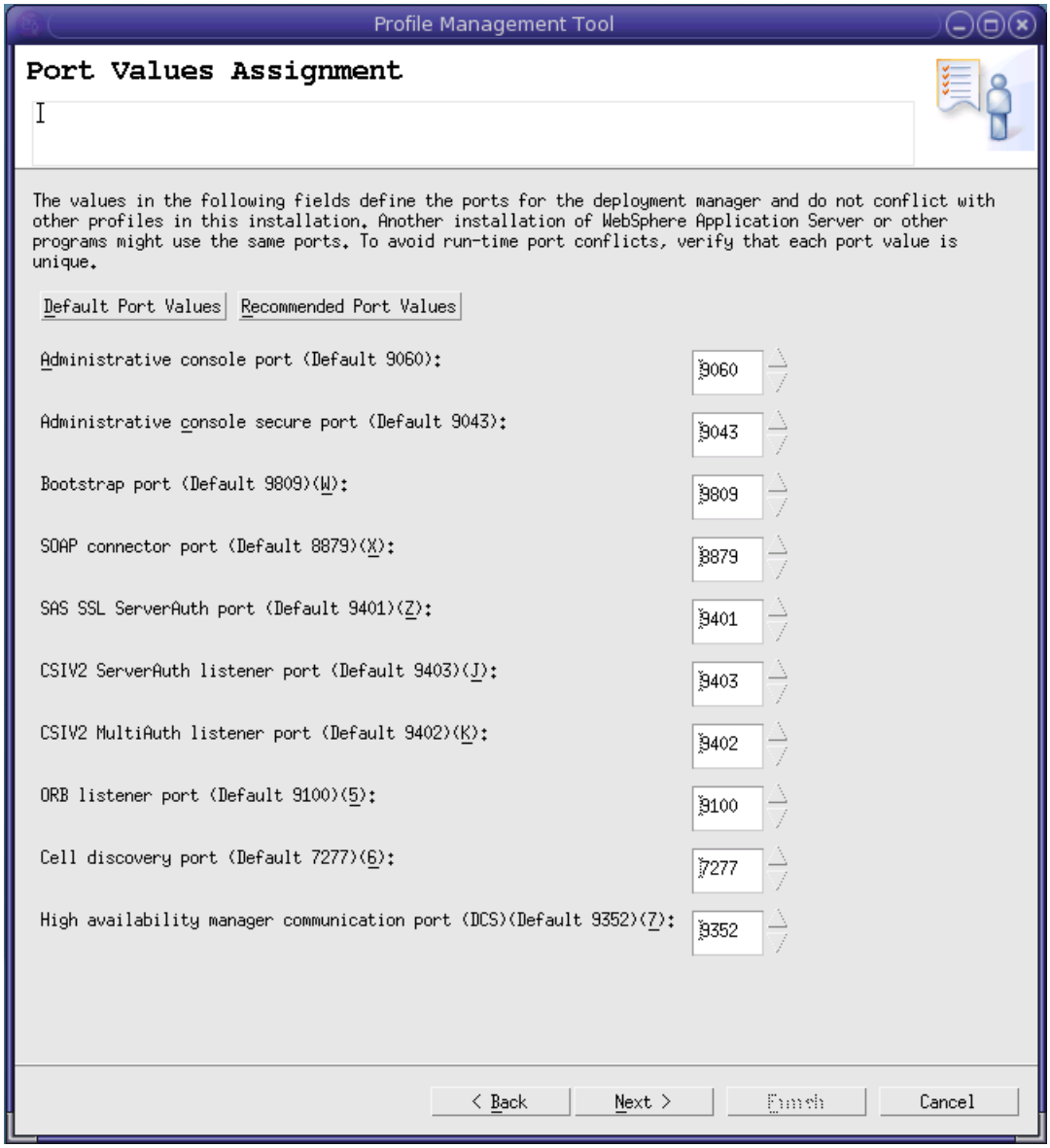


Figure 4-6 Creating a deployment manager profile: Select ports

Important: You might want to note the following ports for later use:

- *SOAP connector port*: If you use the **addNode** command to federate a node to this deployment manager, you need to know this port number. This is also the port you connect to when using the **wsadmin** administration scripting interface.
- *Administrative console port:* You need to know this port in order to access the administrative console. When you turn on security, you need to know the *Administrative console secure port*.

The final window indicates the success or failure of the profile creation. See Figure 4-7 on page 67.

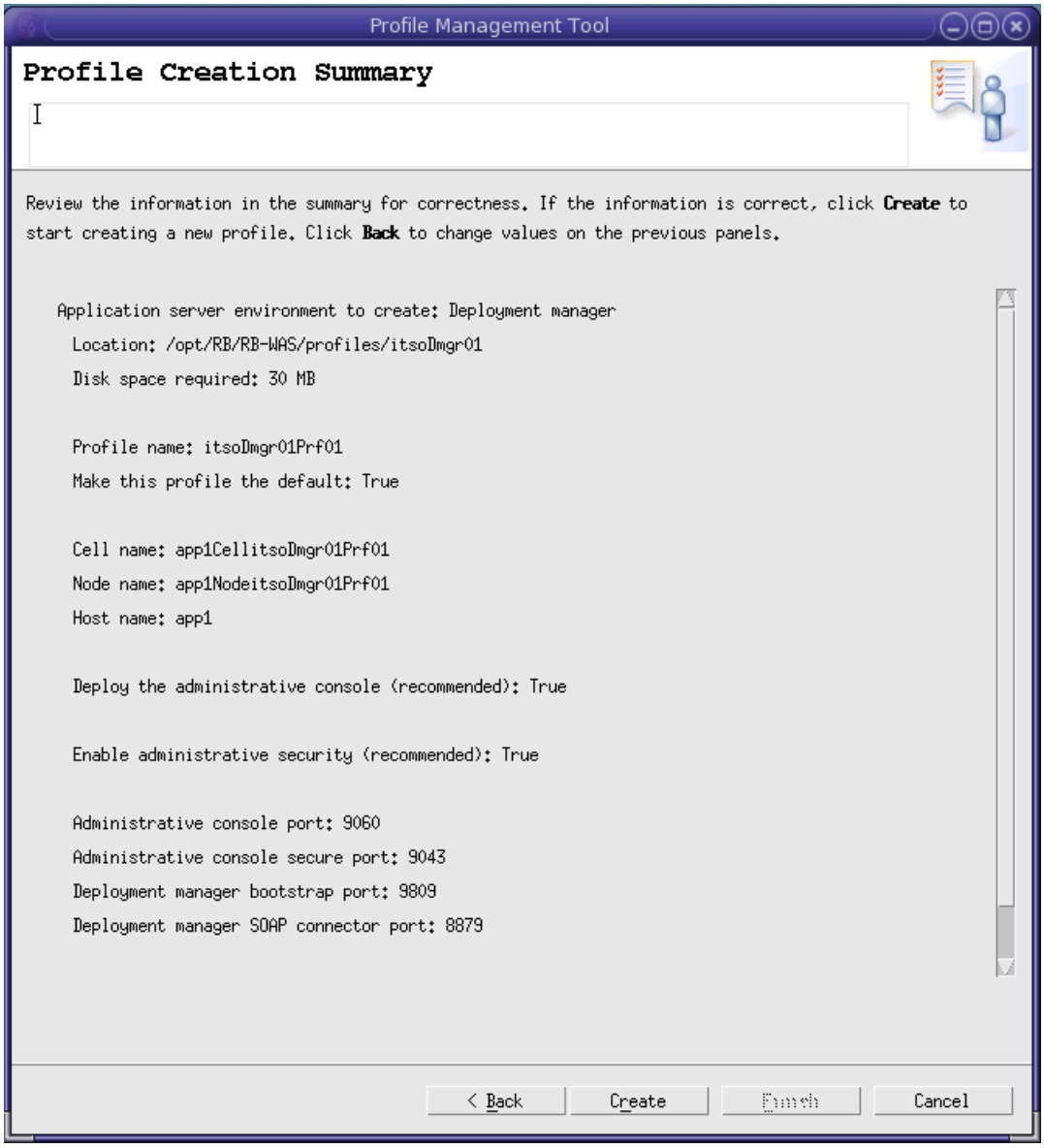


Figure 4-7 Creating a deployment manager profile: Summary

If you have errors, check the log at:

<was_home>/logs/manageprofiles/*<profile_name>*_create.log

You will also find logs for individual actions stored in:

<was_home>/logs/manageprofiles/*<profile_name>*

9. Click **Finish** to close the wizard and start the First steps application, as shown in Figure 4-8.

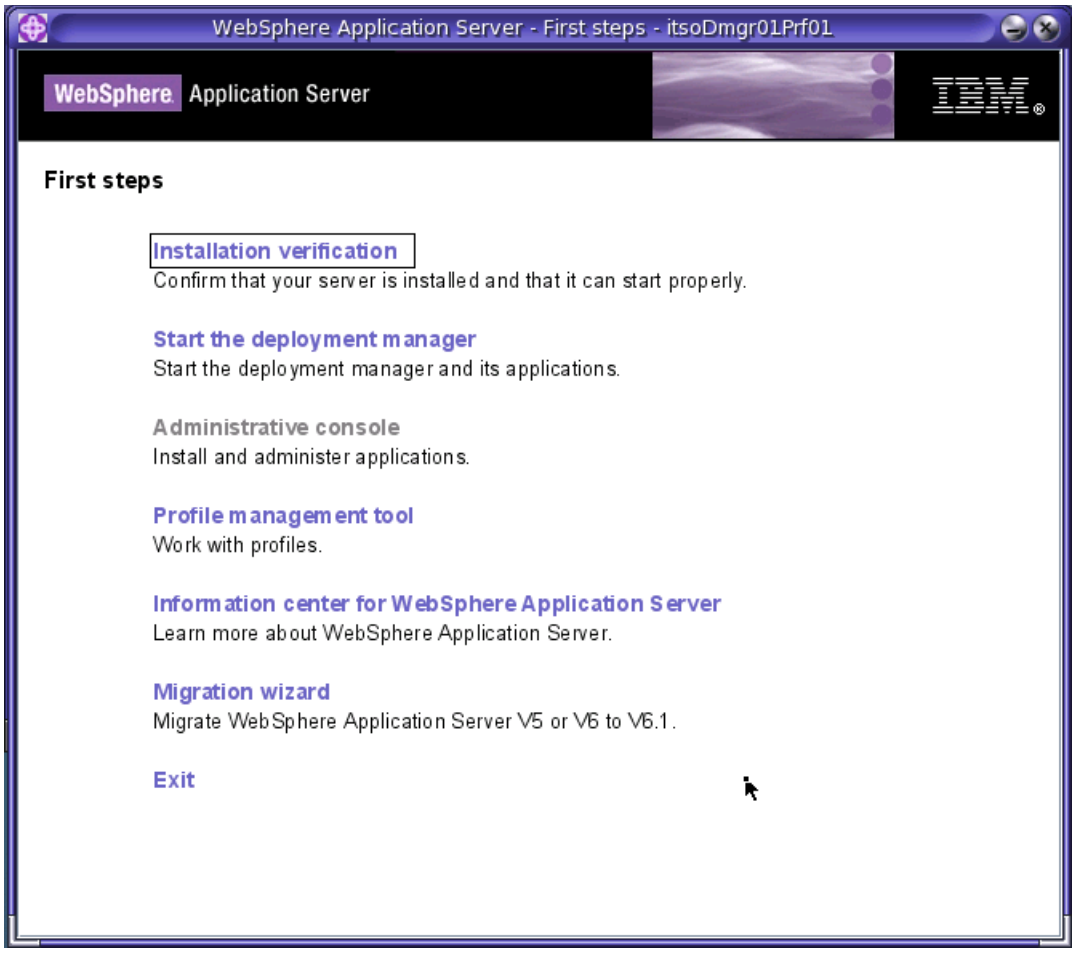


Figure 4-8 Deployment manager First steps menu

Check your results

If the creation was successful, do the following to familiarize yourself with the profile and how to use it:

1. View the directory structure and find the new profile. In this IBM Redbooks publication, we refer to the location as *<profile_home>*. This is where you find, among other things, the config directory containing the deployment manager configuration files, the bin directory for entering commands, and the logs directory where information is recorded.
2. Verify the installation. You can do this directly from the First steps menu. This process starts the deployment manager and checks the log file for warnings or errors on startup. Messages are displayed on the First steps window and logged in the following places:
   - *<profile_home>*/logs/dmgr/startServer.log
   - *<profile_home>*/logs/dmgr/SystemOut.log
3. Open the administrative console, either by selecting the option in the First steps window, or by accessing its URL from a Web browser:

   ```
   http://<dmgr_host>:<admin_console_port>/ibm/console
   ```

   Here is a sample URL in the address bar:

   ```
   http://localhost:9060/ibm/console/
   ```

The administrative console port of 9060 was selected during the Profile Management Tool. See Figure 4-6 on page 66.

Click the **Log in** button. If you did not enable security, you do not have to enter a user name. If you choose to enter a name, it can be any name. It is used to track changes you make from the console. If you enabled security, enter the user ID and password you specified.

4. Display the configuration from the console. You should be able to see the following items from the administrative console:
   a. Cell information: Select **System administration** → **Cell**.
   b. Deployment manager: Select **System administration** → **Deployment manager**.
   c. Deployment manager node: Select **System administration** → **Nodes**.
   d. The default node group: Select **System administration** → **Node groups**.

   Note that at the completion of this process you will not have:

   a. A node agent

      Node agents reside on nodes with managed application servers. You will not see node agents appear until you federate a node to the cell.

   b. Application servers
5. Stop the deployment manager. You can do this from the First steps menu, or better yet, use the **stopManager** command:

```
cd <profile_home>/bin
./stopManager.sh -username <username> -password <password>
```

Tip: In the same manner, you can use the **startManager** command to start the deployment manager, but the user name and password are not required.

4.2.4 Creating an application server profile

An application server profile defines a new stand-alone application server. This server can be run stand-alone or can be later federated to a deployment manager cell for central management.

Table 4-3 shows a summary of all steps involved in process of creating a application server profile.

Table 4-3 Application server profile options for V6.1

Typical	Advanced
The administrative console and default application are deployed by default. The sample applications are not deployed.	You have the option to deploy the administrative console (recommended), the default application, and the sample applications (if installed).
The profile name is AppSrv*xx* by default, where *xx* is 01 for the first application server profile and increments for each one created. The profile is stored in *<was_home>*/profiles/AppSrv*xx*.	You can specify the profile name and its location.
The profile is not the default profile.	You can choose whether to make this the default profile. (Commands run without specifying a profile will be run against the default profile.)
The application server is built using the default application server template.	You can choose the default template, or a development template that is optimized for development purposes.

Typical	Advanced
The node name is <*host*>Node*xx*. The host name is pre-filled in with your system's DNS host name.	You can specify the node name and host name.
You can enable administrative security (yes or no). If you select yes, you will be asked to specify a user name and password that will be given administrative authority.	
TCP/IP ports will default to a set of ports not used by any profiles in this WebSphere installation instance.	You can use the recommended ports (unique to the installation), use the basic defaults, or select port numbers manually.
Does not create a Web server definition.	Allows you to define an external Web server to the configuration.

This section takes you through the steps of creating the application server profile:

1. Start the Profile Management Tool. Click **Next** on the Welcome page.
2. Select the **Application server profile** option. Click **Next**.
3. Select the kind of creation process you want to run: **Typical** or **Advanced**.
   - If Typical is selected, then you will only see one more option (to enable security).
   - If Advanced is selected, you will continue with the next step.
4. Select whether you want to deploy the administrative console and the default application. If you have installed the sample applications (optional during WebSphere Application Server installation), then you can opt to deploy these as well.
5. Enter a unique name for the profile or accept the default. The profile name will become the directory name for the profile files. See Figure 4-9 on page 71.

   Click the box if you want this directory to be the default profile for receiving commands.

   If the application server will be used primarily for development purposes, check the option to create it from the development template.

   Click **Next**.

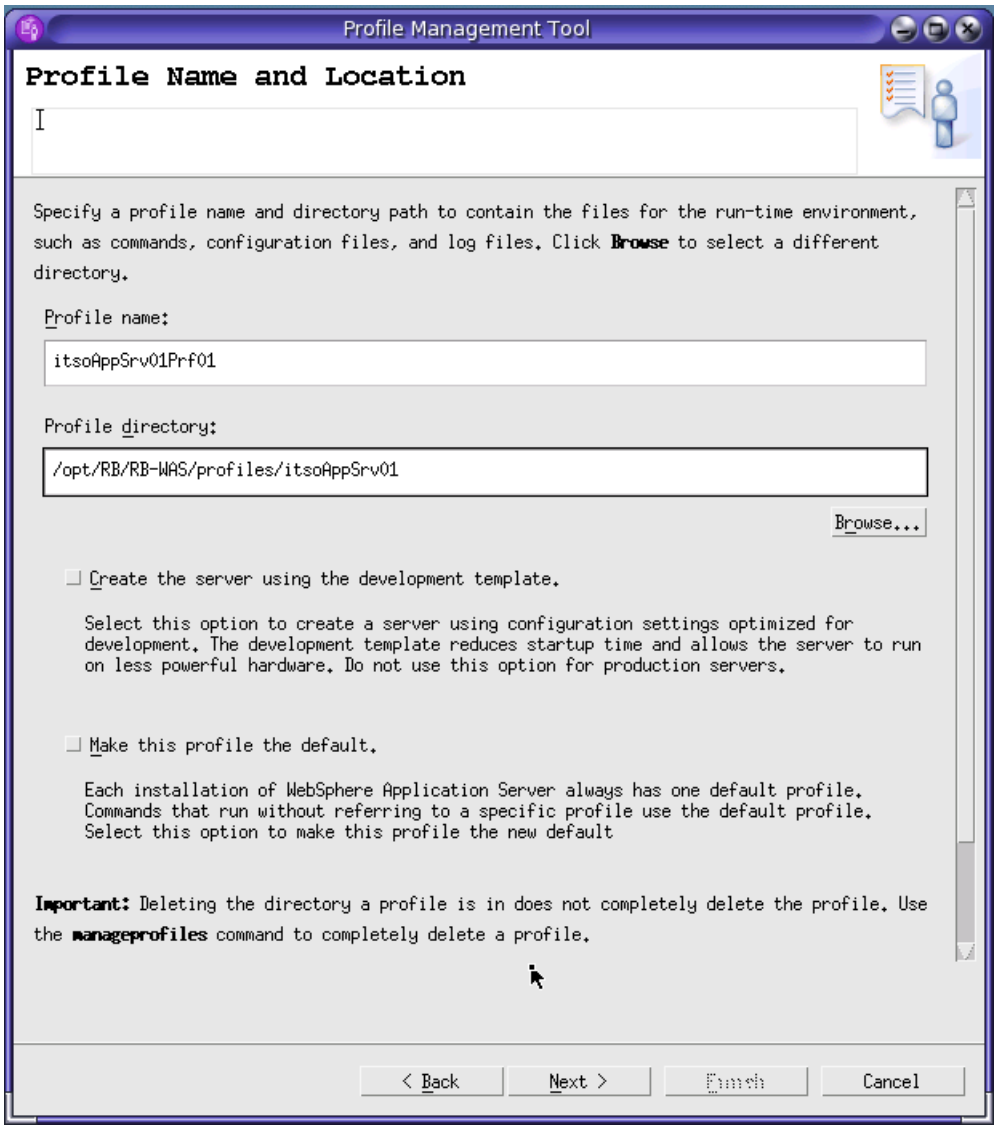


Figure 4-9 Creating an application server profile: Enter name and location

6. Enter the new node name and the system host name, as shown in Figure 4-10. The node name's default is based on the host name of your system. The wizard recognizes if there are existing nodes in the installation and takes this into account when creating the default node name. Click **Next**.

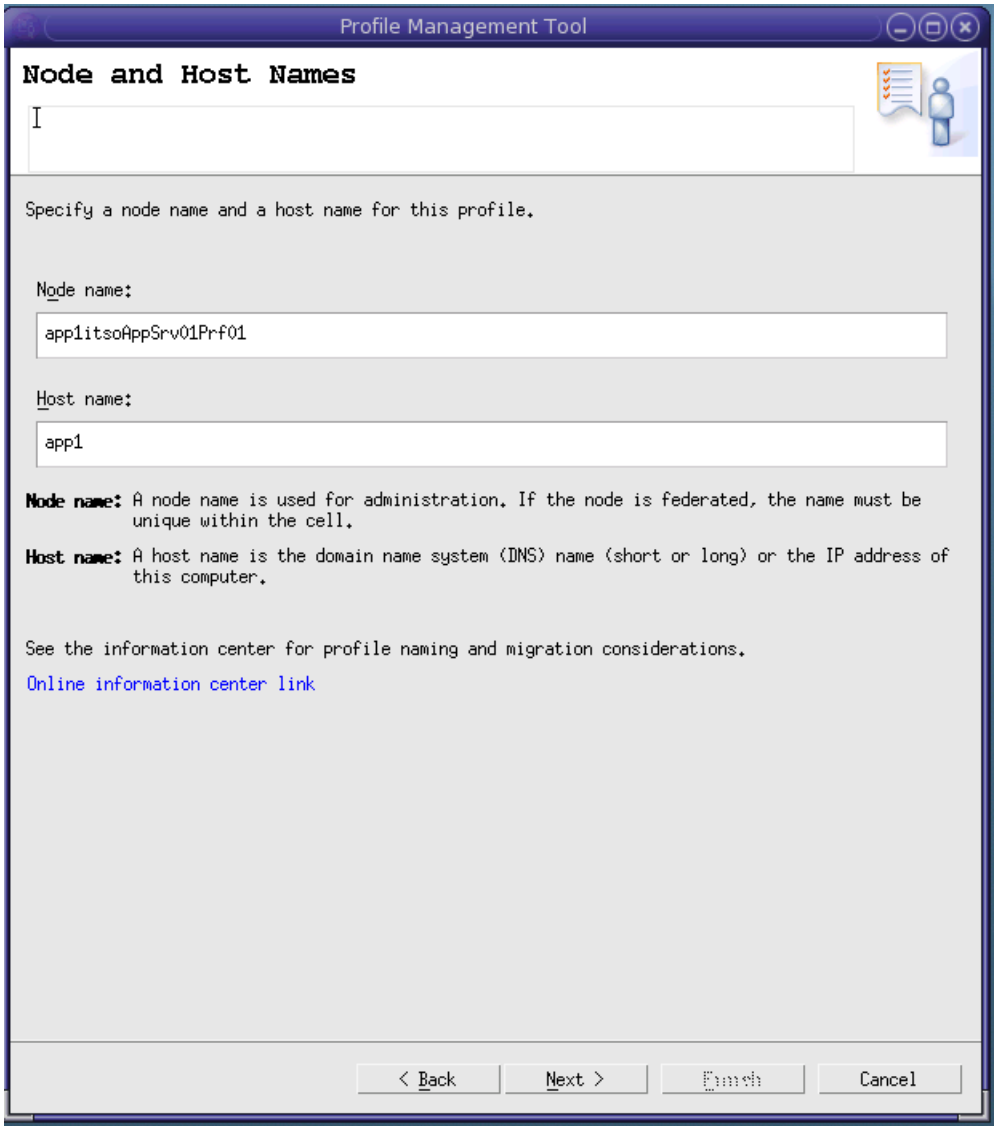


Figure 4-10 Creating an application server profile: Enter host and node names

> **Note:** If you are planning to create multiple stand-alone application servers for federation later to the same cell, make sure you select a unique node name for each application server.

7. Choose whether to enable administrative security. If you enable security here, you will be asked for a user ID and password that will be added to a file-based user registry with the Administrative role. Click **Next**.
8. The wizard will present a list of TCP/IP ports for use by the application server, as shown in Figure 4-11 on page 73. If you already have existing profiles on the system (within this installation), this will be taken into account when the wizard selects the port assignments, but you should verify that these ports will be unique on the system.

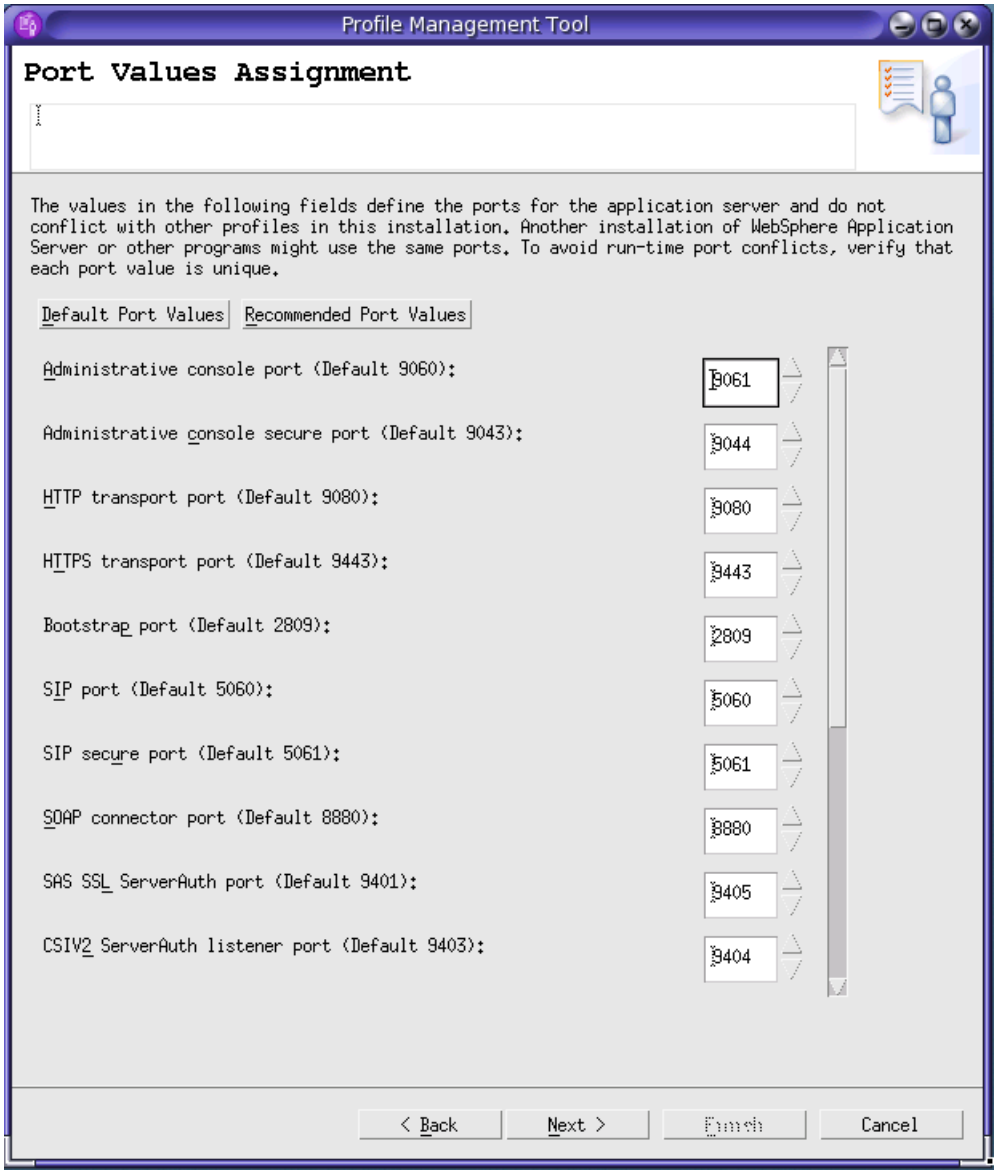


Figure 4-11 Creating an application server profile: Select ports

Important: You might want to note the following ports for later use:

- *SOAP connector port*: If you use the **addNode** command to federate a node to this deployment manager, you need to know this port number. This is also the port you connect to when using the **wsadmin** administration scripting interface.
- *Administrative console port:* You need to know this port in order to access the administrative console. When you turn on security, you need to know the *Administrative console secure port*.

9. The wizard will allow you to create an optional Web server definition, as displayed in Figure 4-12. Web server definitions define an external Web server to WebSphere Application Server. This allows you to manage Web server plug-in configuration files for the Web server and in some cases to manage the Web server. If you have not installed a Web server or wish to do this later, you can easily do this from the administrative console.

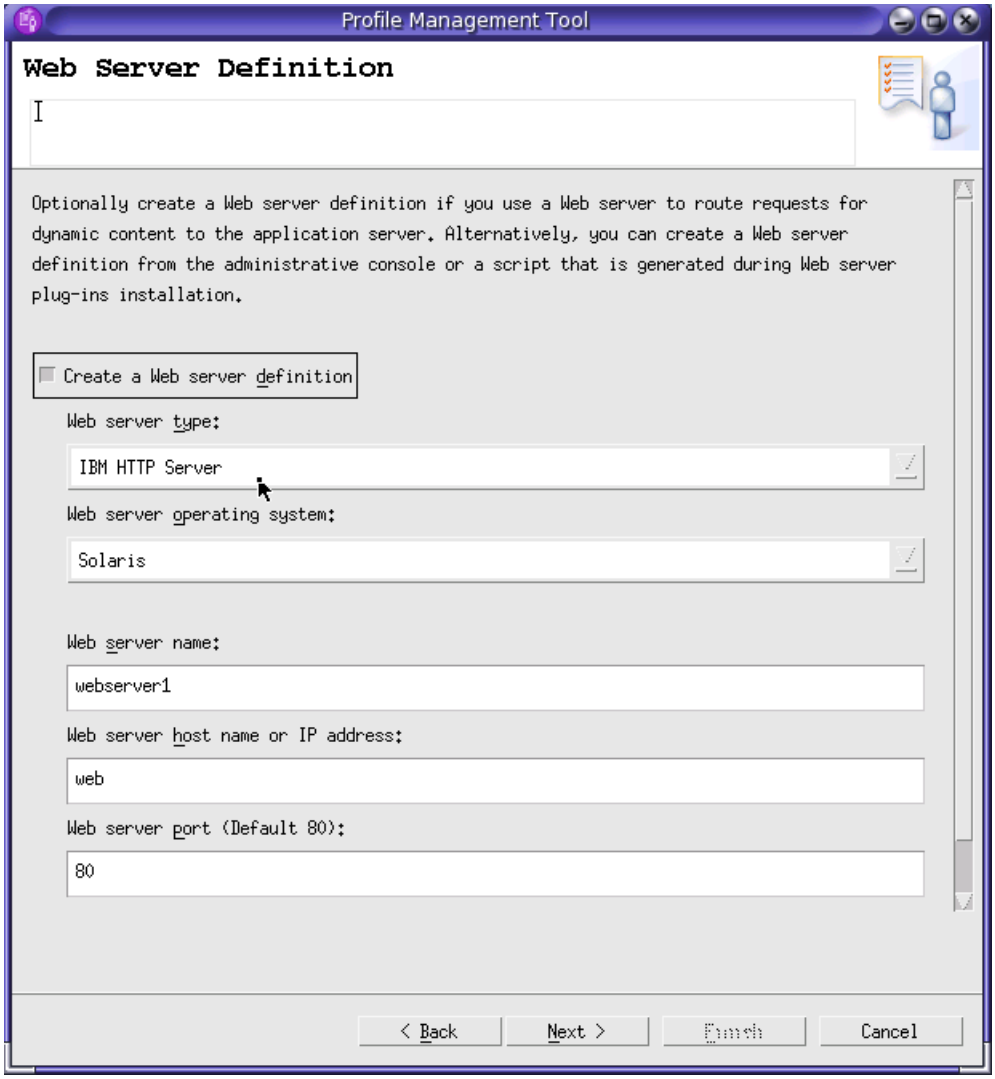


Figure 4-12 Creating an application server profile: Creating a Web server definition

10. Review the options you have chosen and click **Create** to create the profile. See Figure 4-13 on page 75.

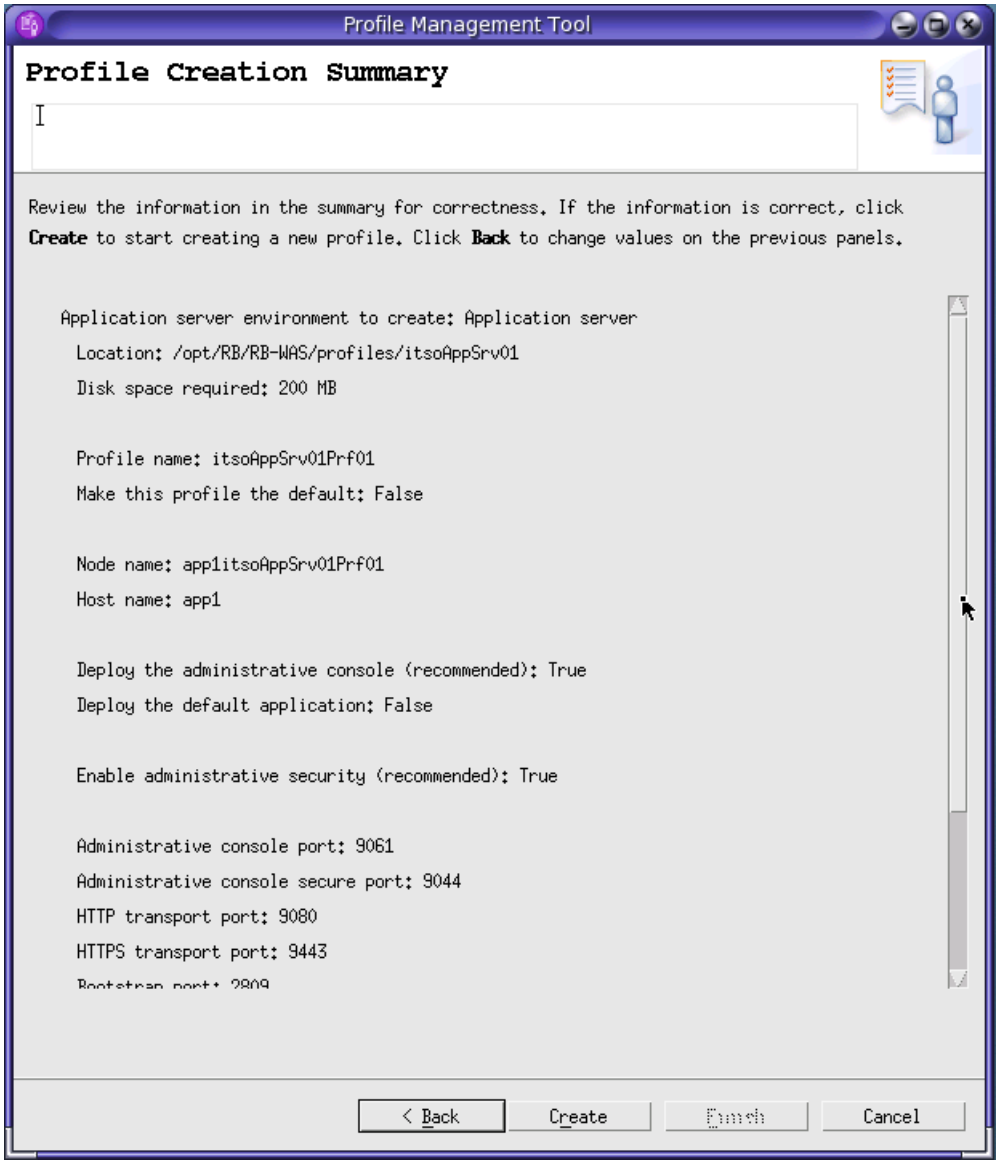


Figure 4-13 Creating an application server profile: Finish

This final window indicates the success or failure of the profile creation.

If you have errors, check the log at:

<was_home>/logs/manageprofiles/<profile_name>_create.log

Note that you will have to click **Finish** on the window to unlock the log.

You will also find logs for individual actions stored in:

<profile_home>/logs

11. Click **Finish** to close the wizard and start the First steps application. See Figure 4-14.

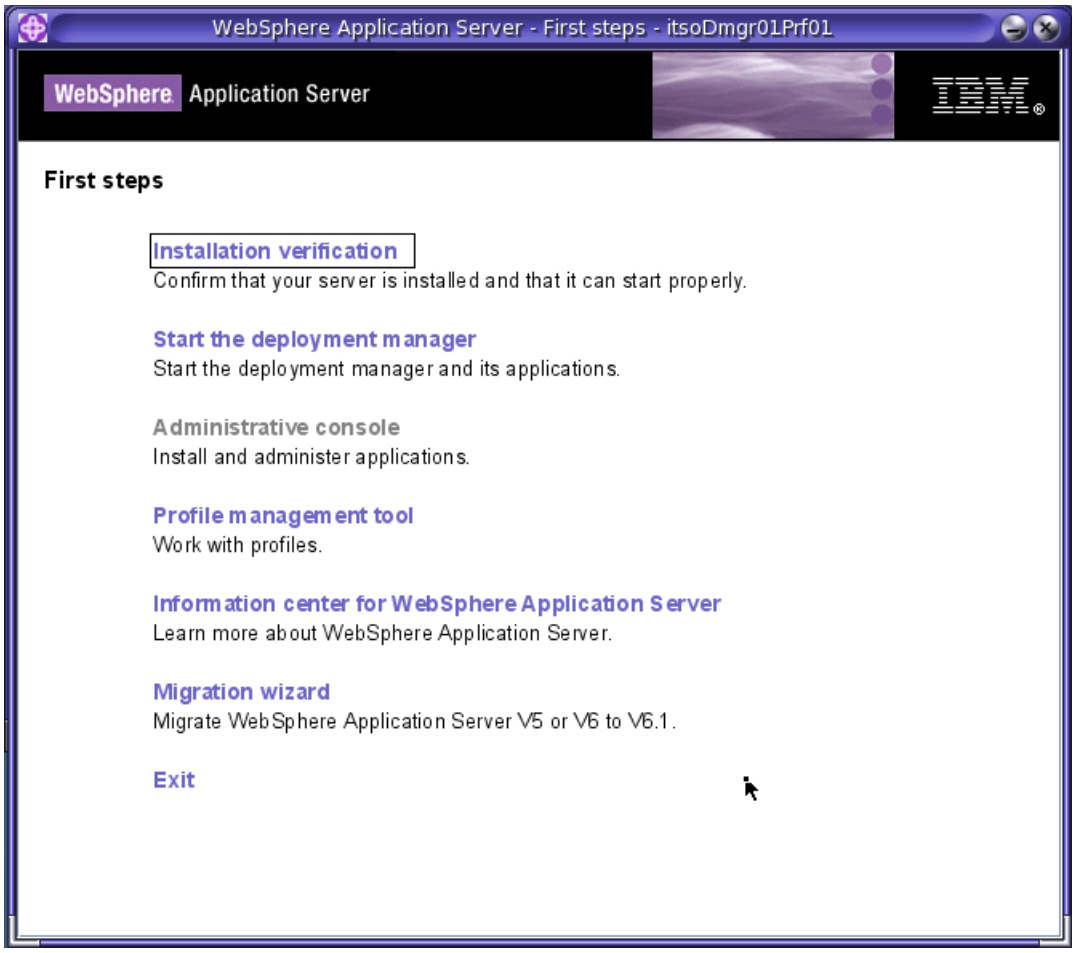


Figure 4-14 Application server First steps menu

Check your results

If the creation was successful, do the following to familiarize yourself with the profile and how to use it:

1. View the directory structure and find the new profile. In this IBM Redbooks publication, we refer to this directory as *<profile_home>*. This is where you will find, among other things, the config directory containing the application server configuration files, the bin directory for entering commands, and the logs directory where information is recorded.
2. Verify the installation. You can do this directly from the First steps menu. This process will start the application server and verify the proper operation of the Web and EJB containers. Messages are displayed on the First steps window and logged in the following places:
   - *<profile_home>*/logs/server1/startServer.log
   - *<profile_home>*/logs/server1/SystemOut.log
3. Start the server. If you ran the installation verification, the server should already be started. You can check it using the following commands:

```
#>cd <profile_home>\bin
#>./serverStatus.sh -all
```

If the server status is not started, then start it from the First steps menu or with the following commands:

```
#>cd <profile_home>\bin
#>./startServer.sh server1
```

4. Open the administrative console, either by selecting the option in the First steps window, or by accessing its URL from a Web browser:

   ```
   http://<appserver_host>:<admin_console_port>/ibm/console
   ```

   Here is a sample URL:

   ```
   http://localhost:9061/ibm/console/
   ```

   Note: The administrative console port of 9061 was selected during the Profile Management Tool (see Figure 4-11 on page 73).

   Click the **Log in** button. If you did not enable security, you do not have to enter a user name. If you choose to enter a name, it can be any name. It is used to track changes you make from the console. If you enabled administrative security, enter the user ID and password you specified.

5. Display the configuration from the console, as shown in Figure 4-15. You should be able to see the following items from the administrative console:

   a. Application servers

      Select **Servers** → **Application servers**. You should see server1. To see the configuration of this server, click the name in the list.

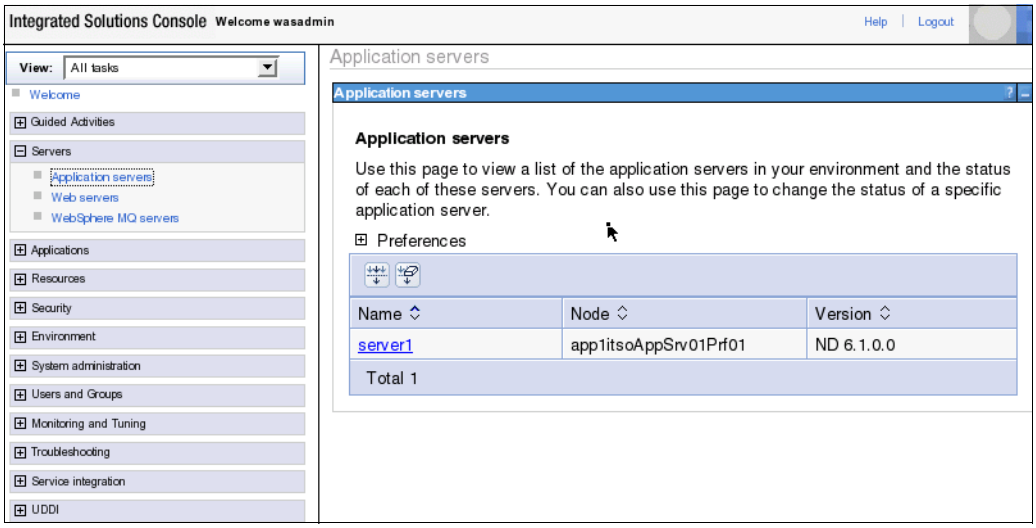


Figure 4-15 Application server defined by the application server profile

b. Enterprise applications

Select **Applications** → **Enterprise Applications**. See Figure 4-16**.** You should see a list of applications. These are the WebSphere sample applications.

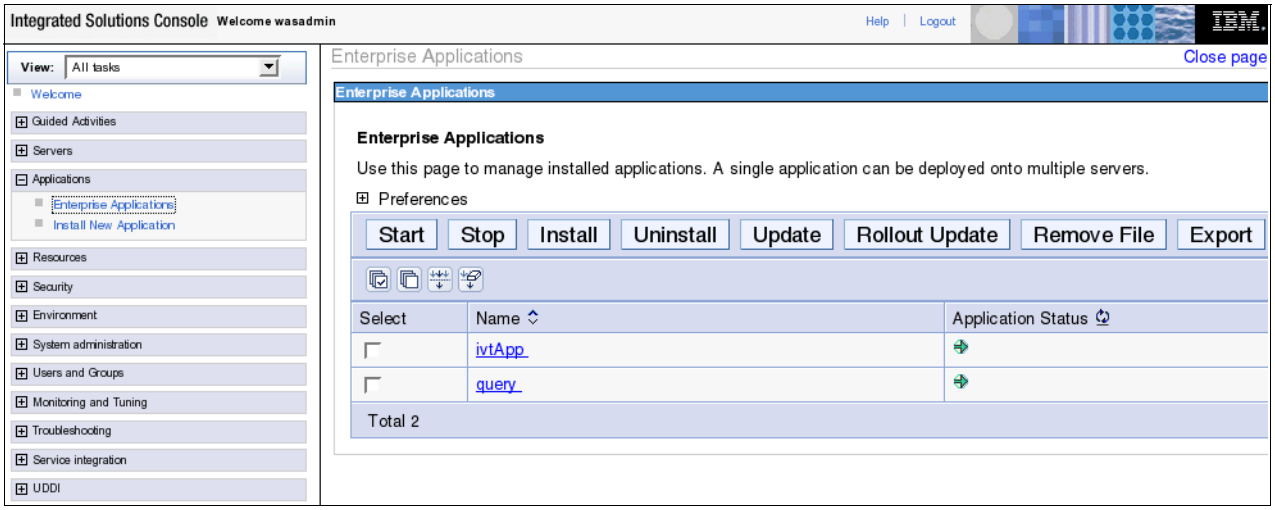


Figure 4-16 Applications installed on server1

Note: Although you cannot display the cell and node from the administrative console, they do exist. You will see this later as you begin to configure resources and choose a scope. You can also see them in the *<profile_home>* /config directory structure.

6. Stop the application server. You can do this from the First steps menu, or better yet, use the **`stopServer`** command:

```
#>cd <profile_home>/bin
#>./stopServer.sh server1 -username <username> -password <password>
```

4.2.5 Creating a cell profile

Table 4-4 shows a summary of the options you have during a cell profile's creation. Using this option actually creates two distinct profiles: a deployment manager profile and an application server profile. The application server profile is federated to the cell. The options you see are a reflection of the options you would see if you were creating the individual profiles versus a cell. The Profile Management Tool windows give you basically the same options that you would see if you created a deployment manager and then an application server.

Table 4-4 Cell profile options

Typical	Advanced
The administrative console and default application are deployed by default. The sample applications are not deployed.	You have the option to deploy the administrative console (recommended), the default application, and the sample applications (if installed).
The profile name for the deployment manager is Dmgr*xx* by default, where *xx* is 01 for the first deployment manager profile and increments for each one created. The profile is stored in *<was_home>*/profiles/Dmgr*xx*.	You can specify the profile name and its location.

Typical	Advanced
The profile name for the federated application server and node is AppSrv*xx* by default, where *xx* is 01 for the first application server profile and increments for each one created. The profile is stored in *<was_home>*/profiles/AppSrv*xx*.	You can specify the profile name and its location.
Neither profile is made the default profile.	You can choose to make the deployment manager profile the default profile.
The cell name is *<host>*Cell*xx*. The node name for the deployment manager is *<host>*CellManager*xx*. The node name for the application server is *<host>*Node*xx*. The host name is pre-filled in with your system's DNS host name.	You can specify the cell name, the host name, and the profile names for both profiles.
You can enable administrative security (yes or no). If you select yes, you will be asked to specify a user name and password that will be given administrative authority.	
TCP/IP ports will default to a set of ports not used by any profiles in this WebSphere installation instance.	You can use the recommended ports for each profile (unique to the installation), use the basic defaults, or select port numbers manually.
Does not create a Web server definition.	Allows you to define an external Web server to the configuration.

4.2.6 Creating a custom profile

A custom profile defines an empty node on a system. The purpose of this profile is to define a node on a system to be federated to a cell for central management.

As you create the profile, you will have the option to federate the node to a cell during the wizard, or to simply create the profile for later federation. Before you can federate the custom profile to a cell, you will need to have a running deployment manager.

Table 4-5 shows a summary of the options you have during profile creation for a a custom node.

Table 4-5 Custom profile options

Typical	Advanced
The profile name is Custom*xx*. The profile is stored in *<was_home>*/profiles/Custom*xx*. By default, it is not considered the default profile.	You can specify profile name and location. You can also specify if you want this to be the default profile.
The node name is *<host>*Node*xx*. The host name is pre-filled in with your system's DNS host name.	You can specify the node name and host name.
You can opt to federate the node later, or during the profile creation process. If you want to do it now, you have to specify the deployment manager host and SOAP port (by default, localhost:8879). If security is enabled on the deployment manager, you will need to specify a user ID and password.	
TCP/IP ports will default to a set of ports not used by any profiles in this WebSphere installation instance.	You can use the recommended ports for each profile (unique to the installation), use the basic defaults, or select port numbers manually.

This section takes you through the steps of creating a custom profile.

1. Start the Profile Management Tool. Click **Next** on the Welcome page.
2. Select the **Custom profile** option. Click **Next**.
3. Select the kind of creation process you want to run, Typical or Advanced. Click **Next**.
   - If Typical is selected, then you will be sent directly to the option to federate (Figure 4-19 on page 82).
   - If Advanced is selected, you will see the next step.
4. Enter a unique name for the profile or accept the default. The profile name will become the directory name for the profile files. See Figure 4-17.

   Click the box if you want this directory to be the default profile for receiving commands. Click **Next**.

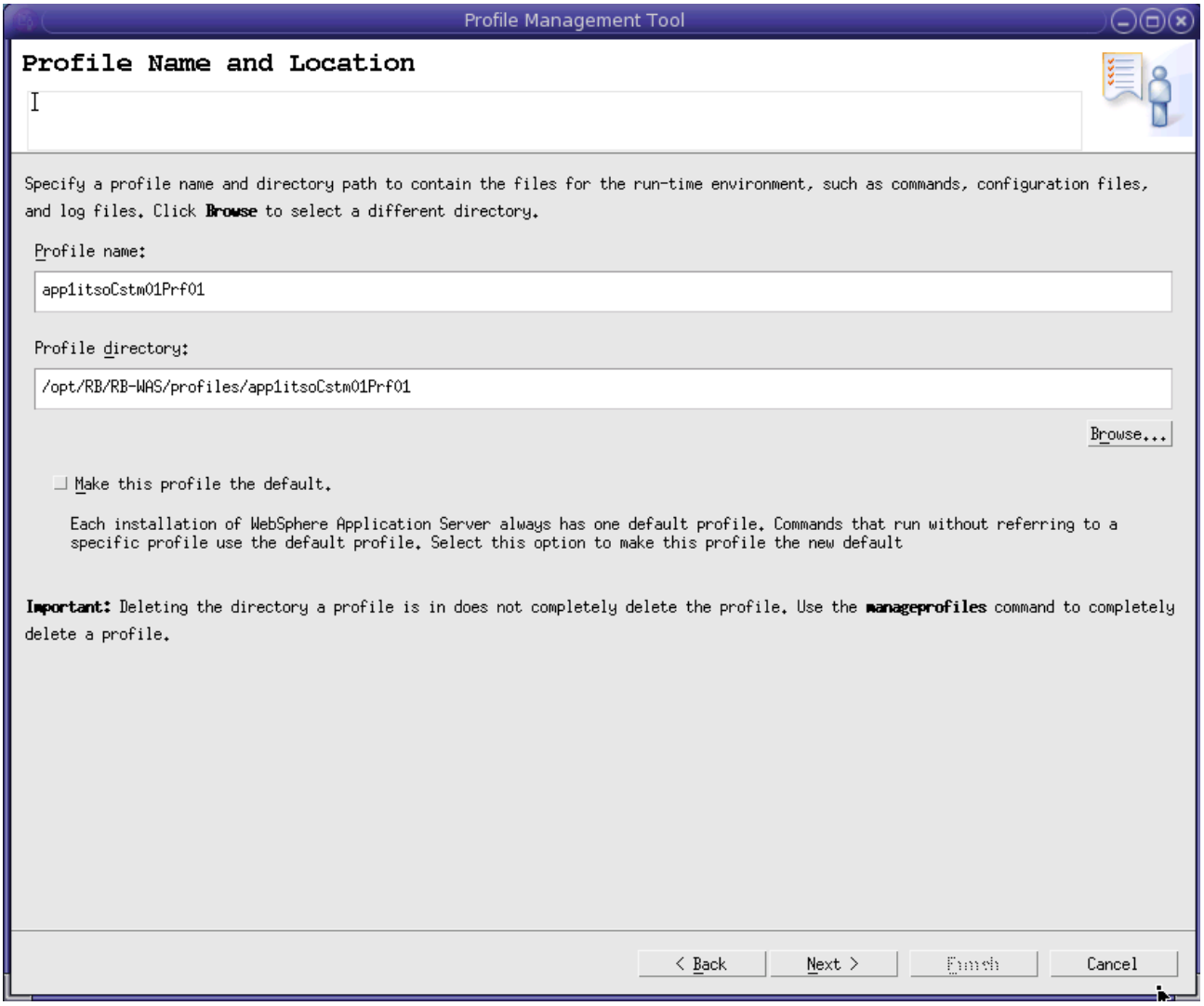


Figure 4-17 Creating a Custom profile: Enter name and location

5. Enter the new node name and the system host name (see Figure 4-18). The node name defaults to the host name of your system. The wizard recognizes if there are existing nodes in the installation and takes this into account when creating the default node name. Click **Next**.

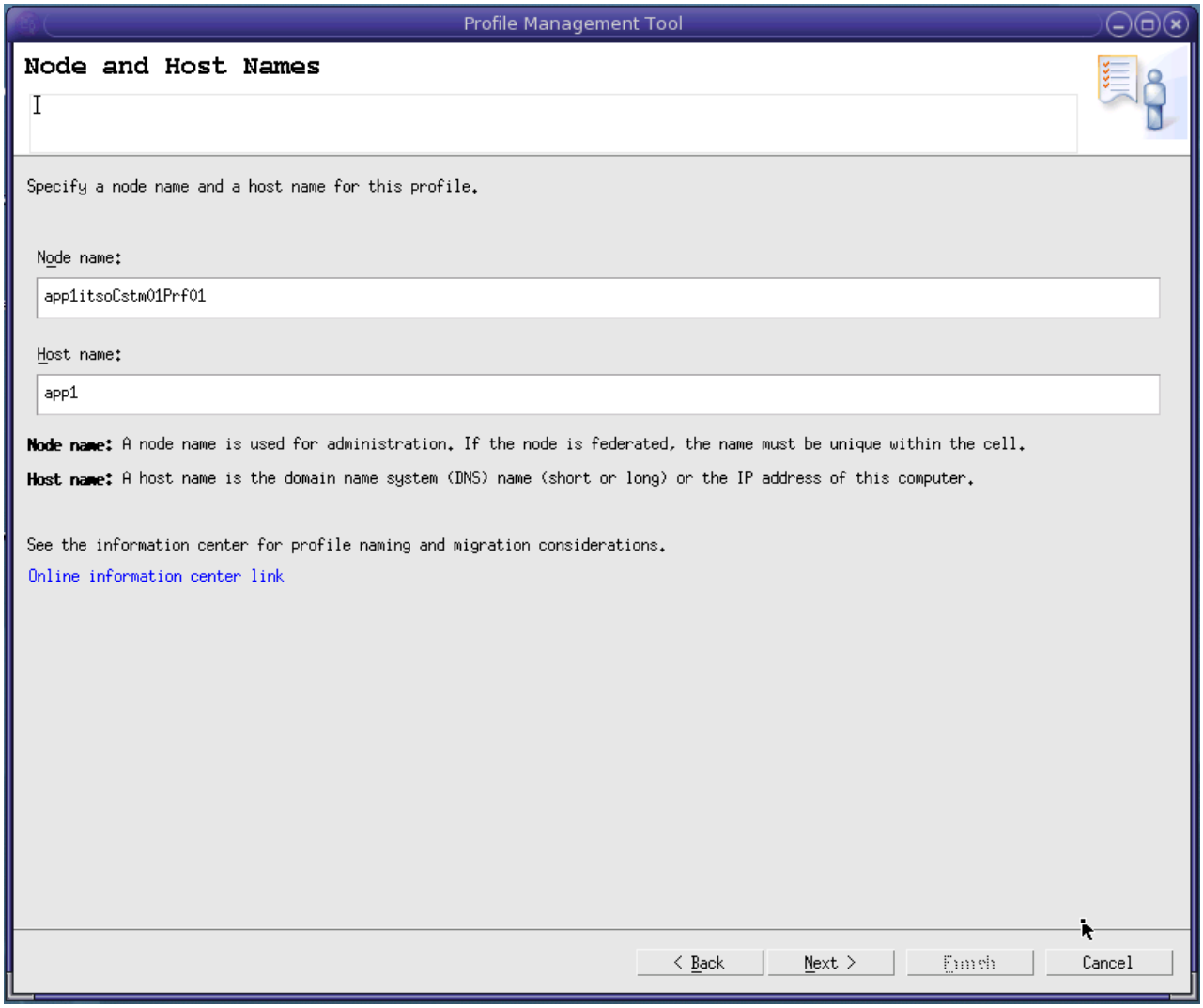


Figure 4-18 Creating a custom profile: Enter host, and node names

If you would like to federate, or add, the new node defined by the profile to a cell as part of the wizard process, leave the **Federate this node later** box unchecked and enter the host name and SOAP connector port (Figure 4-6 on page 66) for the deployment manager. See Figure 4-19.

Note: If you choose to federate now, make sure the deployment manager is started.

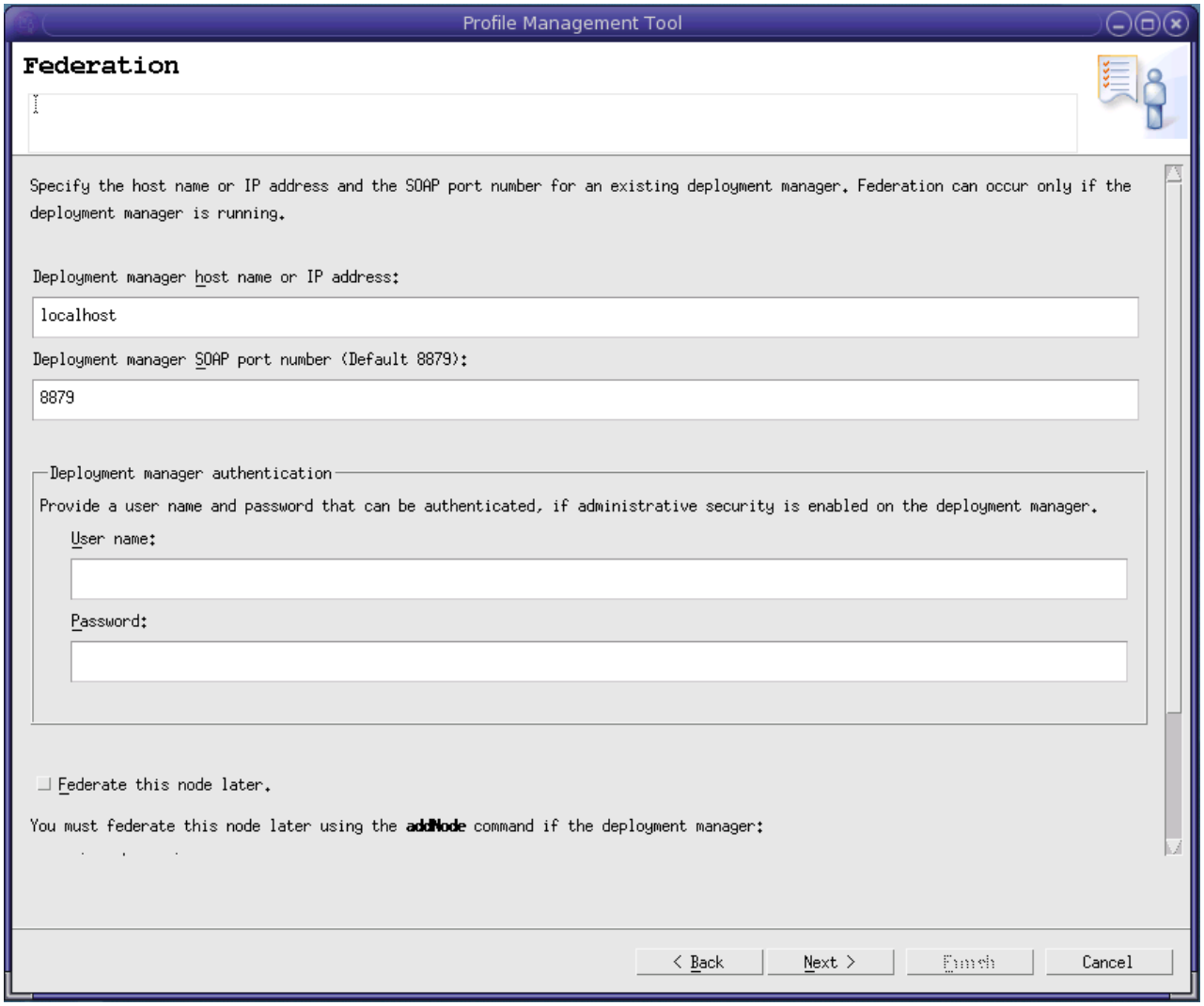


Figure 4-19 Creating a custom profile: Federate now or later

6. Review the options you have chosen, as shown in Figure 4-20 on page 83.

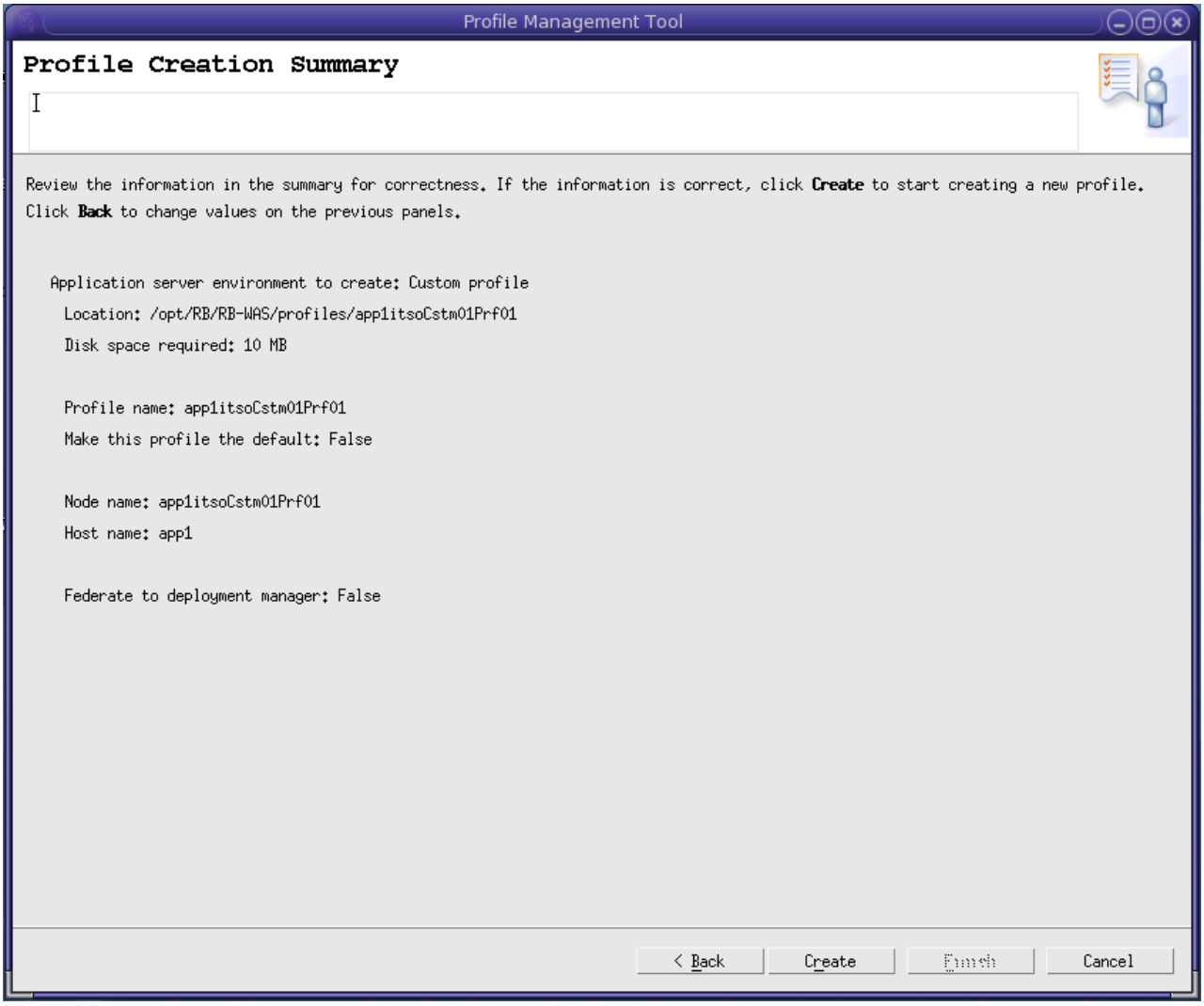


Figure 4-20 Creating a custom profile: Summary

Click **Next** to create the profile.

This final window indicates the success or failure of the Custom profile creation.

If you have errors, check the log at:

<was_home>/logs/manageprofiles/*<profile_name>*_create.log

Note that you will have to click **Finish** on the window to unlock the log.

You will also find logs for individual actions stored in *<profile_home>*/logs.

7. After the wizard has finished, you will be presented with a window containing messages indicating the success or failure of the process. You can launch First steps if you want or even create another profile. If you have errors, check the log at *<was_home>*/logs/manageprofiles/*<profile_name>*_create.log.

   Note that you will have to click **Finish** on the window to unlock the log.

   You will also find logs for individual actions stored in *<profile_home>*/logs.

8. Click **Finish** to close the wizard and start the First steps application if you desire, as shown in Figure 4-21.

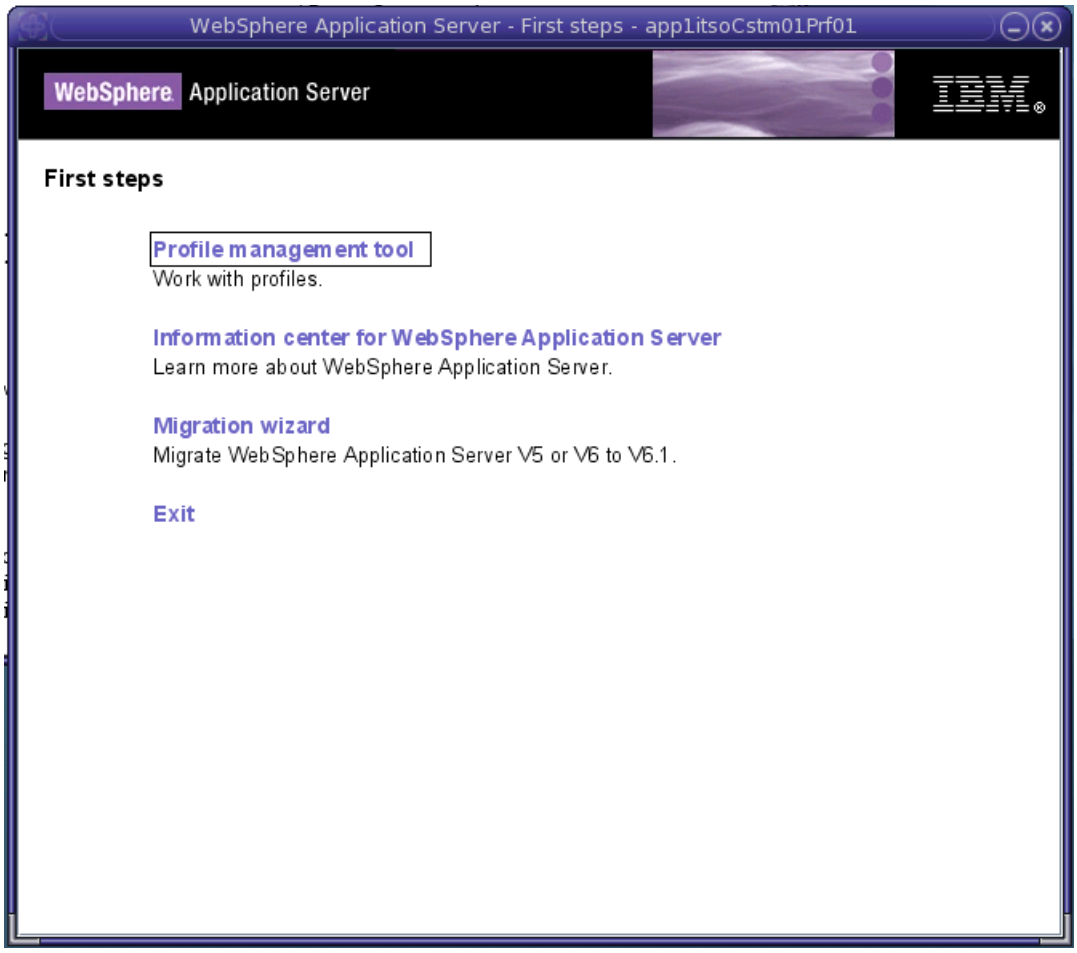


Figure 4-21 Custom profile First steps window

Checking your results

If the creation was successful, do the following to familiarize yourself with the profile and how to use it:

1. View the *<profile_home>* directory structure and find the new profile. This is where you will find, among other things, the config directory containing the node configuration files.
2. If you federated the custom profile, open the deployment manager administrative console and view the node and node agent:
   - Select **System Administration** → **Nodes**. You should see the new node.
   - Select **System Administration** → **Node agents**. You should see the new node agent.
   - Select **System Administration** → **Cells**. Click the **Topology** tab and expand the view. From here, you can see a tree diagram of the cell.
3. The federation process creates a node agent for the new node, federates it to the cell, and starts the node agent.

   You can stop the new node agent from the console or with the following commands on the node system:

   ```
   cd <profile_home>\bin
   ./stopNode.sh -username <username> -password <password>
   ```

While you can restart a node agent from the administrative console, you cannot start a node that has been stopped. To start the new node agent, use the following commands on the node system:

```
#>cd <profile_home>\bin
#>./startNode.sh
```

If you have not federated the node, you will not be able to start it yet. Proceed to 4.2.7, "Federating a custom node to a cell" on page 85. Otherwise, you can continue by defining an application server on the new node. To do this, refer to 4.2.8, "Creating a new application server on an existing node" on page 85.

4.2.7 Federating a custom node to a cell

Note: You only have to do this if you created a custom profile and chose *not* to federate it at the time. This requires that you have a deployment manager profile and that the deployment manager is up and running.

An custom profile is used to define a node that can be added to a cell. To federate the node to the cell, do the following:

1. Start the deployment manager.
2. Open a command window on the system where you created the custom profile for the new node. Switch to the *<profile_home>*/bin directory (for example, **cd \RBProfiles\cstmProfiles\CstmPrf01**).
3. Run the **addNode** command. Here you need the host name of the deployment manager and the SOAP connector address (see Figure 4-4 on page 64 and Figure 4-6 on page 66):

   ```
   addNode <dmgrhost> <dmgr_soap_port>
   ```

4. Open the deployment manager administrative console and view the node and node agent:
   - Select **System Administration** → **Nodes**. You should see the new node.
   - Select **System Administration** → **Node agents**. You should see the new node agent and its status. It should be started. If not, check the status from a command window on the custom node system:

     ```
     cd <profile_home>\bin
     ./serverStatus.sh -all
     ```

     If you find that it is not started, start it with this command:

     ```
     cd <profile_home>\bin
     ./startNode.sh
     ```

4.2.8 Creating a new application server on an existing node

The custom profile does not automatically give you an application server. You can follow these steps to create a new server once the custom profile has been federated to a cell:

1. Ensure the custom profile node agent is started.
2. Open the deployment manager administrative console.
3. Select **Servers** → **Application Servers**.
4. Click **New**.

5. Select the custom profile node and enter a new name for the server (Figure 4-22). Click **Next**.

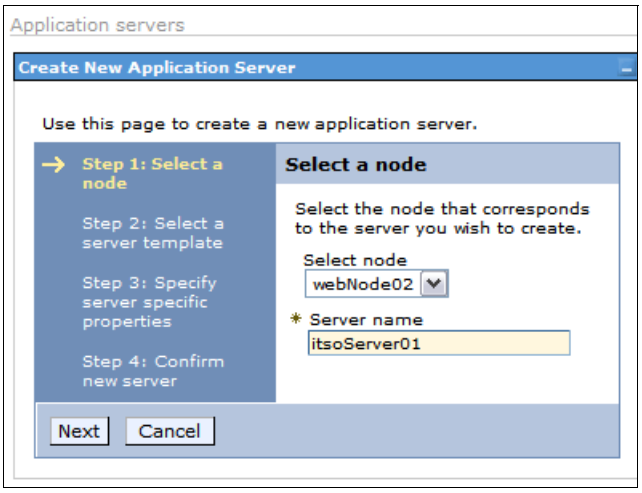


Figure 4-22 Creating a new server: Enter a node and name

6. Select a template to use as a basis for the new application server configuration. See Figure 4-23.

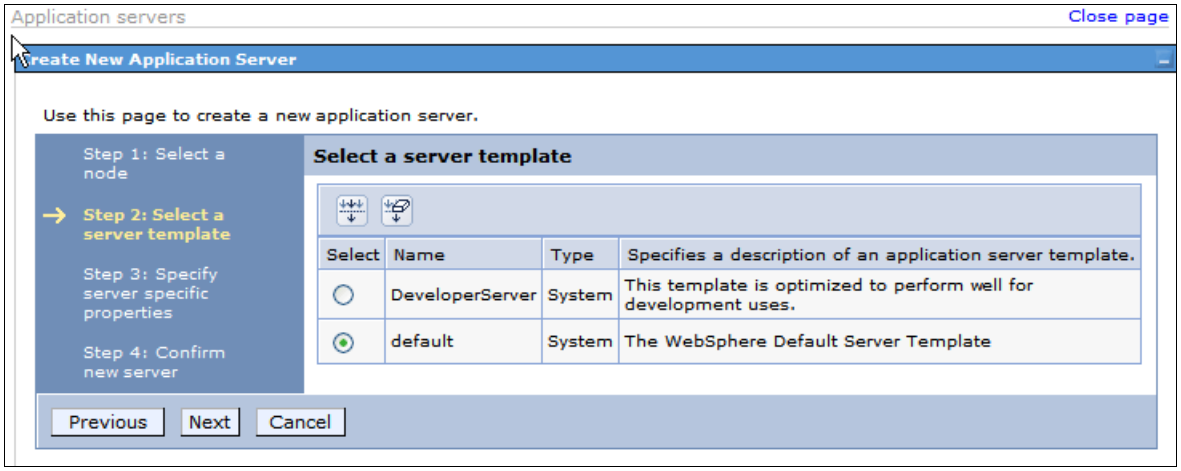


Figure 4-23 Creating a new server: Select a template

The DeveloperServer and default templates have been created for you. The default template is used to create a typical server for production.

New in V6.1: The DeveloperServer template is used to create a server optimized for development. It turns off PMI and sets the JVM into a mode that disables class verification and allows it to startup faster through the **`-Xquickstart`** command. Note that it does not enable the “developmentMode" configuration property (run in development mode setting on the application server window). If you would like to set this to speed up the application server startup, you will need to configure it after server creation using the administrative console.

You can also create templates based on existing servers. If you have not previously set up a template based on an existing application server, select the default template. Click **Next**.

7. Each application server on a node must have unique ports assigned. Figure 4-24 shows you the option of having unique ports generated for this application server, as opposed to the default set. Click **Next**.

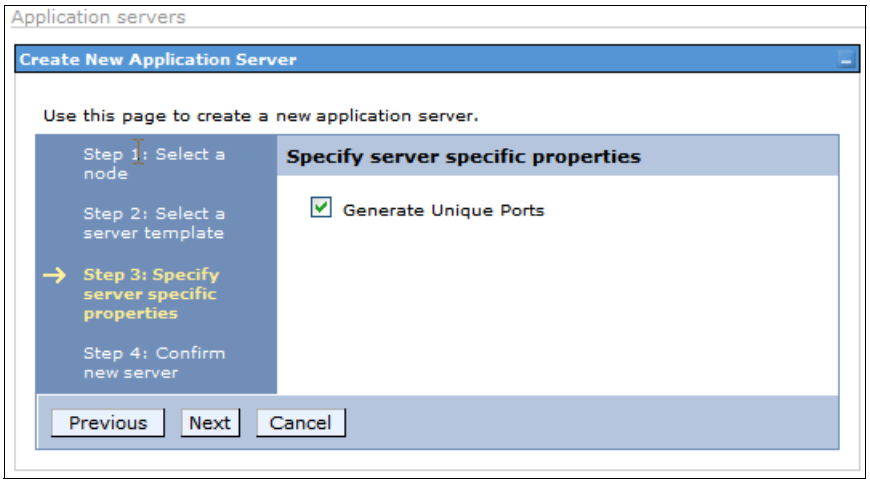


Figure 4-24 Creating a new server: Generate unique ports

8. The last window summarizes your choices. See Figure 4-25. Click **Finish** to create the profile.

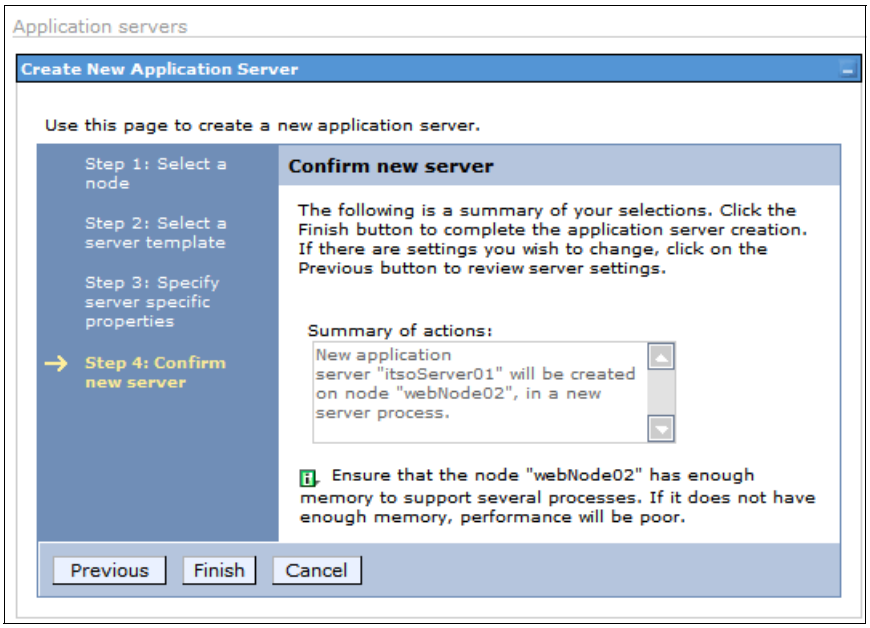


Figure 4-25 Creating a new server: Summary and finish

9. In the messages box, click **Save** to save the changes to the master configuration.
10. Start the application server from the administrative console.
    - Select **Servers** → **Application Servers.**
    - Check the box to the left of the server and click **Start**.

Note: WebSphere Application Server provides sample applications that you can use to familiarize yourself with WebSphere applications. These samples can be installed (optional) when you create an application server profile. If you create an application server from the administrative tools, you will not get the samples installed automatically. For information about the samples available and how to install them, see the Accessing the Samples topic under *Learn about WebSphere Applications* in the Information Center.

4.2.9 Federating an application server profile to a cell

If you created an application server profile and now want to add the node and server to the cell, do the following:

1. Start the application server.
2. Start the deployment manager.
3. Open the deployment manager administrative console.
4. Select **System Administration** → **Nodes**.
5. Click **Add Node**.
6. Select **Managed node** and click **Next**.
7. Enter the host name and SOAP connector port specified when you created the application server profile. See Figure 4-10 on page 72 and Figure 4-11 on page 73.

   If you want to keep the sample applications and any other applications you have installed, check the **Include applications** box. If this is a newly created application server profile, it will contain the sample applications, so be sure to check this box if you want to keep the samples.

   If you have created a service integration bus on the server, you can opt to have it included in the managed server as well. By default, you do not have a service integration bus in a newly created application profile. If you have created a bus, and choose to include it, the name must be unique in the cell, as shown in Figure 4-26 on page 89.

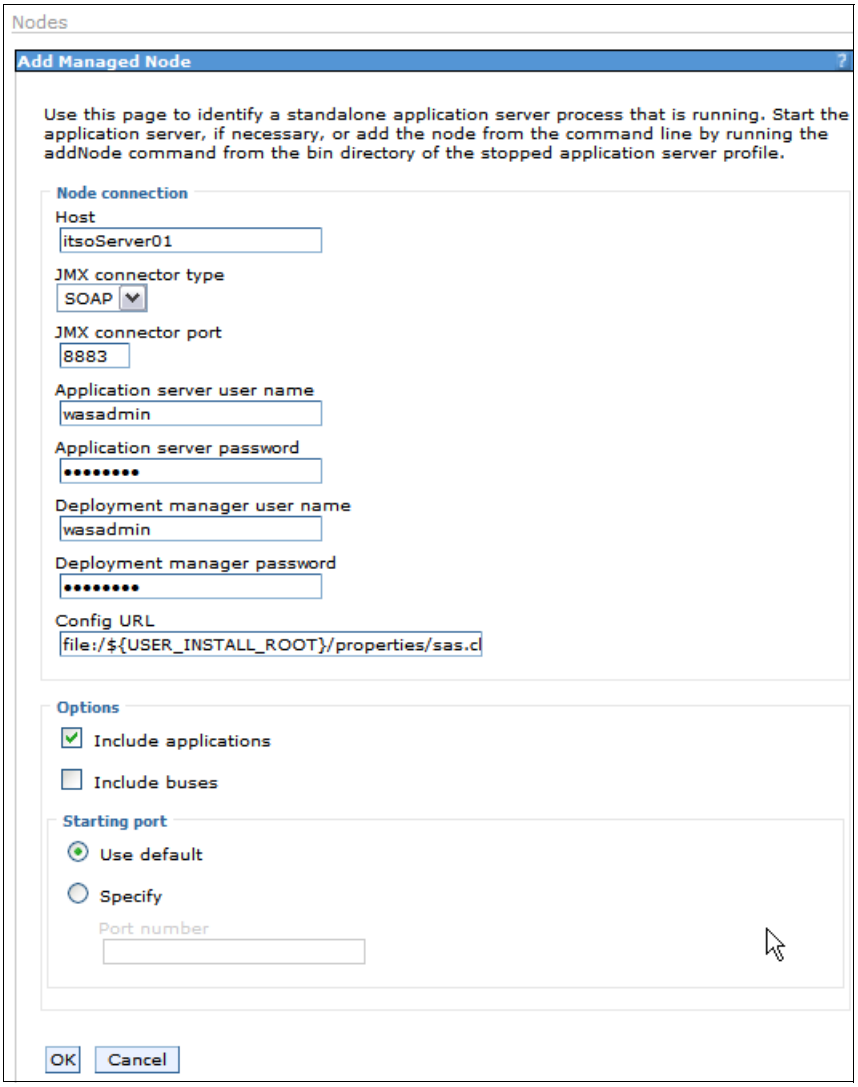


Figure 4-26 Adding a standalone application profile to a cell

Click **OK**.

The federation process stops the application server. It creates a new node agent for the node, and adds the node to the cell. The application server becomes a managed server in the cell. It then starts the node agent, but not the server.

8. You can now display the new node, node agent, and application server from the console. You can also start the server from the console.

At the completion of the process:

- The profile directory for the application server still exists and is used for the application server.
- The old cell name for the application server has been replaced with a profile directory with the cell name of the deployment manager:

  <profile_home>/config/cells/*<dmgr_cellname>/*
- A new entry in the deployment manager profile directory has been added for the new node:

  <dmgr_profile_home>/config/cells/*<dmgr_cellname>*/nodes/*<federated node>*

- An entry for each node in the cell is added to the application server profile configuration. Each node entry contains the serverindex.xml file for the node:

  <profile_home>/config/cells/*<dmgr_cellname>*/nodes/*<federated node>*

In turn, an entry for the new node is added to the nodes directory for each node in the cell with a serverindex.xml entry for the new node.

4.2.10 Default security

The Profile Management Tool provides the option to enable security out of the box. Enabling administrative security out of the box in a production environment will prevent any user from gaining unauthorized access to the administrative tools. This new feature gives you a simple default security setting out of the box, so you do not have to deal with the complexity of setting up basic administrative security.

The Profile Management Tool uses the simple File-Based user registry. If you use Local OS, LDAP, or custom registry for security, you can disable security and use the Administrative console or scripts to enable security after completing the installation.

LTPA will serve as the default authentication mechanism. An internal server identification for interprocess communications will be automatically generated. An administrative user and password will be added to the file-based user registry, which will exist under the profile root for each profile, for example *<profile_home>*/config/cells/*<cellname>/fileRegistry.xml*.

4.3 Managing profiles

Each profile you create is registered in a profile registry:

<was_home>/properties/profileRegistry.xml

You have already seen how profiles are created with the Profile Management Tool. At the heart of this wizard is the **manageprofiles** command. This command provides you with the means to do normal maintenance activities for profiles. For example, you can call this command to create profiles natively or silently, list profiles, delete profiles, validate the profile registry, and other functions.

4.3.1 Using the manageprofiles command

The **manageprofiles** command can be found in the *<was_home>*/bin directory.

Syntax

Use the following syntax for the **manageprofiles** command:

```
manageprofiles.sh -mode -arguments
```

The modes in Table 4-6 on page 91 are available.

Table 4-6 manageprofiles modes

Mode	Use
-create	Creates a new profile.
-augment	Augments the given profile using the given profile template.
-delete	Deletes a profile.
-unaugment	Unaugments the profile.
-deleteAll	Deletes all registered profiles.
-listProfile	Lists the profiles in the profile registry.
-getName	Returns the name of the profile at the path specified.
-getPath	Returns the path of the profile name specified.
-validateRegistry	Validates the profile registry and returns a list of profiles that are not valid.
-validateAndUpdateRegistry	Validates the profile registry and lists the non-valid profiles that it purges.
-getDefaultName	Returns the name of the default profile.
-setDefaultName	Sets the default profile.
-backupProfile	Back ups the given profile into a zip file.
-restoreProfile	Restores the given profile from a zip file.
-response	Manage profiles from a response file.
-help	Shows help.

The following two examples show the results of the **manageprofiles -<mode> - help** and **manageprofiles -listProfiles** modes:

- Enter **manageprofiles -<mode> -help** for detailed help on each mode. See Example 4-3 for an example of the **manageprofiles -create -help** command.

Example 4-3 Getting help for the manageprofiles command

```
# /opt/IBM/Websphere/AppServer/bin>./manageprofiles.sh -create -help
The following command line arguments are required for this mode.
Command-line arguments are case sensitive.
-create: Creates a new profile. Specify -help -create -templatePath <path> to get
template-specific help information.
-templatePath: The fully qualified path name of the profile template that is
located on the file system. The following example selects a template:
-templatePath <app_server_home>/profileTemplates/<Template_name>
-profileName: The name of the profile.
-profilePath: The intended location of the profile in the file system.
The following command line arguments are optional, and have no default values.
Command-line arguments are case sensitive.
-isDefault: Make this profile the default target of commands that do not use their
profile parameter.
```

- Enter **manageprofiles -listProfiles** to see a list of the profiles in the registry. The following is a sample output of -listProfiles:

```
# /was_home/bin./manageprofiles -listProfiles
[itsoDmgr01, itsoAppSrv01]
```

4.3.2 Creating a profile

You can use the **manageprofiles** command to create profiles instead of using the Profile Management Tool.

Profile templates: The profiles are created based on templates supplied with the product. These templates are located in *<was_home>*/profileTemplates. Each template consists of a set of files that provide the initial settings for the profile and a list of actions to perform after the profile is created. Currently, there is no provision for modifying these templates for your use, or for creating new templates. When you create a profile using **manageprofiles**, you will need to specify one of the following templates:

- default (for application server profiles)
- dmgr (for deployment manager profiles)
- managed (for custom profiles)
- cell (for cell profiles)

For example, Example 4-4 shows the commands used to create an application server named itsoServer1 on node itsoNode1 in cell itsoCell1 on host app1 from the command line.

Example 4-4 Creating a profile with the manageprofiles command

```
# cd /opt/IBM/Websphere/AppServer/bin
# ./manageprofiles.sh -create -profileName itsoAppServer1 -profilePath
/opt/IBM/Websphere/Appserver/profiles/AppSrvrProfiles/itsoServer1 -templatePath
/opt/IBM/Websphere/AppServer/profileTemplates/default -nodeName itsoNode1
-cellName itsoCell1 -hostName app1
```

4.3.3 Deleting profiles

To delete a profile, you should do the following:

- If you are removing a custom profile or application server profile that has been federated to a cell:
  - Stop the application servers on the node.
  - Remove the node from the cell using the administrative console or the **removeNode** command. Removing a node does not delete it, but restores it to its pre-federated configuration that was saved as part of the federation process.
  - Delete the profile using the **manageprofiles -delete** command.
  - Use the **manageprofiles -validateAndUpdateRegistry** command to clean the profile registry.
  - Delete the *<profile_home>* directory.
- If you are removing an application server profile that has not been federated to a cell:
  - Stop the application server.
  - Delete the profile using the **manageprofiles -delete** command.
  - Use the **manageprofiles -validateAndUpdateRegistry** command to clean the profile registry.
  - Delete the *<profile_home>* directory.

- If you are removing a deployment manager profile:
  - Remove any nodes federated to the cell using the administrative console or the **removeNode** command. Removing a node does not delete it, but restores it to its pre-federated configuration that was saved as part of the federation process.
  - Stop the deployment manager.
  - Delete the profile using the **manageprofiles -delete** command.
  - Use the **manageprofiles -validateAndUpdateRegistry** command to clean the profile registry.
  - Delete the *<profile_home>* directory.

Deleting a profile with manageprofiles

To delete a profile, use the **manageprofiles -delete** command. Use the following syntax:

```
./manageprofiles.sh -delete -profileName <profile>
```

At the completion of the command, the profile will be removed from the profile registry, and the runtime components will be removed from the *<profile_home>* directory with the exception of the log files.

If you have errors while deleting the profile, check the following log:

<was_home>/logs/manageprofile/*<profile_name>*_delete.log

For example, in Example 4-5, you can see the use of the **manageprofiles** command to delete the profile named itsoNode1.

Example 4-5 Deleting a profile using manageprofiles

```
# /opt/IBM/Websphere/AppServer/profiles/itsoDmgr01/bin> ./manageprofiles -delete
-profileName itsoNode1
INSTCONFSUCCESS: Success: The profile no longer exists.
```

As you can see in Example 4-5, all seems to have gone well. But, as an additional step to ensure the registry was properly updated, you can list the profiles to ensure the profile is gone from the registry and validate the registry.

Note: If there are problems during the deletion, you can manually delete the profile. For more information about this topic, refer to *WebSphere Application Server V6.1 installation problem determination,* REDP-4305.

4.4 Installation Factory

IBM Installation Factory is a tool that is used to create Custom Installation Packages (CIPs) and Integrated Installation Factories (IIPs). It has a user interface (ifgui) and a command-line utility (ifcli). The result of using the Installation Factory is the creation of CIP or IIP, and a build definition that can be reused.

Using Installation Factory GUI (ifgui), users can create CIPs/IIPs in Connected mode or Disconnected mode. Connected mode is used when all input is available on the local system, so that users can browse and select. Disconnected mode is used when input is not available on the local system, such as when users try to create a build definition file for Linux on

Windows. When Disconnected mode is used, users will have to move the build definition file to the target system and run **`ifcli`** there to generate the CIP or IIP there.

4.4.1 Installation Factory overview

The existing process for installing, updating, and configuring a WebSphere Application Server installation can be very time consuming and repetitive. The process includes installing WebSphere Application Server and then updating the installation with refresh packs, fix packs and interim fixes as appropriate. Once WebSphere Application Server has been updated to the appropriate version, profiles need to be created and configured before applications can be properly installed and tuned. Then, depending on the environment and number of systems, this process must be repeated to create multiple installations.

Independent software vendors, customers, and even other IBM products can all benefit from an easier way to quickly install and update WebSphere Application Server.

The Installation Factory feature provides an automated method for creating custom installation packages or Customized Installation Packages for WebSphere Application Server. The WebSphere Application Server installation that is created is the same robust installer normally used, supporting the same silent install options as well. The custom installer can also be used to update an existing installation of WebSphere Application Server, providing an easier alternative to installing multiple fix packs.

Features of the Installation Factory

The Installation Factory has a number of significant features that can be combined to create some interesting capabilities:

- Can be used to create a custom service refresh.
  - An installation image that can install WebSphere Application Server at any level of maintenance.
  - Fastest, smallest, easiest, and most reliable way to install WebSphere Application Server at a given level of maintenance.
  - The installed maintenance packages are registered in the usual way, and are easily discoverable and removable.
- Custom installation package may contain a configuration archive.
  - Contains one or more configuration entities (cell, node, server, and application) to be automatically restored into a new profile, with appropriate changes to make it relocatable.
  - Provides an easy way to restore a predefined configuration, including deployed applications during the installation.
  - Scripts can also be included and run to further configure the instance.

The above features provide the capability to create a custom install package that can replicate an existing installation of WebSphere Application Server.

Note: A number of new features have been introduced for the Installation Factory in WebSphere Application Server V6.1. The first big change is the ability to do cross-platform generation of Customized Installation Packages. It also introduces 64-bit operating system support.

Custom Installation Packages (CIPs)

Customized Installation Packages (CIPs) are WebSphere Application Server or Feature Pack for Web Services installation images with pre-applied maintenances, fix packs, interim fixes, and a set of customizations to profiles, and so on. CIPs make installing WebSphere Application Server or Feature Pack for Web Services more convenient because one installation can bring the system to the required level. Users do not need to go through multiple steps involving the Update Installer.

CIPs can be created based on product. A WebSphere Application Server CIP can include WebSphere Application Server fix packs, SDK fix packs, WebSphere Application Server interim fixes, profile customizations, and additional user files. A Feature Pack for Web Services CIP can include Feature Pack for Web Services fix packs, Feature Pack for Web Services interim fixes, profile customizations, and additional user files. Users cannot include a WebSphere Application Service Fix Pack in the Feature Pack for Web Services CIP. CIP is a vertical bundle of a single product, while IIP is a horizontal bundle of multiple products.

Using Custom Installation Packages for update installations

As mentioned previously, a Custom Installation Package can also be used to upgrade an existing installation of WebSphere Application Server. In the example illustrated in Figure 4-27, a CIP has been built that contains WebSphere Application Server V6.1, the Fix Pack for V6.1.0.11, an upgrade for the Java SDK and 2 i-Fixes. When this custom install package is installed on the various systems shown in the figure, they will be upgraded to the same maintenance level as the custom install package. The figure illustrates that the custom installer will only install what is needed to upgrade the installations.

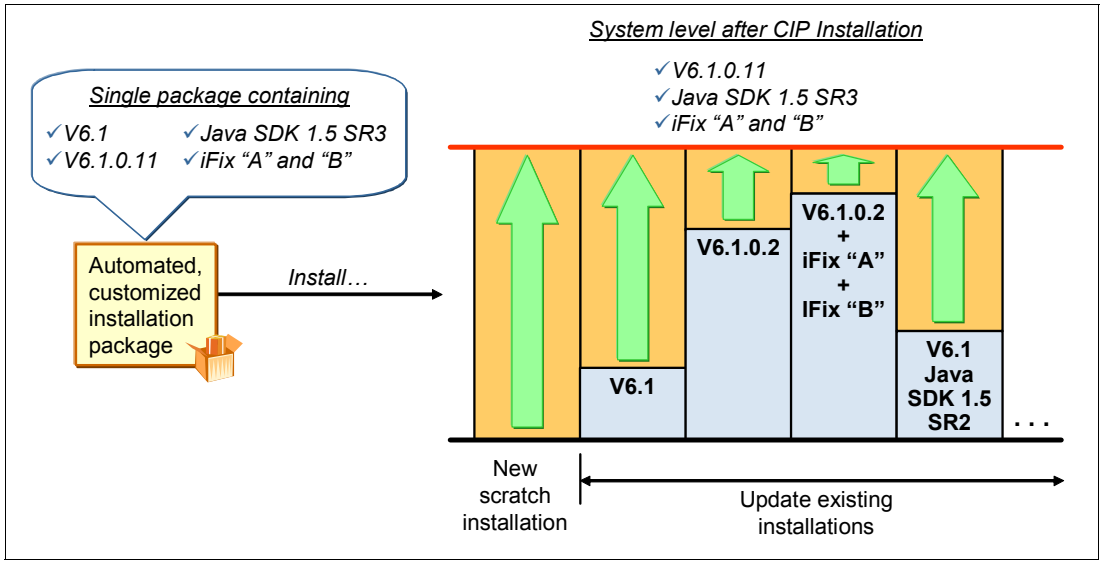


Figure 4-27 Example of a CIP build

Scripts

The capability to include scripts with a Custom Installation Package can provide a very flexible mechanism for configuring an installation. The Installation Factory supports a number of different script formats, including Java class files, JACL, Jython, Ant, and batch and shell scripts. Scripts of different formats can be included in the same Custom Installation Package, and associated with either the installation or uninstallation option.

Installation scripts are run after the installation is complete, while uninstallation scripts are run before the uninstallation takes place.

The scripts included can also reference other data stored within the Custom Installation Package, such as additional files or EARs.

Note: The scripts and additional files are located at:

Scripts: <WAS_HOME>/cip/cipID/config

Additional files: <WAS_HOME>/cip/cipID/userFiles

Deploy EARs: <WAS_HOME>/cip/cipID/installableApps

Creating a WebSphere Application Server CIP

Do the following steps:

1. Sign on as a non-root user, and download the Installation Factory tool to the /WAS/installation factory directory. The Installation Factory for WebSphere Application Server V6.1 on Solaris can be downloaded from:

   http://www.ibm.com/support/docview.wss?rs=180&uid=swg24016062

2. Change the current directory to the /WAS/installation factory directory. Unzip the Installation Factory tool using the following commands:

   ```
   nonroot@bash$ gunzip installfactory.6100.solaris.sparc.tar.gz
   nonroot@bash$ tar xvf installfactory.6100.solaris.sparc.tar
   ```

3. Launch Installation Factory in GUI mode by running **nonroot@bash$./ifgui.sh**.

4. Click **Create New Customized Installation Package**, as shown in Figure 4-28.

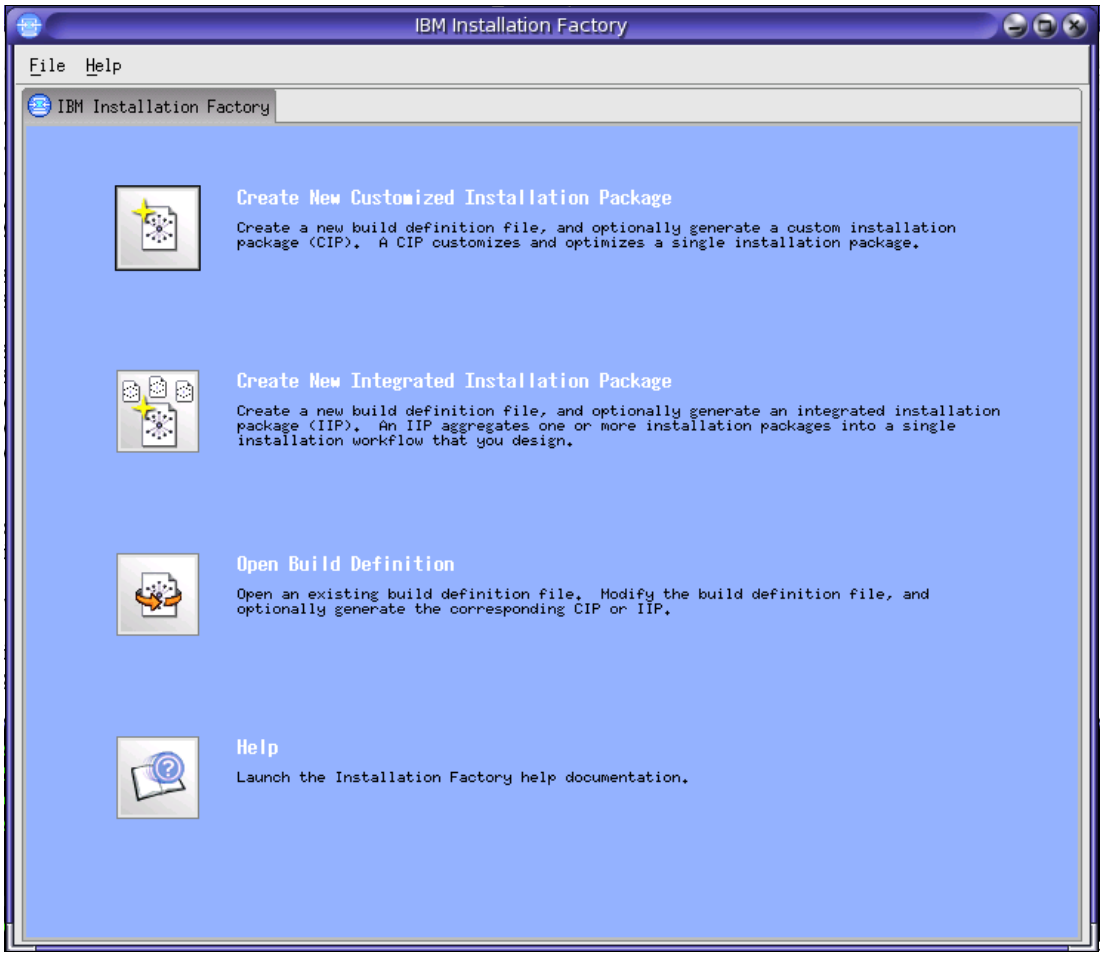


Figure 4-28 Customized Installation Factory: Creating a new CIP

5. Choose **WebSphere Application Server** and **Network Deployment**, as shown in Figure 4-29. Click **Finish**.

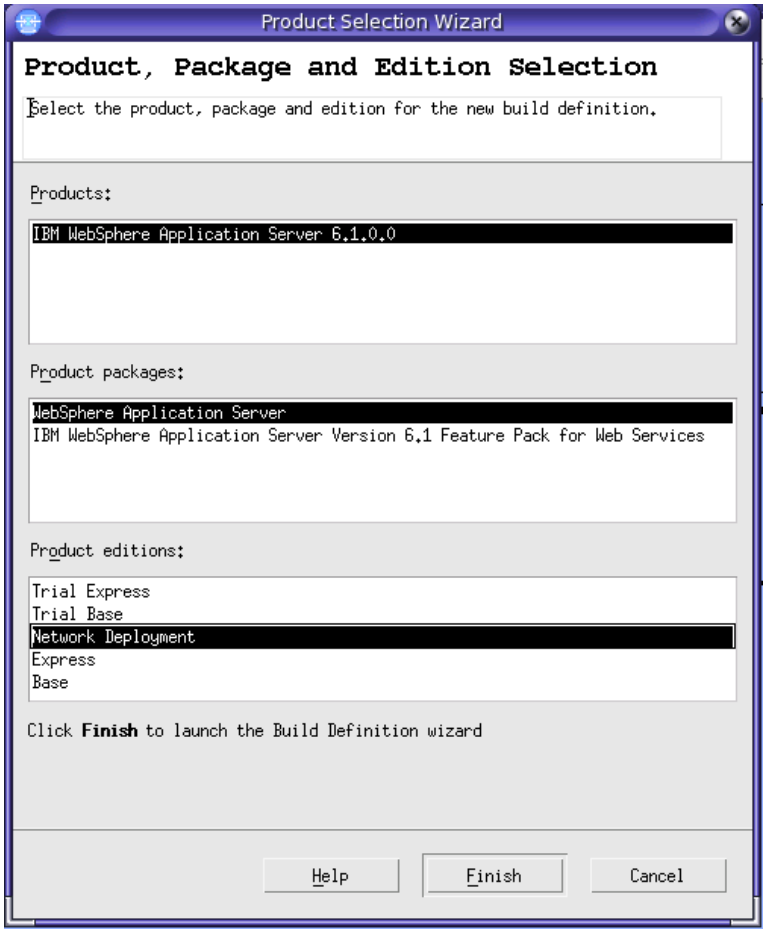


Figure 4-29 Customized Installation Factory: Select the product

6. Select the default **Connected mode** and **Sun Solaris** platform on the Mode selection window, as shown in Figure 4-30. Click **Next**.

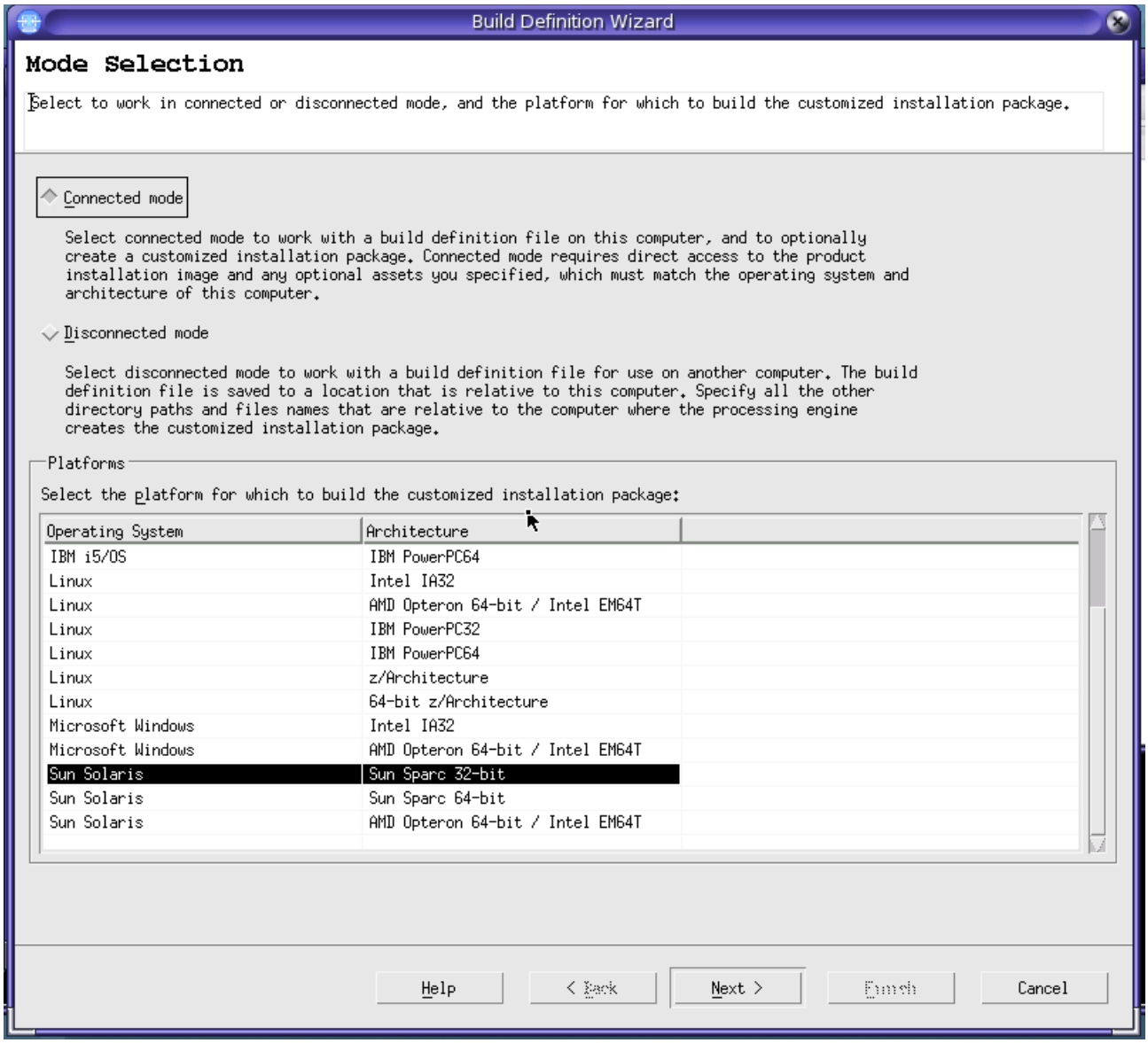


Figure 4-30 Customized Installation Factory: Mode Selection

7. Change the Identifier to com.ibm.samplewascip and use the default value for Version (which is 1.0.0.0), as shown in Figure 4-31. Click **Next**.

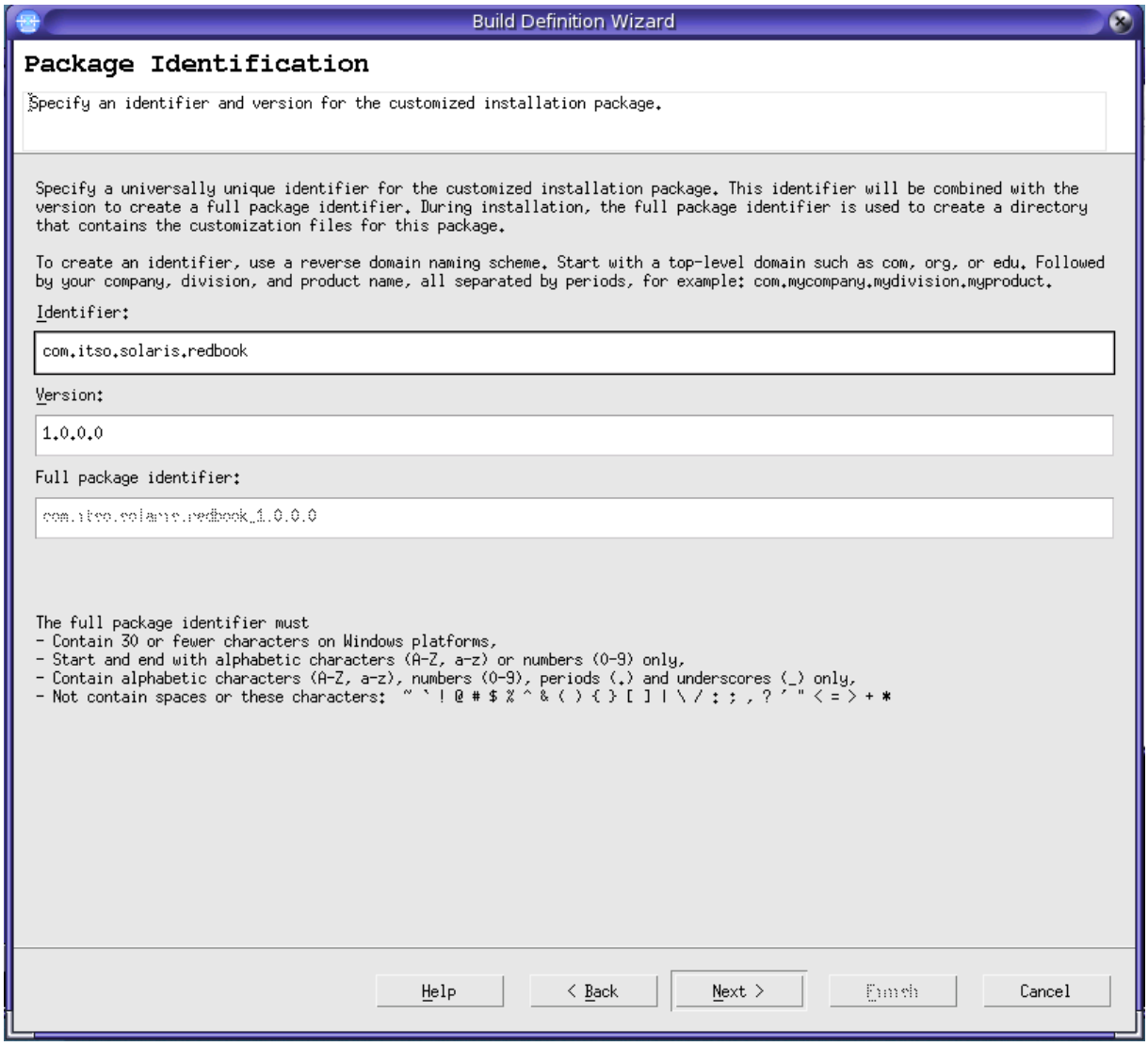


Figure 4-31 Customized Installation Factory: Package Identification

8. Leave the default for the Build definition file name and CIP build directory path, as shown in Figure 4-32. Click **Next**.

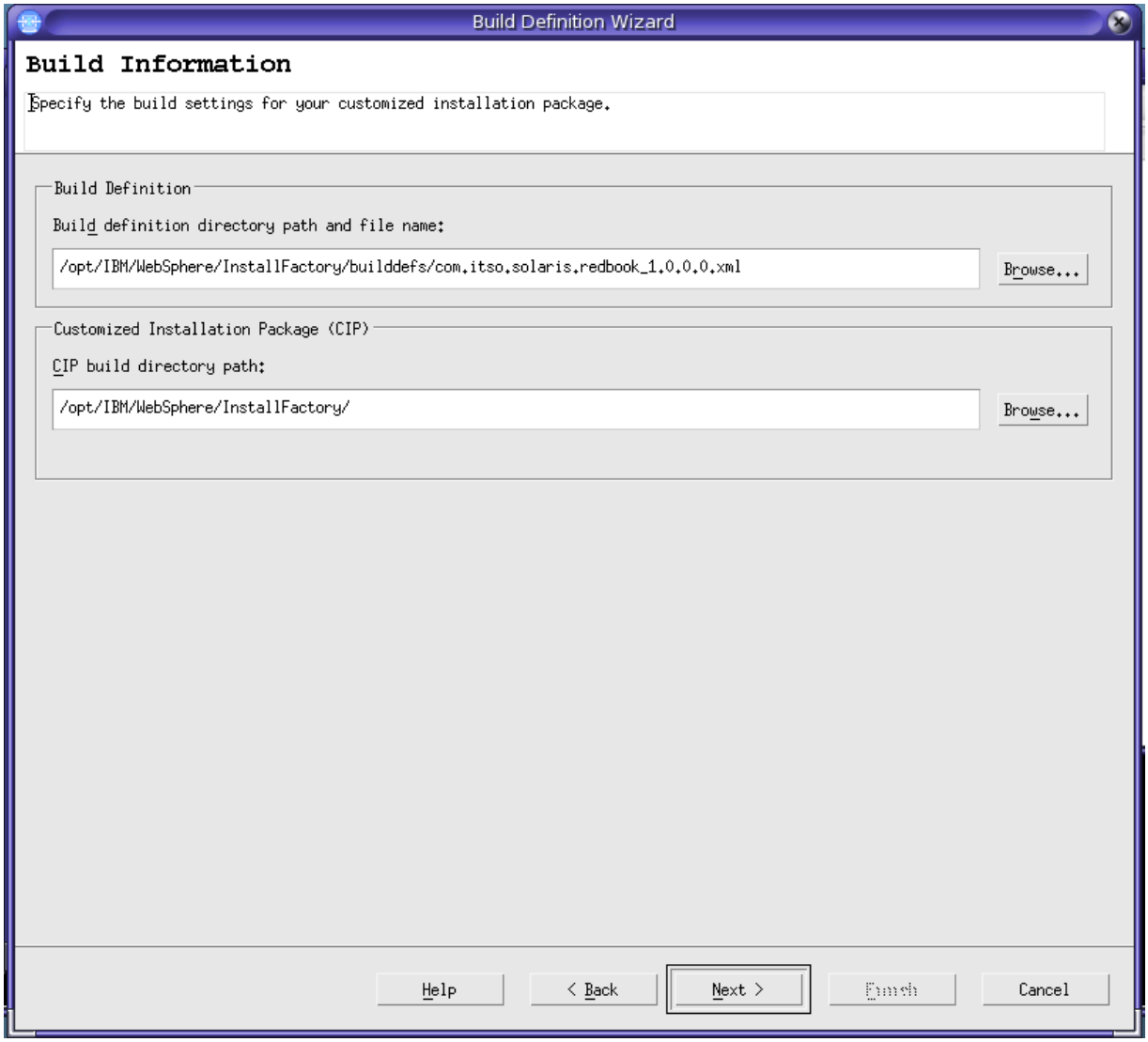


Figure 4-32 Customized Installation Factory: Build Information

9. Browse to the WebSphere Application Server Version 6.1 installation image location, as shown in Figure 4-33. Click **Next**.

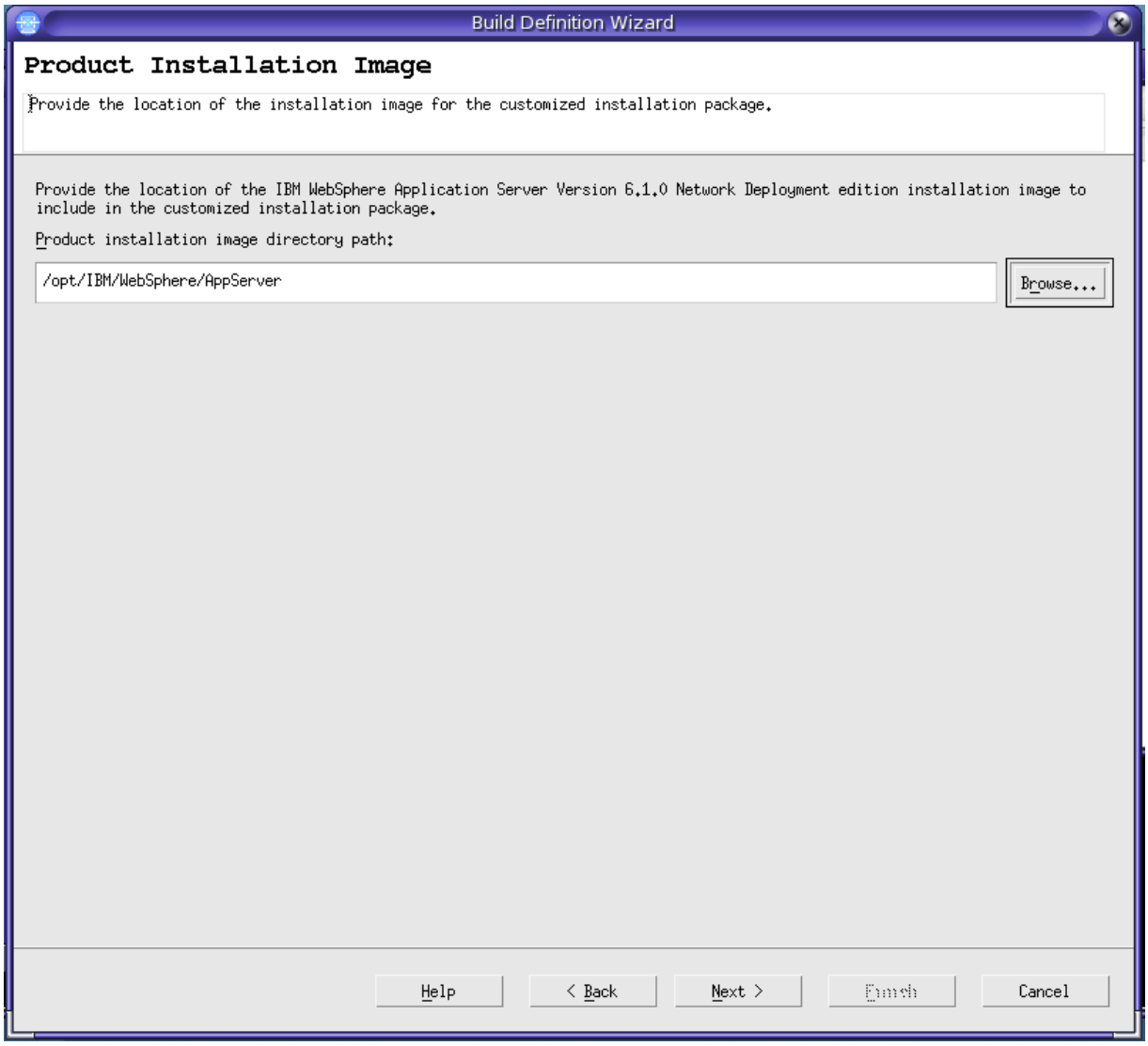


Figure 4-33 Customized Installation Factory: Product Installation Image

Note: Note that this image is the installable image from a CD or download location that can be used to install WebSphere Application Server. It is not the image that has already been installed on a system.

10. Use the defaults in the Feature Selection window, as shown in Figure 4-34. Click **Next**.

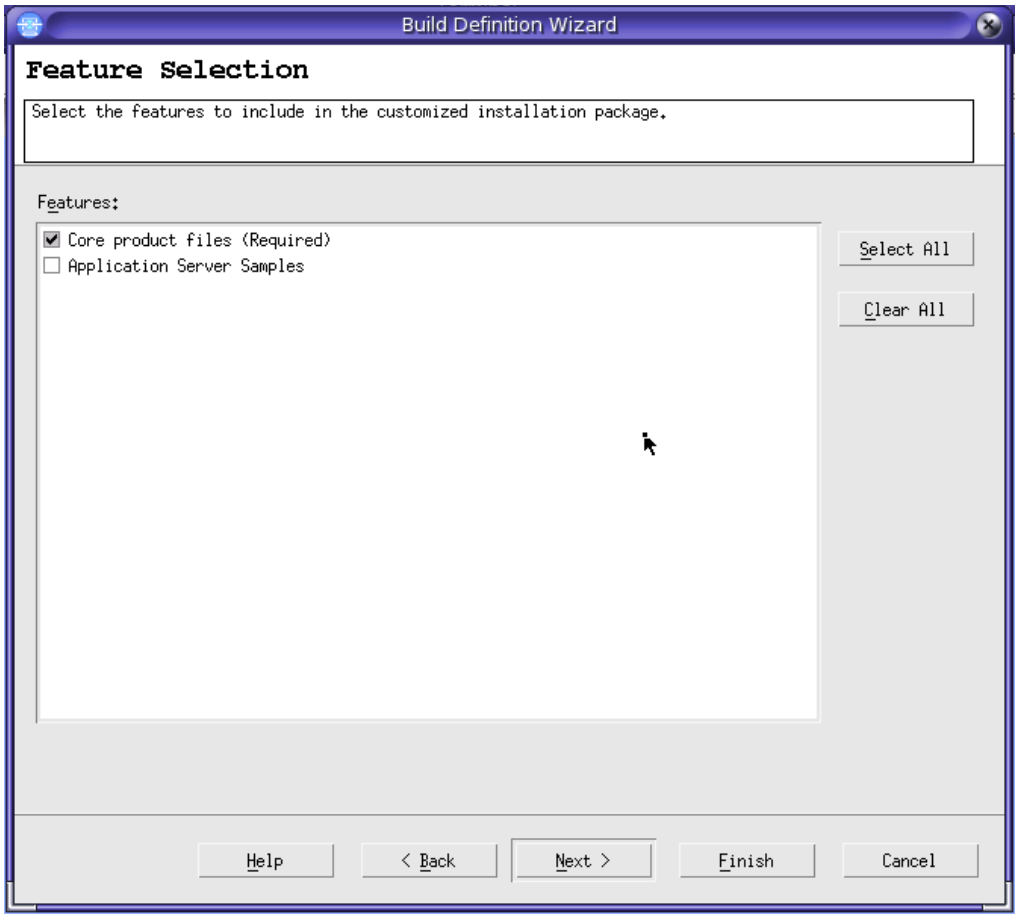


Figure 4-34 Customized Installation Factory: Feature Selection

11. Enter in the paths to WebSphere Application Server Fix Pack 6.1.0.11 and SDK fix pack 6.1.0.11, as shown in Figure 4-35. Click **Next**.

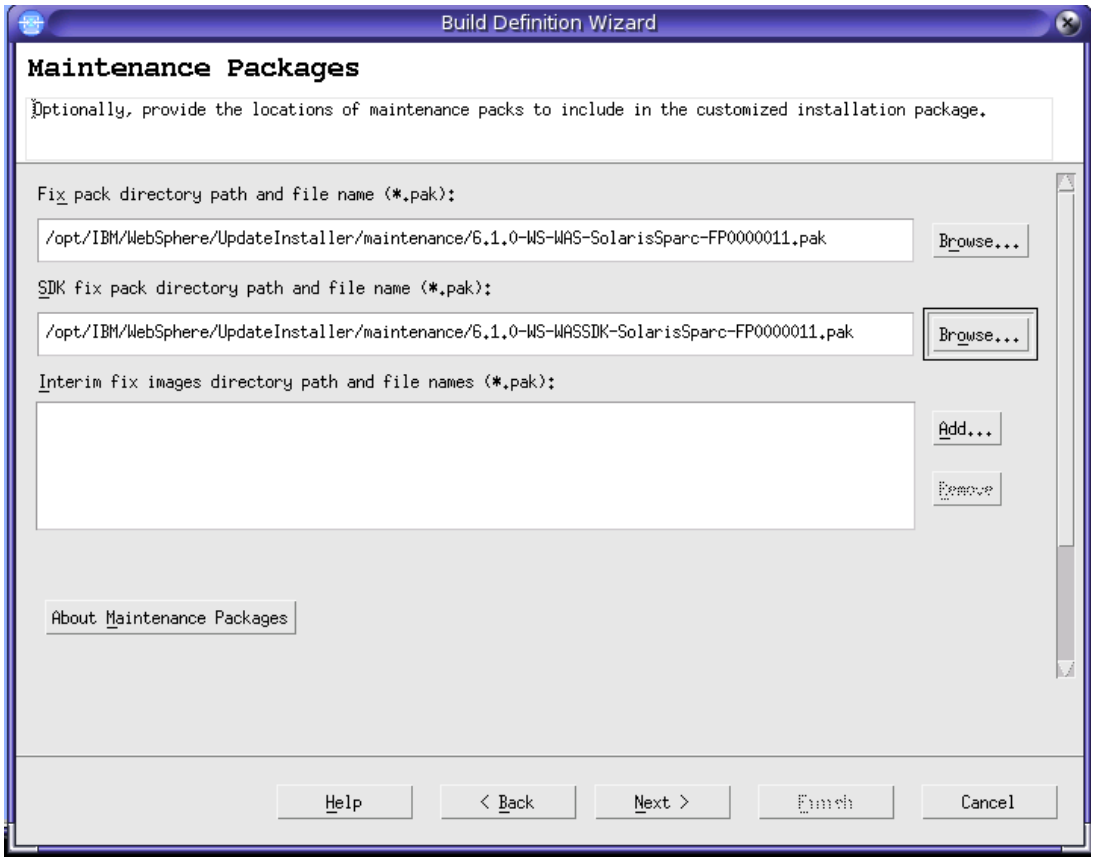


Figure 4-35 Customized Installation Factory: Main ten ace Package

12. Use the defaults in the Installation and Uninstallation Scripts window. Click **Next**.
13. Use the defaults in the Profile Customization window. Click **Next**.
14. Use the defaults in the Additional Files window. Click **Next**.
15. In the Authorship window, input the Organization and Description of the CIP. Click **Next**.
16. Click the **Save build definition file and generate customized installation package** radio button. Click the **Estimated Size and Available Space** button to check the available space, as shown in Figure 4-36 on page 105. Click **Finish**.

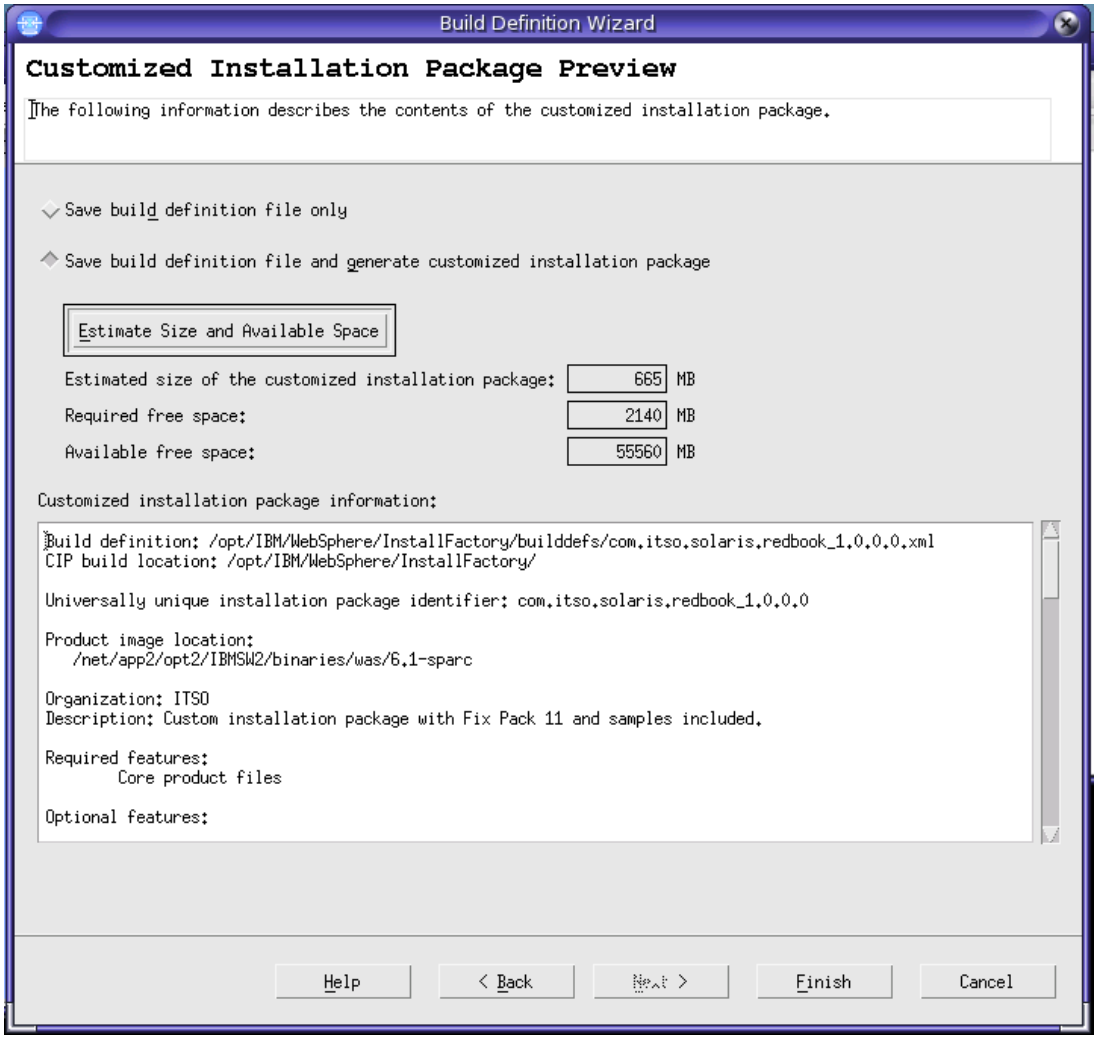


Figure 4-36 Customized Installation Factory: Preview

17. Watch the progress of the CIP generation.
18. After the CIP is created successfully, click **OK** in the confirmation window.

4.4.2 Problem determination

The Installation Factory uses the same logging strategy as the other installation packages in WebSphere Application Server V6.1, meaning they use a consistent location for log and trace files, and a consistent format for those files.

When running the command-line option to generate a custom installation package, logging can be enabled using the -logFile and -logLevel options. By default, the Installation Factory will log information to the logs directory in the Installation Factory home directory. By increasing the level, additional information can be captured about a problem with the generation of a Customized Installation Package.

These two options are discussed in more detail here:

- -logFile log_file_path_name

  Identifies the log file. The default value is working_directory/logs/log.txt.

- -logLevel log_level

  Sets the level of logging. The default value is INFO and the valid values are:

  - ALL
  - CONFIG
  - INFO
  - WARNING
  - SEVERE
  - OFF (Turns off logging)

Installation logs

When using a custom install package to install WebSphere Application Server, errors will be logged to the standard logs/install directory. It is the same as a standard WebSphere Application Server installation. The easiest method to determine if an installation was created from a custom installation package is to look in the <WAS_HOME>/CIP directory.

Custom errors and resolution

- Could not install contributions individually in silent mode.

  The response file for individual invocation has errors, for example, the installation location is not correct, or you did not mark `-OPT silentInstallLicenseAcceptance="true"`.

  Therefore, there are three copies of the same response file under the following location:

  <CIP_build_location>/ifpackages/WAS/responsefile.nd.txt

- Bundling WebSphere Application Server fix packs in Feature Pack for Web Services CIP fails validation.

  This is by design. Users are not supposed to bundle WebSphere Application Server fix packs into a Feature Pack for Web Services CIP. Only Feature Pack for Web Services fix packs and interim fixes can be bundled into a Feature Pack for Web Services CIP. If the Feature Pack for Web Services CIP installation requires you to upgrade WebSphere Application Server fix pack levels, you can specify the parameter in the Feature Pack for Web Services silent installation response file:

  ```
  -OPT fixpackLocation="<fix_pack_location>"
  ```

- You need to include installation and uninstallation scripts, or install profile customizations.

  Use the Installation Factory to create individual CIPs, then include the CIPs in an IIP.

- You need to push out periodic upgrades (WebSphere Application Server Fix Packs) to multiple sites.

- Use CIP and modify the silent response file with the new Fix Pack location.

4.5 Uninstalling

This section describes how to uninstall WebSphere Application Server Network Deployment.

The uninstaller program is created during installation, as described in 4.5.1, “Uninstall command” on page 108. The uninstaller program is customized for each product installation, with specific disk locations and routines for removing installed features.

The uninstaller program removes registry entries, uninstalls the product, and removes all related features. The uninstaller program does not remove log files in the installation root directory.

The uninstaller program removes all profiles, including all of the configuration data and applications in each profile. Before you start the uninstall procedure, back up the config folder, the installableApps folder, and the installedApps folder of each profile, if necessary. Back up all applications that are not stored in another location.

Estimating the time required to uninstall

The time required to uninstall is dependent on the processing speed of your machine. As a rough guideline, uninstalling the core product files and one application server profile takes approximately 10 minutes when using the **`uninstall`** command.

The uninstallation procedure

1. Log on as root or as a non-root user who belongs to the administrator group.
2. Run the uninstaller program for the Web server plug-ins for WebSphere Application Server. If a Web server is configured to run with the Application Server, uninstall the plug-ins to remove the configuration from the Web server.
3. Stop each running Application Server with the **`stopServer`** command.

   If security is disabled, the uninstaller program can stop all WebSphere Application Server processes automatically. If servers are running and security is enabled, the uninstaller program cannot shut down the servers and the uninstall procedure fails. Manually stop all servers before uninstalling.

   Stop all server processes in all profiles on the machine. For example, issue the following command from the /opt/IBM/WebSphere/AppServer/profiles/app_server_profile/bin directory to stop the server1 process in the profile:

   ```
   ./stopServer.sh server1
   ```

   If the servers are running and security is enabled, use the following command:

   ```
   ./stopServer.sh server1 -user user_ID -password password
   ```

4. Stop the nodeagent process with the **`stopNode`** command.

   Stop the nodeagent process that might be running on the machine. For example, issue the following command from the /opt/IBM/WebSphere/AppServer/profiles/app_server_profile/bin directory of a federated node to stop the nodeagent process:

   ```
   ./stopNode.sh
   ```

   If servers are running and security is enabled, use the following command:

   ```
   ./stopNode.sh -user user_ID -password password
   ```

5. Stop the deployment manager dmgr process with the **stopManager** command.

   Stop all dmgr processes that are running on the machine. For example, issue the following command from the /opt/IBM/WebSphere/AppServer/profiles/dmgr_profile/bin directory:

   ```
   ./stopManager.sh -user user_ID -password password
   ```

6. Optional: Back up configuration files and log files to refer to later, if necessary.

   The uninstaller program does not remove log files in the installation root directory. The uninstaller program removes all profiles and all of the data in all profiles.

   Back up the config folder and the logs folder of each profile to refer to it later, if necessary. You cannot reuse profiles so there is no need to back up an entire profile.

7. Uninstall the product.

   After running the **uninstall** command, the directory structure has only a few remaining directories. The logs directory is one of the few directories with files.

8. Review the *<was_home>/logs/*uninstlog.txt file.

   The *<was_home>/logs/*uninstlog.txt file records file system or other unusual errors. Look for the INSTCONFSUCCESS indicator of success in the log:

   ```
   Uninstall, com.ibm.ws.install.ni.ismp.actions.
   ISMPLogSuccessMessageAction, msg1,
   INSTCONFSUCCESS
   ```

9. Manually remove the log files and installation root directory before reinstalling.

   The uninstaller program leaves some log files, including the *<was_home>/logs/*uninstlog.txt file.

   Manually remove all artifacts of the product so that you can reinstall into the same installation root directory. If you do not plan to reinstall, you do not need to manually remove these artifacts.

4.5.1 Uninstall command

The **uninstall** command uninstalls the product. When you uninstall the product, the uninstaller program removes the core product files and all of the profiles.

Before you begin uninstalling the product

If the servers are running and security is enabled, the uninstaller program cannot shut down the servers and the uninstall procedure fails. Manually stop all servers before uninstalling the product.

You can disable security in the administrative console before uninstalling the product. Then the uninstaller program can stop all server processes. Select **Security → Global security** and clear the check box for enabling global security in the administrative console.

You can also modify the *<was_home>/profiles/profile_name/properties/*soap.client.props file in each profile to allow the uninstaller program to issue commands against the profile. After modifying the file, you can run the uninstaller program in a secure environment without having to manually stop each server process.

4.5.2 Uninstalling Network Deployment

This section describes uninstalling the WebSphere Application Server Network Deployment product.

The uninstaller program removes registry entries, uninstalls the product, and removes all related features. The uninstaller program does not remove log files in the installation root directory.

The uninstaller program removes all profiles, including all of the configuration data and applications in each profile. Before you start the uninstall procedure, back up the *config* folder, the *installableApps* folder, and the *installedApps* folder of each profile. Back up all applications that are not stored in another location.

This procedure uninstalls the WebSphere Application Server Network Deployment product.

1. Log on as root or as a non-root user who belongs to the administrator group.
2. Run the uninstaller program for the Web server plug-ins for WebSphere Application Server.

   If a Web server is configured to run with the Application Server, uninstall the plug-ins to remove the configuration from the Web server.
3. Stop the deployment manager dmgr process with the **stopManager** command. Stop all dmgr processes that are running on the machine. For example, issue this command from the /opt/IBM/WebSphere/AppServer/profiles/dmgr_profile/bin directory:

   ```
   ./stopManager.sh -user user_ID -password password
   ```
4. Stop the nodeagent process with the **stopNode** command.

   Stop the nodeagent process that might be running on the machine. For example, issue the following command from the /opt/IBM/WebSphere/AppServer/profiles/app_server_profile/bin directory of a federated node on a Linux machine to stop the nodeagent process:

   ```
   ./stopNode.sh
   ```

   If servers are running and security is enabled, use the following command:

   ```
   ./stopNode.sh -user user_ID -password password
   ```
5. Stop each running Application Server with the **stopServer** command.

   If security is disabled, the uninstaller program can stop all WebSphere Application Server processes automatically. If servers are running and security is enabled, the uninstaller program cannot shut down the servers and the uninstall procedure fails. Manually stop all servers before uninstalling.

   Stop all server processes in all profiles on the machine. For example, issue the following command from the /opt/IBM/WebSphere/AppServer/profiles/app_server_profile/bin directory to stop the server1 process in the profile:

   ```
   ./stopServer.sh server1
   ```

   If servers are running and security is enabled, use the following commands:

   ```
   ./stopServer.sh server1 -user user_ID -password password
   ```
6. Optional: Back up configuration files and log files to refer to them later, if necessary.

   The uninstaller program does not remove log files in the installation root directory. The uninstaller program removes all profiles and all of the data in all profiles.

   Back up the config folder and the logs folder of each profile to refer to later, if necessary. You cannot reuse profiles so there is no need to back up an entire profile.

7. Issue the uninstall command located at */<was_home>*/uninstall.

   The command file is named uninstall.

   The uninstaller wizard begins and displays the Welcome window displayed in Figure 4-37.

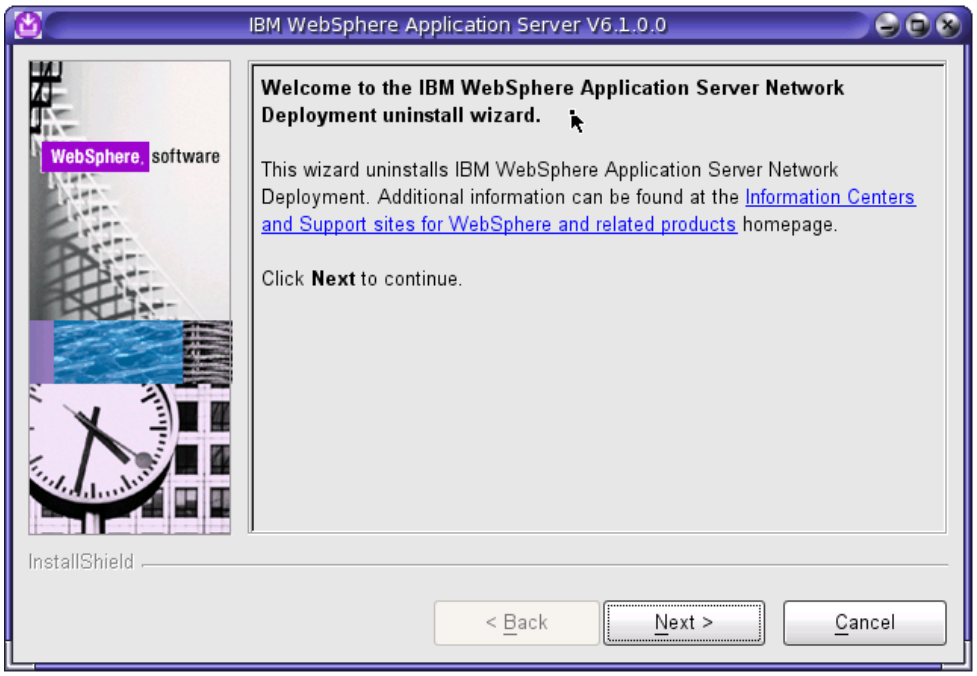


Figure 4-37 Uninstall Network Deployment: Welcome panel

8. Click **Next** to begin uninstalling the product. The uninstaller wizard displays a confirmation window that lists a summary of the product and features that you are uninstalling, a shown in Figure 4-38.

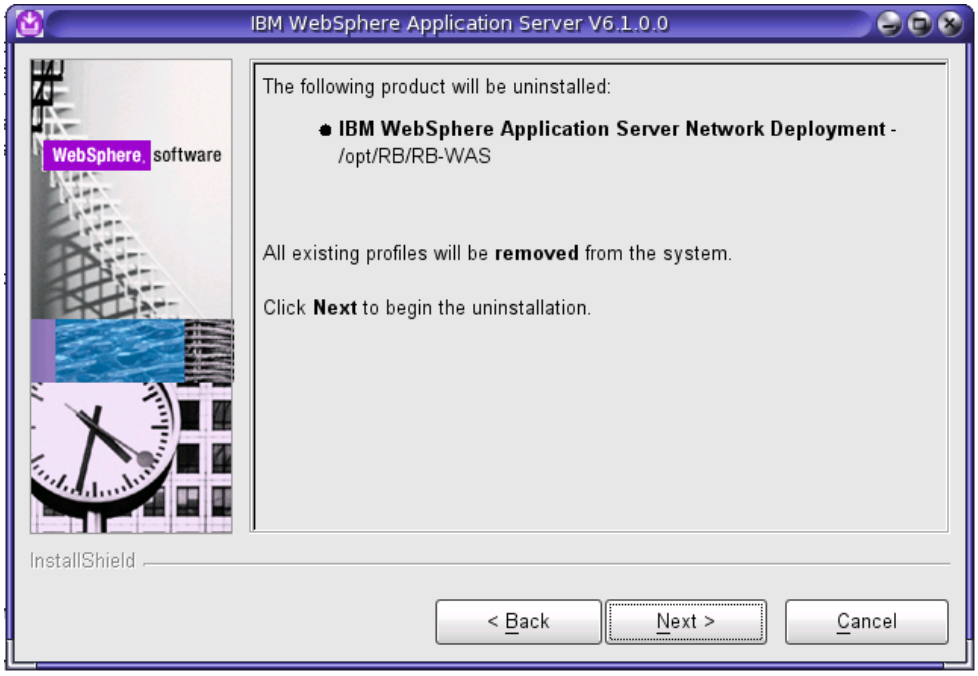


Figure 4-38 Uninstall Network Deployment: Summary of the product

9. Click **Next** to continue uninstalling the product.

   The uninstaller deletes all profiles before it deletes the core product files, as shown in Figure 4-39. Before using the uninstaller, back up any profile data that you intend to preserve. The uninstaller does not retain data files or configuration data that is in the profiles.

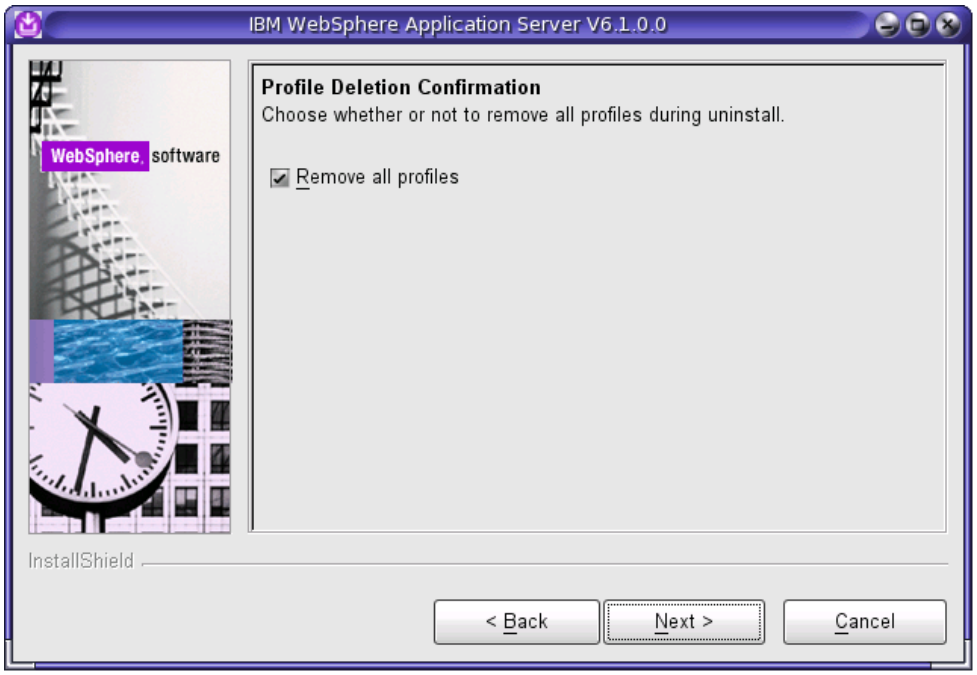


Figure 4-39 Uninstall Network Deployment: Profile deletion

After uninstalling profiles, the uninstaller program deletes the core product files in component order.

10. Click **Finish** to close the wizard after the wizard removes the product, as shown in Figure 4-40.

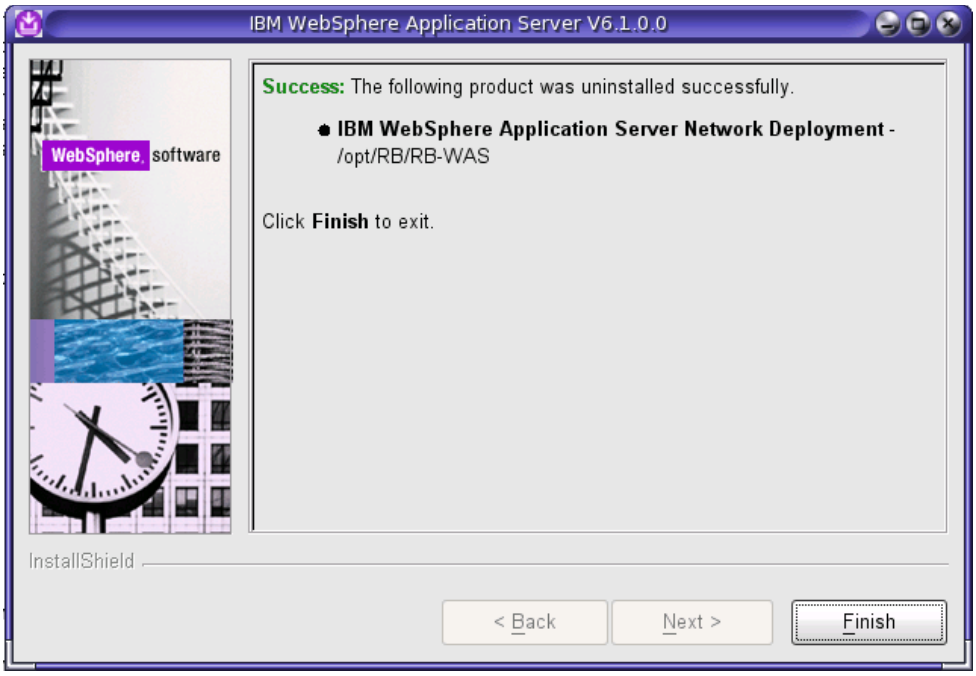


Figure 4-40 Uninstall Network Deployment: Finish installation

11. Remove any configuration entries in the managed node that describe a deleted deployment manager.

    A common topology is to install the core product files on multiple machines. One machine has the deployment manager and other machines have managed nodes created from custom profiles or federated application server profiles. If you delete a Network Deployment installation where you created an application server profile or a custom profile and federated the node into a deployment manager cell in another installation, you must remove the configuration from the deployment manager.

    The official statement of support for a node configuration problem in the managed node is that you use the **`backupConfig`** command after the initial installation. Use the command again whenever you make significant changes to the configuration that you must save. With a valid backup of the configuration, you can always use the **`restoreConfig`** command to get back to a previously existing state in the configuration.

    You can also use the following command to remove the node when the deployment manager is not running. Issue the command from the <was_home>/profiles/managed_node_profile/bin directory on the machine with the managed node:

    ```
    ./removeNode.sh -force
    ```

    If you must manually clean up the configuration on the managed node, you can attempt the following *unsupported* procedure:

    a. Rename the cell_name directory for the node to the original name if the current name is not the original name.

       Go to the <was_home>/profiles/node_profile_name/config/cells/ directory. Rename the *cell_name* directory to the original name.

    b. Delete the dmgr_node_name directory if it exists.

Go to the <was_home>/profiles/node_profile_name/config/cells/original_cell_name/nodes directory to look for the dmgr_node_name directory that you must delete.

c. Edit the **setupCmdLine.sh** file and change the cell name to the original cell name.

The file is in the <was_home>/profiles/node_profile_name/bin directory. Change the value of the WAS_CELL variable to the original cell name.

12. Remove any configuration entries in the deployment manager that describe a deleted managed node.

Open the administrative console of the deployment manager and select **System administration** → **Nodes** → ***node_name*** → **Remove node**.

If the administrative console cannot successfully remove the node, run the following command with the deployment manager running:

```
<was_home>/bin/cleanupNode.sh node_name
```

The official statement of support for a node configuration problem in the deployment manager is that you use the **backupConfig** command after the initial installation. Use the command again whenever you make significant changes to the configuration that you must save. With a valid backup of the configuration, you can always use the **restoreConfig** command to get back to a previously existing state in the configuration.

If you must manually clean up the configuration, you can attempt the following unsupported procedure:

a. Within the nodes directory of the deployment manager, remove the configuration directory for the node that you deleted.

Go to the <was_home>/profiles/dmgr_profile_name/config/cells/cell_name/nodes directory to find the deleted_node_name file.

b. Within the buses directory of the deployment manager, remove the configuration directory for the node that you deleted.

Go to the <was_home>/profiles/dmgr_profile_name/config/cells/cell_name/buses directory to find the deleted_node_name file.

c. Edit the coregroup.xml file in each subdirectory of the coregroups directory of the deployment manager. Look for elements of type coreGroupServers. Remove any coreGroupServers elements that have a reference to the node that you deleted.

Go to the <was_home>/profiles/dmgr_profile_name/config/cells/cell_name/coregroups/deleted_node_name directory to find the file.

d. Edit the nodegroup.xml file in each subdirectory of the nodegroups directory of the deployment manager. Look for elements of type members. Remove any members elements that have a reference to the node that you deleted.

Go to the <was_home>/profiles/dmgr_profile_name/config/cells/cell_name/coregroups/deleted_node_name directory to find the file.

This procedure uninstalls the Network Deployment product.

After running the **uninstall** command, the directory structure has only a few remaining directories. The logs directory is one of the few directories with files.

The uninstaller program leaves some log files, including the <was_home>/logs/uninstlog.txt file.

Manually remove the log files and installation root directory so that you can reinstall into the same installation root directory. If you do not plan to reinstall, you do not need to manually remove these artifacts.

5

Configuration of WebSphere Application Server in an advanced Solaris 10 Environment

In this chapter, we describe how to configure WebSphere Application Server in the Solaris 10 environment using its advanced features, such as Solaris Containers, Service Management Facility (SMF), Process Rights Management (Least Privilege model), and ZFS. We also discuss deploying WebSphere Application Server with Sun virtualization technologies along with deployment strategies and configuration scenarios.

5.1 Deploying WebSphere Application Server with Sun virtualization technologies

Sun offers various virtualization capabilities ranging from partitioning at the hardware level to isolating at the OS virtualization level or with the Solaris Resource Manager, as shown in Figure 5-1. You can leverage these capabilities to maximize your resource utilization, consolidate application hosted environments, and minimize the server sprawl in your data center, improving overall efficiency and reducing the total cost of ownership. We explain briefly how each virtualization capability is designed and how you can leverage it for deploying your applications in the IBM WebSphere Application Server environment.

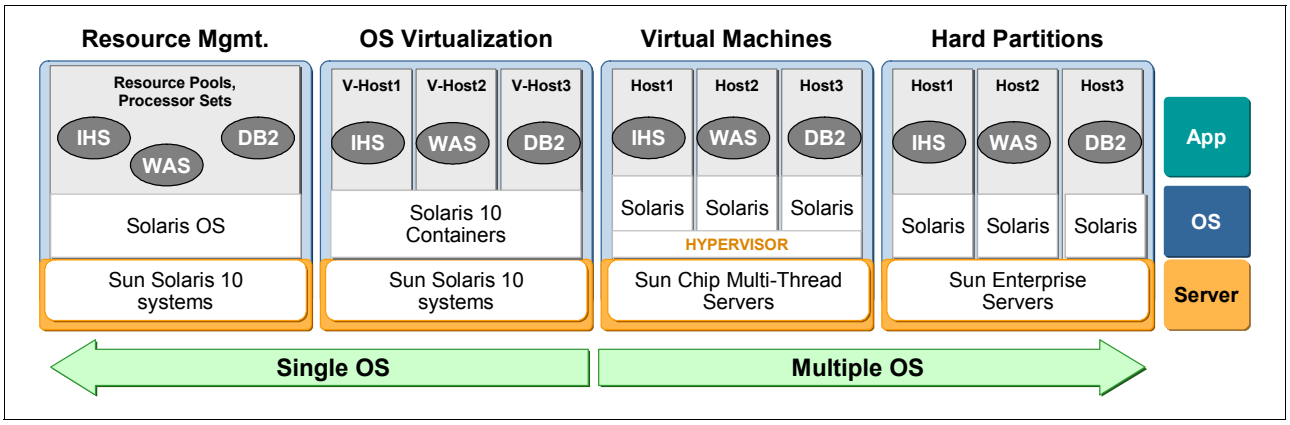


Figure 5-1 Sun virtualization technologies

For more information about Sun virtualization technologies, refer to *Sun Server Virtualization Technology*, found at:

http://sun.com/blueprints/0807/820-3023.html

5.1.1 Dynamic System Domains

Dynamic System Domains (DSD) enable mainframe like hardware partitioning, full fault isolation, and full dynamic reconfiguration (DR) capabilities. It is available on many of Sun's mid-range and high-end SPARC servers. Each domain on a server represents a system with its own OS, network interfaces, disks, I/O, MAC addresses, IP Addresses, host name, name service, locale, time zone, and so on, as shown in Figure 5-2 on page 117. With DR, you can dynamically reconfigure system's hardware resources even while the domains are active. For example, you can move a system board with CPUs and memory from one domain to another, add a new system board to increase CPU and memory capacity in an existing domain, or hot-swap an I/O PCI board. No system reboot is required for the system to recognize and utilize those resources. You can grow and shrink the domains as needed. For more information, see http://sun.com/servers/highend/dr_sunfire.

With the Sun Fire Capacity on Demand (COD) program, you can purchase these systems by paying only for the system with CPU and memory resources that you need. As your business demand grows, your resource requirement will grow. COD enables you to instantly activate those spare resources and utilize them only for the duration that you need. For more information, see http://sun.com/datacenter/cod.

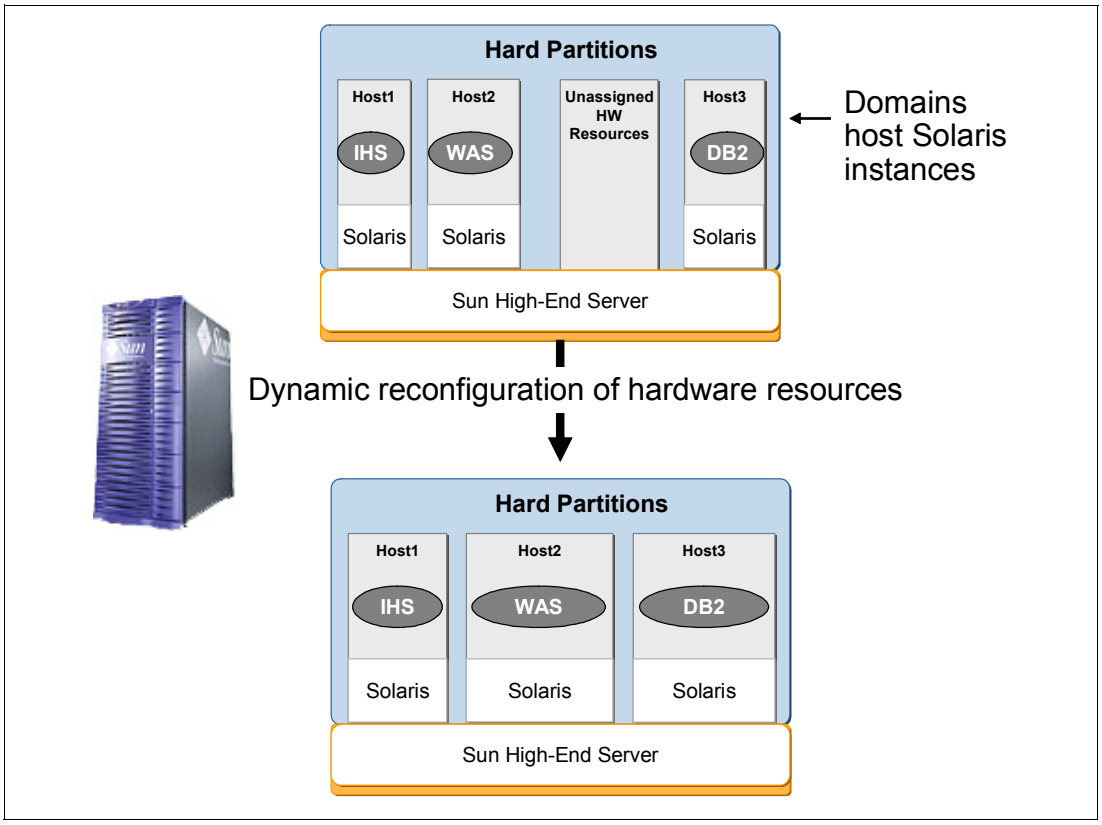


Figure 5-2 Dynamic System Domains and dynamic reconfiguration

5.1.2 Logical Domains

Logical Domains (LDoms) enable server virtualization capabilities based on a Hypervisor™. They are offered on Sun CoolThread servers (for example, T5220, T5120, T2000, T1000, and so on). Each LDom on a CoolThread server represents a virtual system with its own OS, network interface, disks, MAC address, IP Address, host name, name service, locale, time zone, and so on, just as with Dynamic System Domains, but with much finer resource granularity and available on smaller, low-cost servers. The LDoms can also be dynamically reconfigured with certain system resources such as adding or removing logical processors to or from LDoms without needing to reboot the system.

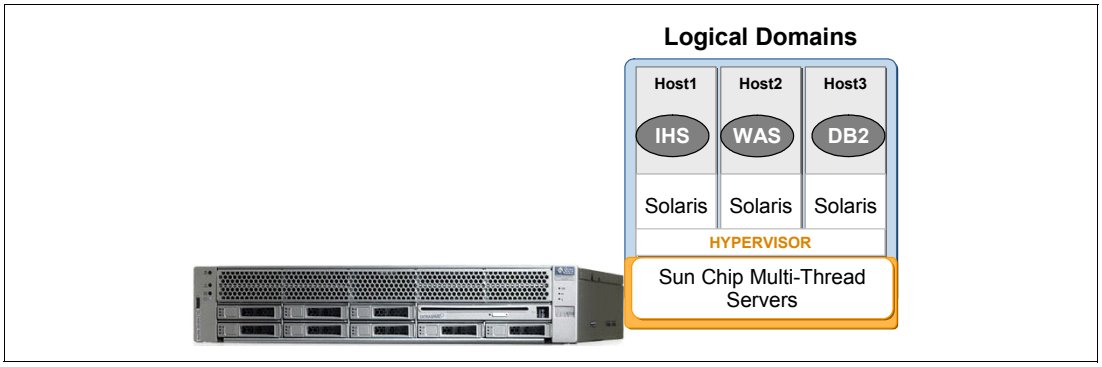


Figure 5-3 Sun T5220 with multiple domains

For more information about LDoms, refer to the *Beginners Guide to LDoms: Understanding and Deploying Logical Domains for Logical Domains 1.0 Release*, found at:

http://sun.com/blueprints/0207/820-0832.html

5.1.3 Solaris Containers

Solaris Containers is an OS-based virtualization feature built into Solaris, allowing large numbers of separate application environments to be hosted with extremely low overhead on any system (SPARC or x86) running Solaris 10 or later releases. It consists of Solaris Zones, Dynamic Resource Pools, and Solaris Resource Manager. Unlike LDoms and Dynamic System Domains, all Solaris Containers running in a given OS instance use the same Solaris kernel. Each zone represents a virtual system with its own node name, IP address, or addresses, name service, locale, time zone, and so on.

In addition, a Solaris container has the ability to provide support for alternate operating environments:

- Solaris 8 Migration Assistant, an optional product offering, allows an existing Solaris 8 system image to be rehosted within a Solaris Container running on Solaris 10; the environment within the container will present applications with the same execution environment as the original Solaris 8 system, and will be administered internally as a Solaris 8 OS image. This is discussed in greater detail in Appendix B, “Sun Solaris 8 Migration Assistant” on page 453.
- Solaris Containers for Linux Applications, included with Solaris 10 8/07 or later updates, makes it possible to host existing Linux applications on Solaris when running on an Intel or AMD server.

Figure 5-4 on page 119 shows an overview of Solaris Containers.

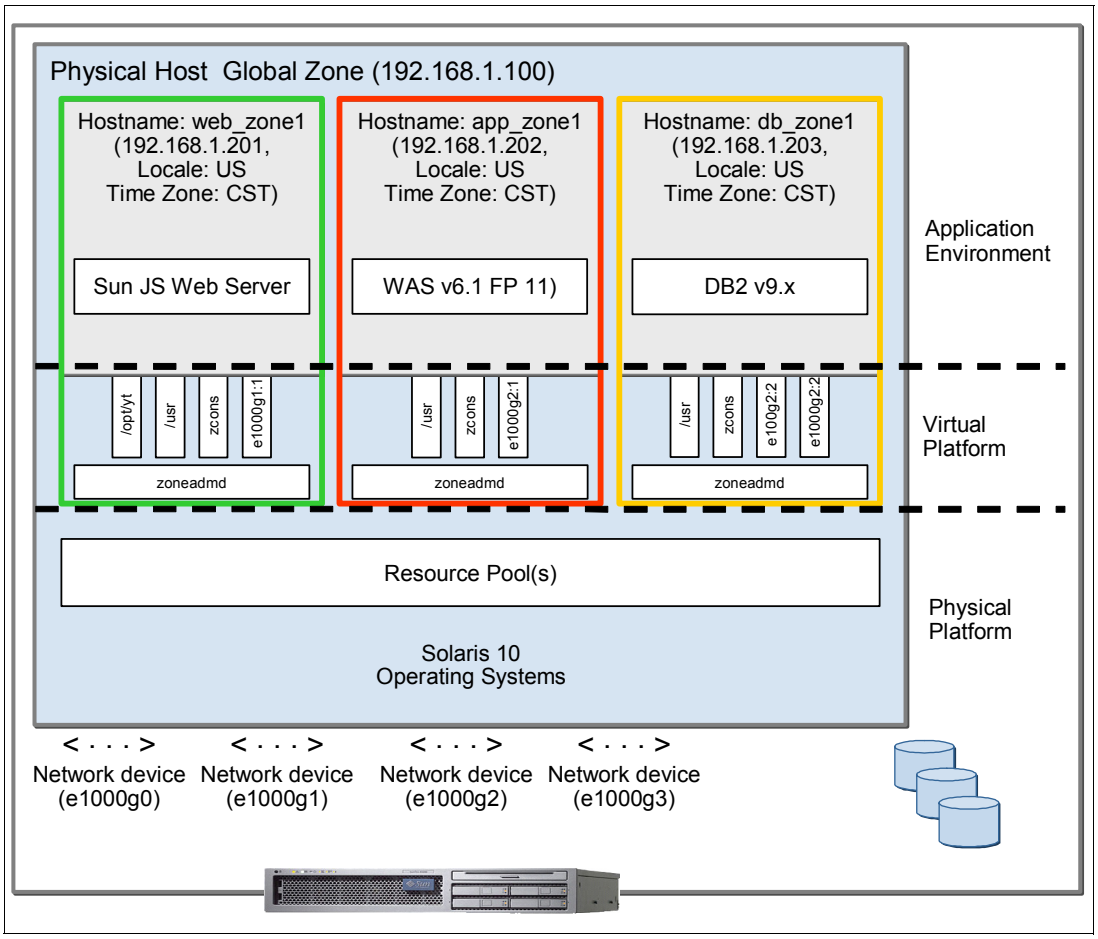


Figure 5-4 Solaris Containers

For more information, see *The Sun BluePrints Guide to Solaris Containers*, found at:

http://sun.com/blueprints/1006/820-0001.html

5.1.4 Sun xVM

Sun xVM is the umbrella name for Sun's next generation virtualization and management technologies, combining LDoms on SPARC systems with open source Xen-based virtualization capabilities on x64 systems. This will allow one virtualization management environment to provision, deploy, and manage virtualized Solaris and Linux OS instances on SPARC systems, and Solaris, Linux, and Windows OS instances on x64 systems.

Since Sun xVM is not generally available as of this writing, it will not covered here; see Sun's introductory article at http://sun.com/featured-articles/2007-1005/feature for more information.

Figure 5-5 gives an overview of Sun xVM virtual machines.

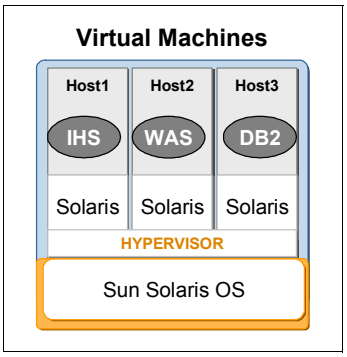


Figure 5-5 Sun xVM virtual machines

5.1.5 Choosing the right virtualization technology for your solution

Each virtualization technology is designed to meet different deployment requirements. We can characterize them in different levels of virtualization for application hosting as follows and provide the selection criteria in Table 5-1:

- Physical (Hardware-based)
- Virtual (Hypervisor)
- Virtual (OS-level software based)

Some of the requirements listed in this table, such as energy efficiency, may not necessarily be part of virtualization, but you gain those as benefits provided by the design of the underlying hardware.

Table 5-1 Selection criteria for Sun virtualization capabilities

Requirements	Technology	Platform
► Vertical scalability ► Large physical memory (greater than 64 GB of RAM) ► Complete fault isolation ► Dynamic reconfiguration ► Reliability, Availability, and Serviceablility (RAS) ► Multiple Solaris OS instances	Dynamic System Domains (Hardware Partitioning)	Sun Fire mid-range to high-end SPARC servers
► High throughput ► Best price per performance ► Energy efficiency ► Multiple Solaris OS ► Improved system utilization with consolidated workloads ► High-speed networking between virtual environments	Logical Domains (Virtualization with Hypervisor)	Sun CoolThread servers (T5220, T5120, Sun Blade T6320, T2000, T1000, and so on)

Requirements	Technology	Platform
► Vertical Scaling ► Virtually no impact to host in many application environments ► Secure and isolated process environment even while sharing a single Solaris OS instance ► Flexible resource management ► System observability ► Improved system utilization with consolidated workloads ► High-speed networking between virtual environments	Solaris Containers (Virtualization at the OS level)	Any SPARC or x86/64 (from a single CPU system to Sun Fire 25K) that runs Solaris 10 or later
► Solaris to host multiple guest operating systems, including Solaris, Linux, and Microsoft Windows	Sun xVM (Virtualization with a cross-platform, open source Hypervisor)	(See http://sun.com/xvm; not generally available at this writing)

WebSphere Application Server configuration and installation in DSD or LDoms are exactly the same as a stand-alone Solaris system. The configuration of DSD and LDoms must be done as part of the preparation steps to get the Solaris environment ready for WebSphere Application Server installation. These tasks are usually performed by Solaris system administrators or Sun service personnel. It is transparent to the WebSphere Application Server installation and configuration process.

Solaris Containers are also configured by the Solaris administrator. This task is performed from the default Solaris super-user environment on a physical server, referred to as the Global Zone. We discuss the details related to Containers in 5.2, “WebSphere in Solaris Containers” on page 121; it is critical for the Solaris system administrator to work together with the WebSphere Application Server administrator to create a configuration that best uses the full benefits of Solaris for your business requirements.

It is also possible to use Solaris Containers on systems that are also running DSDs or LDOMs; this combination can provide even greater flexibility and security.

IBM Support for Sun virtualization technologies

The statement for IBM support position for various virtualization technologies, including Sun Containers and Logical Domains, is available in IBM Technote (SWG21242532):

http://www.ibm.com/support/docview.wss?uid=swg21242532

5.2 WebSphere in Solaris Containers

Although the terms "Solaris Zones" and "Solaris Containers" are often used interchangeably, Solaris Containers are actually the combination of Solaris Zones and Solaris Resource Management utilities. Solaris Zones provides isolation capabilities to separate application hosting boundaries within the hosted system. Resource management provides partitioning capabilities to allocate and divide application resource utilization with more predictability.

Solaris Containers can be built with or without resource management (that is, just zones or zones with resource controls). In the former case, the zones on a system share all available resources based on the underlying Solaris scheduler's default policy. Solaris Zone provides virtualized environments where each zone behaves like a Solaris system. Solaris Zones offer the following features:

- Implemented through a light-weight layer in the Solaris OS.
- Underlying physical resources are hidden.
- Has its own host name, IP address(es), port space, super-user, time zone, locale, and name services.
- Isolated process space for security.

Figure 5-4 on page 119 shows three Solaris Containers, each with its own system identity and space, based on a single Solaris instance sharing underlying system resources.

There are many ways Solaris Resource Management controls can be imposed on Solaris:

- An individual zone with its own specified resources
- Multiple zones sharing specified resources
- An individual process
- A project for various processes (for example, Application Services processes that include WebSphere Application Server and DB2)

To get started with Solaris Containers, refer to the *Solaris Containers How To Guide*, found at:

`http://sun.com/software/solaris/howtoguides/containersLowRes.jsp`

Figure 5-6 on page 123 shows resource controls in effect for local zones for IHS, WebSphere Application Server, and DB2 deployments. The Web zone for IHS has a resource pool having 4 CPUs and 4 GB of RAM, while the zones for WebSphere Application Server and DB2 share another resource pool with 24 CPUs and 16 GB of RAM.

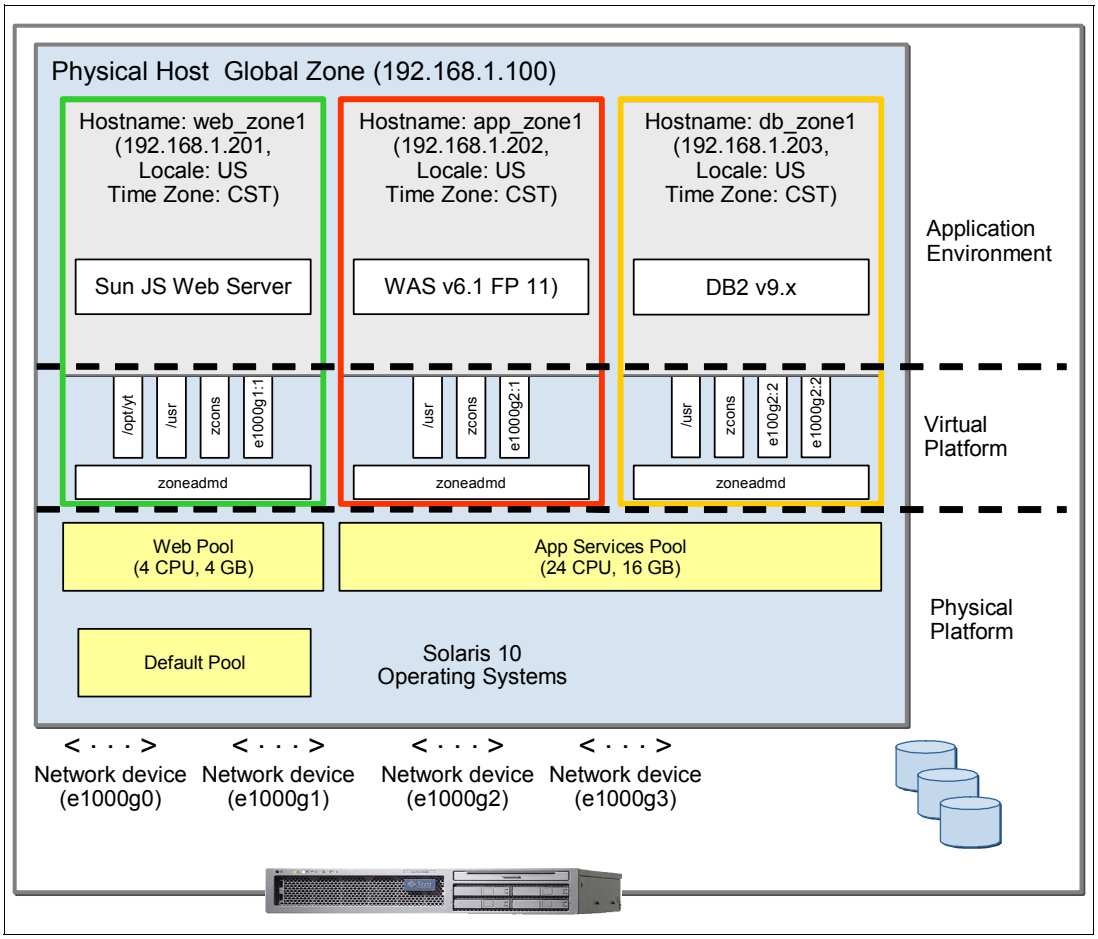


Figure 5-6 Multiple Solaris Zones with different resource pools

The advantage of using resource control is that no zone or business application can become a resource hog causing resource starvation of other services on the system. The resource control provides another dimension to contain your deployed services so that they can safely coexist with other services on the same physical system.

You can also build a Solaris Container as a plain zone and put resource controls at later time. We discuss further details about how you can accomplish this task in “Resource control for Solaris Zones” on page 148.

Tip: To collaborate with others on this topic, you can join the OpenSolaris community for zones at `http://www.opensolaris.org/os/community/zones`.

5.2.1 Deployment strategy

Solaris Zones are an inexpensive and flexible way to implement virtualization on any system that is running Solaris 10, from a single CPU AMD/Intel or SPARC system to a high-end SunFire 25K with 144 processor cores. Theoretically, you can create over 8000 zones per Solaris instance. From a practical standpoint, you should do proper capacity planning before you decide on the number of zones to host your applications.

There are many reasons why you should virtualize your applications with Solaris Zones. You may need a Non-Global Zone to consolidate servers in your data center because of a space shortage, or because you are overloaded with HVAC units, unmanageable growth, or simply underutilized systems, as shown in Figure 5-7, Since zones provide virtual system environments, you can re-host many of your applications from physical to virtual systems, isolate your co-located services safely and securely, and you can even relocate your virtualized application services easily from one hosted environment to another.

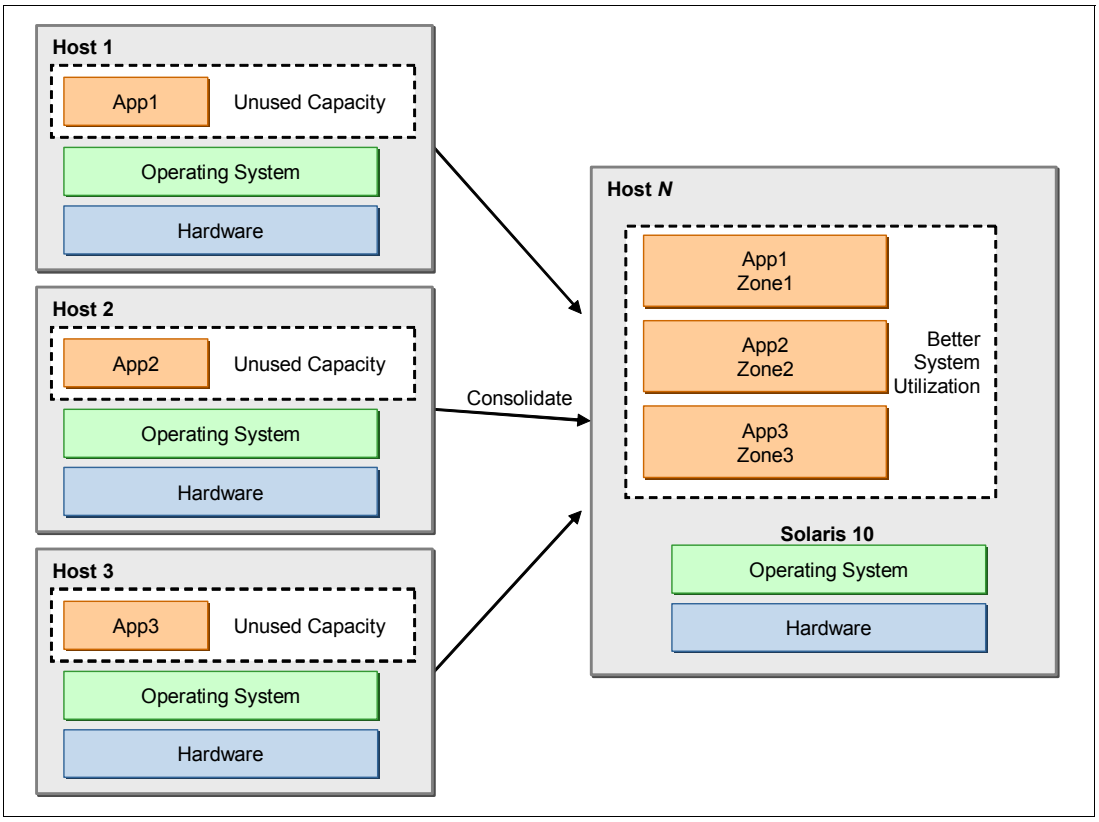


Figure 5-7 Consolidation through virtualization of many systems with unused capacity onto a Solaris 10 system with zones

Another reason, which really is a benefit you gain with Solaris zone, can be for redundancy. You can deploy WebSphere Application Server in one or more zones on a system. While this system is active for service, you can replicate these zones on another physical system. These replica zones are installed and configured, but not booted; thus, they are dormant, as shown in Figure 5-8 on page 125. If Host-A goes out of service for any reason, you can bring up your WebSphere Application Server zones on Host-B immediately. Since these redundant zones are replicas, they have identical host identities between the active and the dormant zones. Booting of zones take just several seconds. Once the zones boot up and your WebSphere Application Server services start up, the WebSphere Application Server servers are immediately back in service.

Tip: For full dynamic and automated deployment architecture, you can use this strategy along with WebSphere eXtended Deployment (XD), Tivoli Intelligent Orchestra, Tivoli Provisioning system, or Sun N1™ System Manager and N1 System Provisioning system.

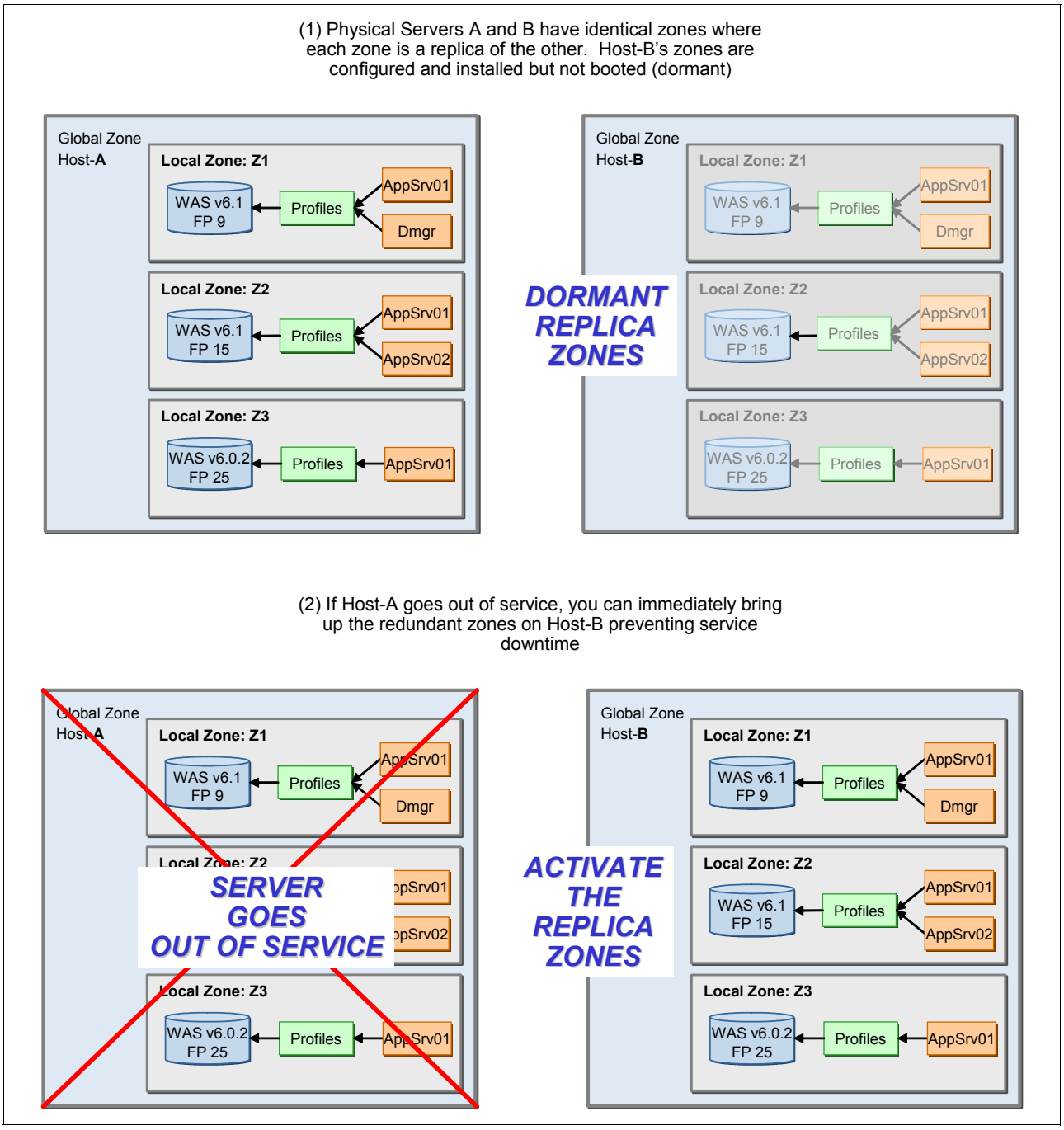


Figure 5-8 An example of virtual redundant systems using Solaris Zones

There are many other possibilities with Solaris Zones for your WebSphere Application Server deployment. You may currently have a number of physical systems that are needed for various application tiers or life cycle phases, as shown in Figure 5-9. For example, you need to have Development, Test, QA, Stage, and Production phases for your business applications. They usually span multiple tiers such as Edge, Web, Application, and Data tiers. You may be fully utilizing these systems in Production while you may be underutilizing some of these systems in Test, QA, and Stage. What if you wanted to re-purpose these underutilized systems to balance the system load in Production for a short term? Physical systems certainly require much work and effort or complex environments to support business agility. That is a challenge and a costly way to manage your data center today.

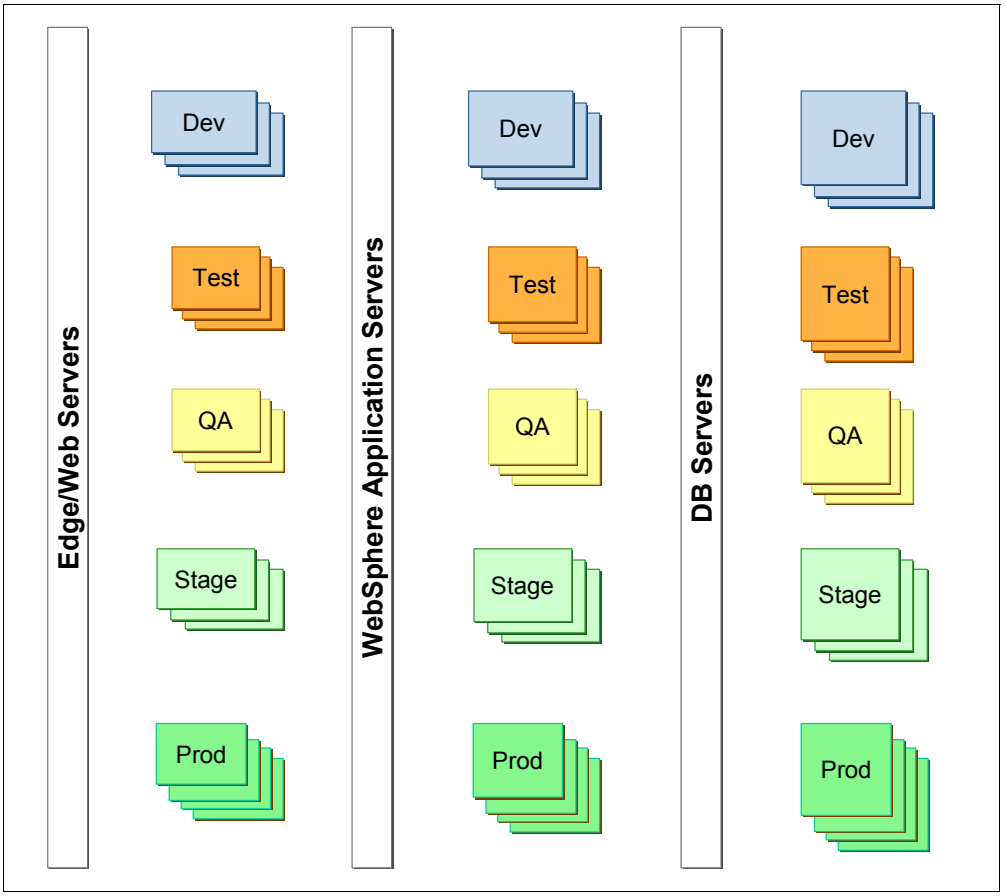


Figure 5-9 Physical servers needed for business applications

With virtualization capabilities, you may take an approach to consolidate the servers based on their life cycle phase, as shown in Figure 5-10. For example, development servers that span across its Web, App, and Data tiers. You may create these virtual systems and bring them online only as you need them.

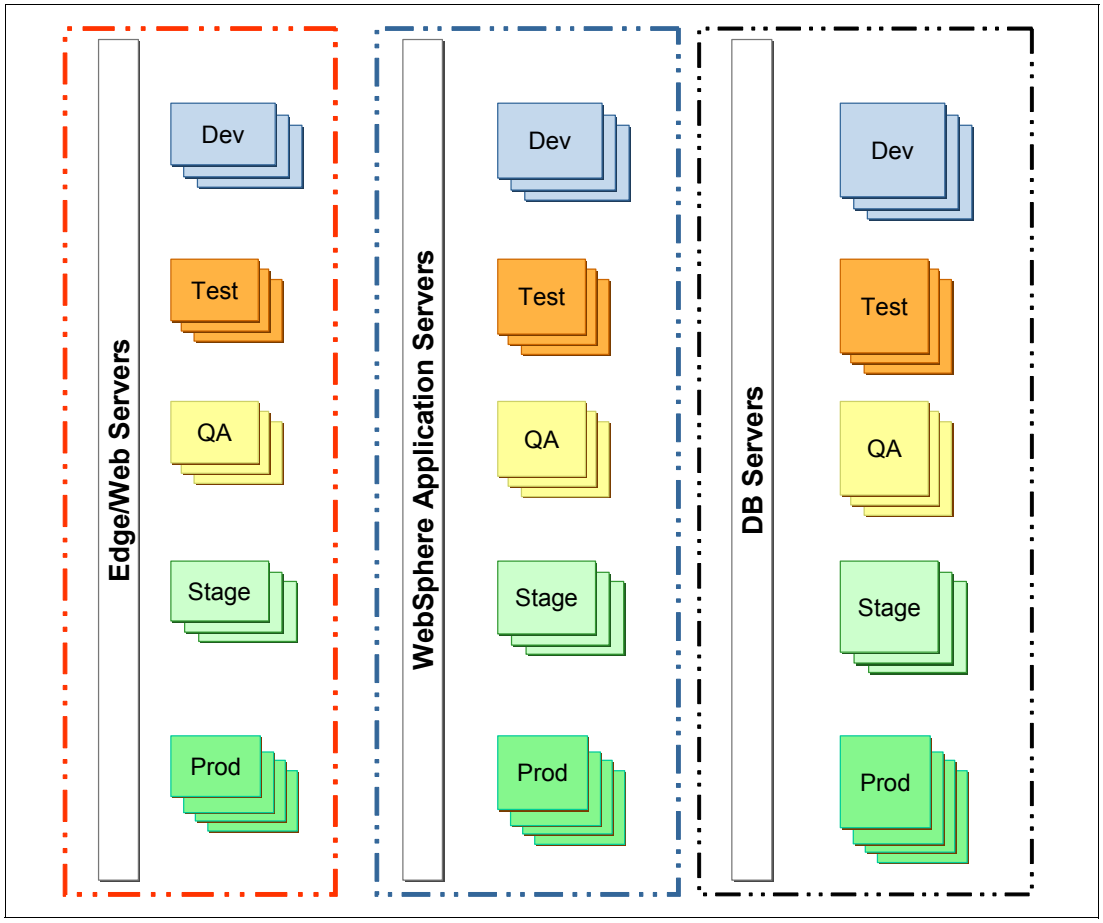


Figure 5-10 Virtualization/Consolidation by application tiers

Another approach that you may take is to virtualize and consolidate the servers based on their tier, as shown in Figure 5-11. For example, you may have all the Web tier servers virtualized on some servers while your app and data tier servers may be virtualized respectively.

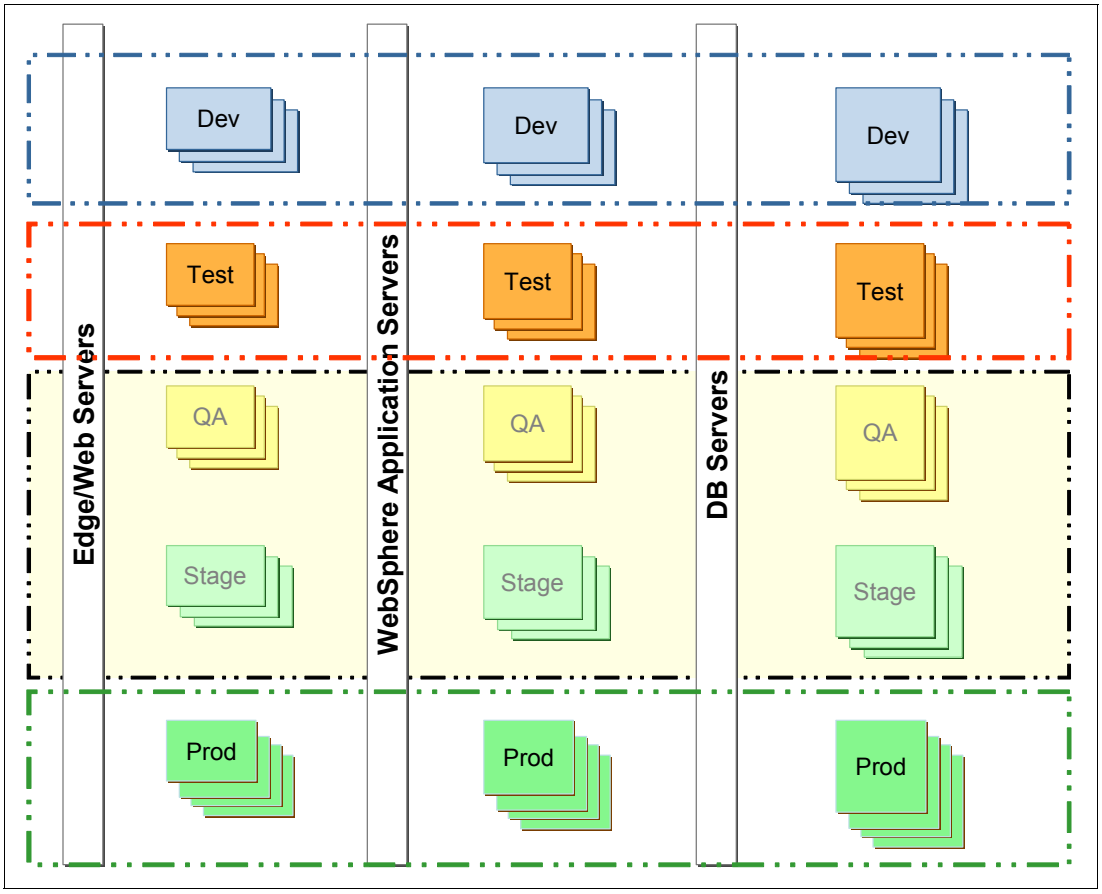


Figure 5-11 Virtualization/Consolidation by application life-cycle phases

Tip: By virtualizing your application environments, you gain the flexibility to bring them up when you need them and down when they are not in use. A great example would be Test/QA environments where you may not need to have dedicated physical systems at all times.

The key benefits of going with a virtualized system, especially Solaris Zones, are:

- Increased system utilization
- Virtually no impact using virtualization with zones
- Reduced the number of systems required (For example, by virtualizing Test systems in zones, they can be booted up only as needed.)
- Dynamically provision applications (For example, boot up the hosted services on a whim, which ties in nicely with WebSphere XD deployments.)
- Increased agility (For example, services can be redeployed and servers can be re-purposed easily and quickly.)
- Reduced the total operational costs (For example, reduction in need for Real Estate, Power, Cooling, and Sys Admin)

To realize these benefits, we now need to look into the considerations needed to configure WebSphere Application Server in Solaris Zones.

5.2.2 Configuration considerations

Before you start designing your WebSphere Application Server deployment architecture with Solaris Zones, it is imperative to understand the Solaris Container concept, that is, configuration of zones and resource management to be able to make decisions where and when to apply different types of zones, and WebSphere Application Server configuration for your business requirements and enterprise policies. First, starting from Solaris 10, every Solaris environment has a default operating environment called the Global Zone. Global Zone is the base Solaris operating environment that has direct access to all the physical layers through the standard device drivers and interfaces. This is also where you would create and manage virtualized environments called zones, officially known as Non-Global Zones. Often times, the Non-Global Zones are also known as local zones.

Different types of Solaris Zones

There are two different types of zones: Sparse Root or Whole Root Zones. The main difference is how their system directories are structured, as shown in Figure 5-12.

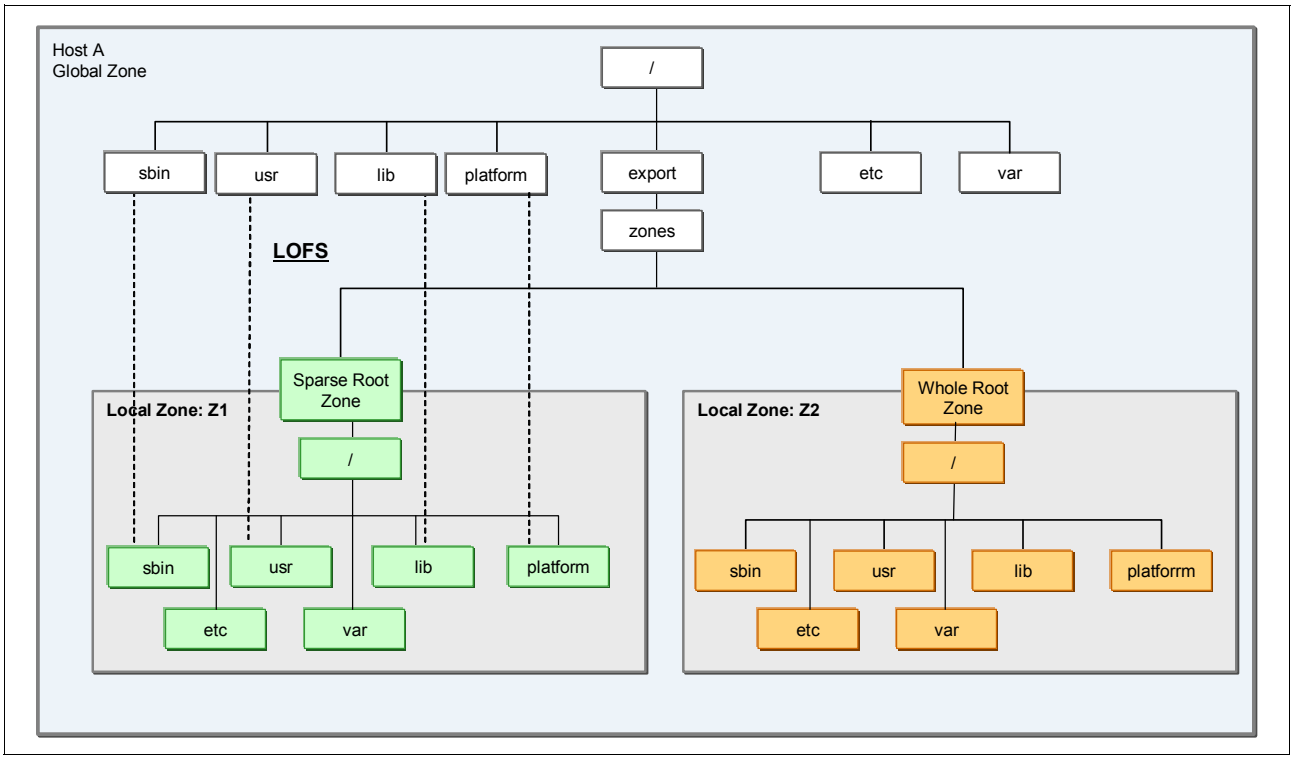


Figure 5-12 The key difference between Sparse Root and Whole Root Zone file system structures

Sparse Root Zone

Sparse Root Zone is a very light-weight zone that has references to system directories, such as /usr (also known as *loop-back-file system* (LOFS)), rather than the actual copies. It has the following advantages over the Whole Root Zone:

- Smaller storage requirements
- Faster creation and maintenance times
- Reduces VM footprint, as read-only library code pages are shared across zones

The left side of Figure 5-12 on page 129 shows that the Sparse Root Zone has LOFS, depicted with the dotted lines, of /usr and /sbin, which are read-only directories in the zone. All updates to these directories in the Global Zone will affect all Sparse Root local zones.

Whole Root Zone

Whole Root Zone has copies of the system directories, providing more flexibility for software installation. You can configure a Whole Root Zone to have copies of the /sbin and /usr directories, as shown in the right half of Figure 5-12 on page 129. It supports applications that need to write in the /usr directory during installation. Whole Root Zone also provides more flexible patching capability (that is, non-kernel patches) to the local zone.

Some enterprises set specific policies favoring one type of Solaris Zone over the other. In either case, Solaris Zones provide secure and separated operating environments for multiple WebSphere Application Server installations on a physical system. Users of each WebSphere Application Server node think that it is deployed in a dedicated system. If the WebSphere Application Server application environment in a zone ever gets compromised, only that zone gets affected. The remaining servers on the system can continue to operate safely.

In the next two sections, we discuss how WebSphere Application Server binaries can be structured as an independent installation or globally shared installation. You can apply either Sparse Root Zones or Whole Root Zones, depending on your needs.

Independent WebSphere Application Server binary installation in zones

The independent WebSphere Application Server installation in zones means each zone has its own copy of WebSphere Application Server, as shown in Figure 5-13. This is useful when you need to deploy different versions or Fix Pack levels of WebSphere Application Server in multiple local zones on the same physical system. When a Fix Pack needs to be applied on all WebSphere Application Server installation copies, you will need to apply it in all the existing zones.

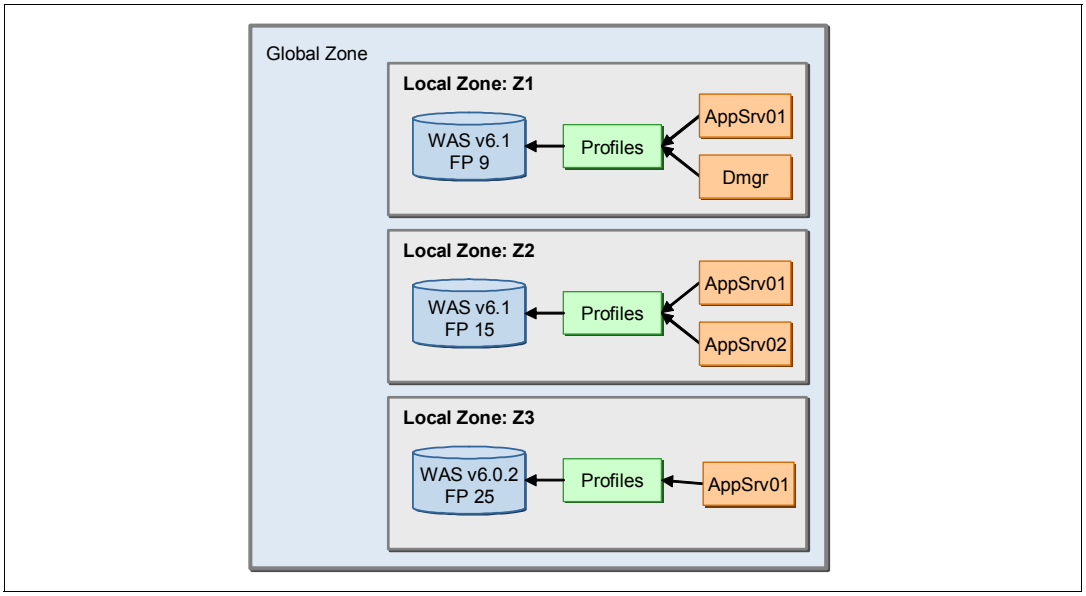


Figure 5-13 Independent WebSphere Application Server Installation in local zones

Shared WebSphere Application Server binary installation for zones

Another installation option is to share the centralized WebSphere Application Server binaries from the Global Zone to multiple local zones, as shown in Figure 5-14 on page 131. This option provides an easier way to standardize one version or Fix Pack level of WebSphere

Application Server. It also eases the Fix Pack maintenance by needing to apply the Fix Pack only once, that is, in the base installation in the Global Zone. Detailed steps for deploying a shared WebSphere Application Server installation is discussed in 5.3.4, “Scenario 4: Share the WebSphere Application Server installation with zones from the Global Zone” on page 137, and steps for Fix Pack maintenance are discussed in Chapter 6, “Management and maintenance of WebSphere Application Server on the Solaris Operating System” on page 169.

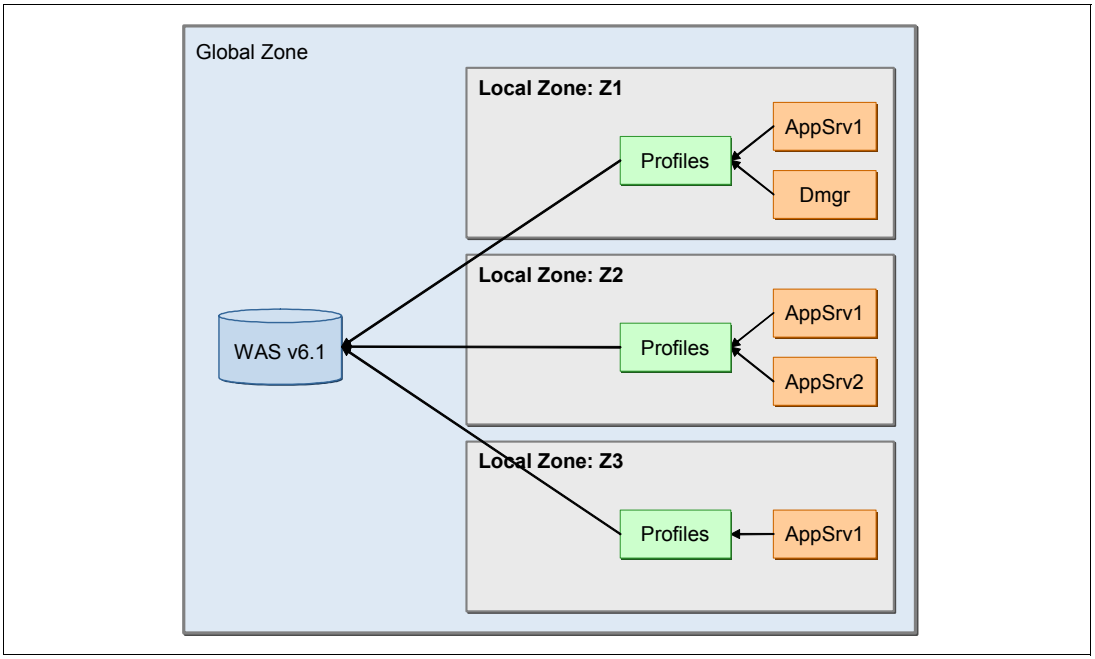


Figure 5-14 Shared WebSphere Application Server binary installation among zones

Zone cloning

The Solaris Zone cloning capability became available as of Solaris 10 11/06.

Zone cloning is a faster way to create new local zones in Solaris 10. It is more than 30% faster in comparison to creating zones from scratch and lets you reproduce any customization or software installation you have done without further manual effort. This does not exclude the fact that the zone must be created and configured before cloning.

The following commands show an example of how a zone can be cloned:

```
global# zonecfg -z new_zone -f new_zone_create_script
global# zoneadm -z new_zone clone source_zone
```

Cloning is a powerful feature that you can leverage to improve efficiency and time in deploying your WebSphere environment. Figure 5-15 on page 132 is a graphical representation of the following steps, demonstrating how quickly you can deploy multiple versions of WebSphere Application Server with different levels of fix packs to provide business agility:

1. You can create a zone, either Whole Root Zone or Sparse Root Zone, as needed. Keep this as your template.
2. Clone the template zone to create a WebSphere Application Server template zone. Install the WebSphere Application Server V6.1 base binaries.

3. Clone the WebSphere Application Server template zone to create a new zone with the base WebSphere Application Server plus a Fix Pack. Now you have another copy of a zone with WebSphere Application Server V6.1 already installed. Apply the desired Fix Pack level to this zone. Let us use Fix Pack 11 as an example. Once completed, you have a zone with WebSphere Application Server V6.1 + FP11. This can serve as a template zone to create new zones to host applications with WebSphere Application Server V6.1.0_11.
4. Similarly, you can repeat this process to have another cloned zone with WebSphere Application Server V6.1 + FP 13. You now have another zone to host applications with WebSphere Application Server V6.1.0_13.
5. If you wish to deploy two zones with WebSphere Application Server V6.1.0_11 and two zones with WebSphere Application Server V6.1.0_13 on this system, you can clone twice, once for each FP level of WebSphere Application Server V6.1.
6. Now, you have four additional zones that have the desired WebSphere Application Server version with the required Fix Pack levels. All you have left to do is bring the zones up and configure the necessary profiles. Once these profiles are federated into the ND cell, you can easily deploy business applications.

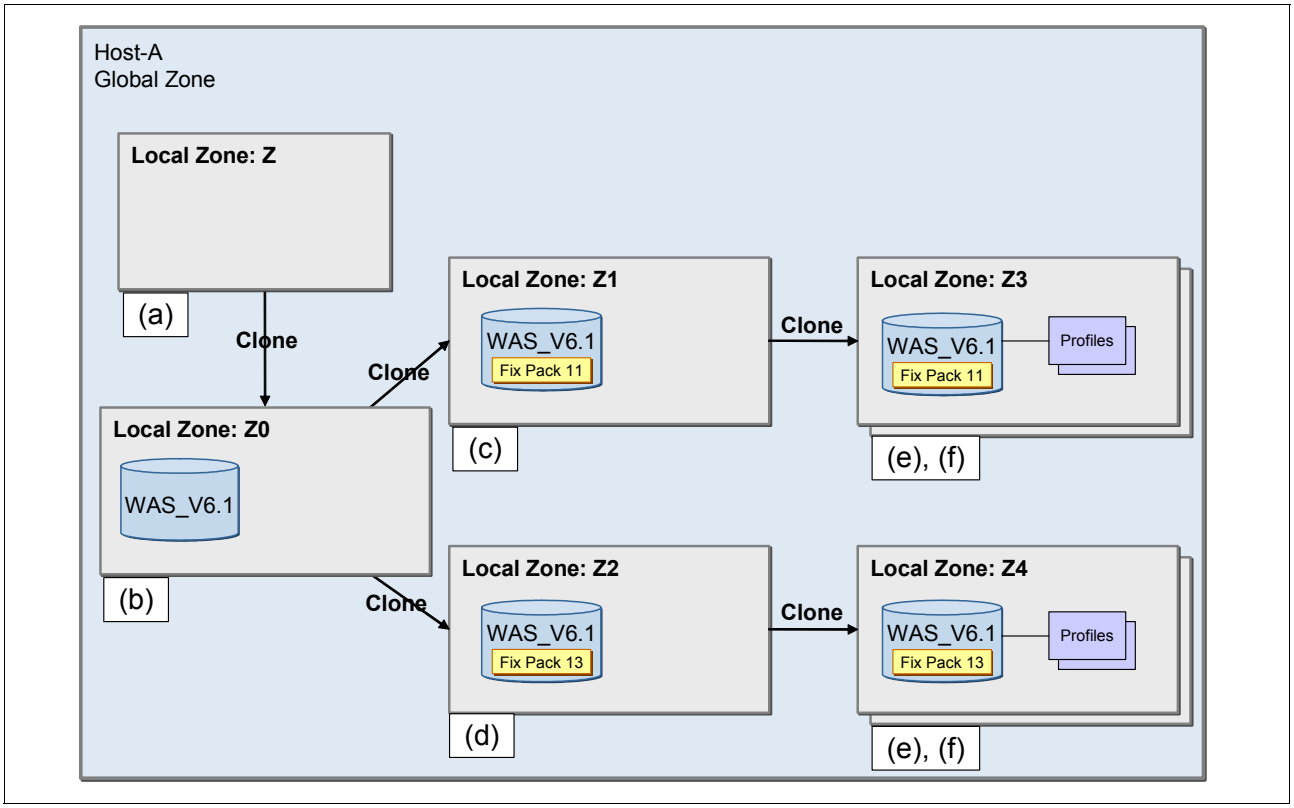


Figure 5-15 Applying zone cloning to a WebSphere Application Server deployment

You can also move a cloned zone to another system, which is described in “Migrating a Zone from one system to another” on page 132.

Migrating a Zone from one system to another

Solaris 10 8/07 adds a new zone feature to let you migrate a zone from one system to another. We recommend you migrate a zone between two systems with the same level of the Solaris OS version and patch level as well as the same type of hardware. However, if your migration takes place between two very different systems, you will have to take special items

into consideration, such as differences in patch levels, packages, network interfaces, and so on. Migrating a zone is a fast and reliable method.

We discuss how WebSphere Application Server zone migration can be accomplished in 6.4, “Rehosting of an existing WebSphere Application Server environment” on page 195.

Additional information is available at:

http://sun.com/software/solaris/howtoguides/moving_containers.jsp

5.2.3 IBM Software licensing

In today’s economy, saving costs is one of the top priorities for enterprises. With rising costs for software, personnel, and infrastructure requirements, such as real estate and electricity, organizations are finding ways to improve operational efficiency and reducing costs for deploying business applications. It is important for you to understand the IBM Software licensing structure based on the Processor Value Units (PVU) and how the sub-capacity licensing can be applied in your application environment to maximize your Return On Investment (ROI).

IBM Software licensing

A Processor Value Unit (PVU) is a unit of measure used to differentiate licensing of middleware on distributed processors and, over time, will evolve to differentiate processor families based on their relative performance, among other factors. The Processor Value Unit structure consists of four broad tiers or levels, and all supported processor families have been assigned to one of those tiers. For each product, customers will need to acquire the appropriate number of Processor Value Units for the level or tier of all processor cores activated and ready for use on which the software is deployed. Figure 5-16 on page 134 shows the PVU table at this writing, which is subject to change. For the most current information, go to:

http://ibm.com/software/lotus/passportadvantage/pvu_licensing_for_customers.html

Table: Processor Value Units (PVUs) per Core[2]

Processor[1] Families			Processor Type					PVUs per Processor Core
Vendor	Brand	Number of cores	One-Core	Dual-Core	Quad-Core	Hexa-Core	Octi-Core	
IBM	POWER6	2		■				120
IBM	POWER5	2		■				100
Fujitsu	SPARC64 VI	2		■				
HP	PA-RISC	2		■				
Intel®	Itanium®	2		■				
Sun	UltraSPARC IV	2		■				
IBM	System z9, eServer zSeries, or System/390	1	■[3]					
Any	Any single core	1	■					
IBM	PowerPC 970	2		■				50
IBM	POWER5 QCM	4			■			
AMD	Opteron	2,4		■	■			
Intel®	Xeon®	2,4		■	■			
Sun	UltraSPARC T2	4,6,8			■	■	■	
IBM	Cell/B.E.™	1	■[4]					30
Sun	UltraSPARC T1	4,6,8			■	■	■	

Notes:

[1] IBM SW defines "Processor" as a Core.

[2] For current generally available processors only (as of Nov 16, 2007). PVU requirements for future processor technologies may differ. For PVU requirements for any processors not listed above, please contact IBM.

[3] For each IFL or CP engine.

[4] Entitlements required for PPE cores only. Cell/Broadband Engine (Cell/B.E.) consists of 1 PPE (Power Processing Engine) & 8 SPE (Synergistic Processing Engine) cores.

Figure 5-16 IBM Software Processor Value Unit table

Sub-capacity licensing

Sub-capacity licensing lets you license an eligible software program for use on less than the full processor core capacity of your machine, when the software program is used with one or more supported virtualization technologies. Without sub-capacity licensing, customers are required to obtain Processor Value Unit license entitlements for all the processor cores in the server, regardless of how the software is deployed.

The IBM sub-capacity offering leverages Processor Value Units (PVUs) to provide the licensing granularity our customers need to leverage multi-core chip technologies and advanced virtualization technologies.

IBM Software Group has made selected middleware programs available for sub-capacity licensing on selected processor technologies and selected virtualization technologies (for example, Solaris Containers and Dynamic System Domains). Lists of the supported products and technologies can be found at:

http://www.ibm.com/software/lotus/passportadvantage/subcaplicensing.html

5.3 Configuration scenarios for WebSphere Application Server in zones

In this section, we describe the potential configuration scenarios and the procedures to accomplish the successful installation and configuration of WebSphere Application Server V6.1 ND in Solaris 10 and zones.

Sample scenarios

Here are some sample scenarios:

1. Install and configure WebSphere Application Server in a Global Zone (a conventional way of installation on Solaris).
2. Install and configure WebSphere Application Server in a Whole Root Zone (an independent WebSphere Application Server installation in a local zone).
3. Install and configure WebSphere Application Server in a Sparse Root Zone (an independent WebSphere Application Server installation in a local zone).
4. Share a WebSphere Application Server installation in zones from the Global Zone.
5. Install IHS in a Non-Global Zone to front end WebSphere Application Server.

5.3.1 Scenario 1: WebSphere Application Server in a Global Zone

To install WebSphere Application Server in a Solaris Global Zone, you need to:

1. Install and configure Solaris, as described in Chapter 2, “Configuring a Solaris environment for WebSphere Application Server” on page 15.
2. Install WebSphere Application Server, as described in 4.1, “Installing WebSphere Application Server V6.1 Network Deployment” on page 52.
3. Create a profile, as described in 4.2.2, “Creating profiles with Profile Management Tool” on page 60.

5.3.2 Scenario 2: WebSphere Application Server in a Whole Root Zone (independent installation)

To install WebSphere Application Server into a Whole Root Zone, you need to:

1. Create a Whole Root Zone with the zone name, as shown in Example 5-1.

Example 5-1 Configuring a Whole Root Zone

```
global# zonecfg -z yourzone
zonecfg:yourzone>create
zonecfg:yourzone>set zonepath=/export/zones/yourzone
zonecfg:yourzone>set autoboot=true
zonecfg:yourzone>remove inherit-pkg-dir dir=/lib
zonecfg:yourzone>remove inherit-pkg-dir dir=/usr
zonecfg:yourzone>remove inherit-pkg-dir dir=/sbin
zonecfg:yourzone>remove inherit-pkg-dir dir=/platform
zonecfg:yourzone>add net
zonecfg:yourzone:net>set physical=’zone_network_interface’
zonecfg:yourzone:net>set address=’zone_ip_address’
zonecfg:yourzone:net>end
zonecfg:yourzone>verify
```

```
zonecfg:yourzone>commit
zonecfg:yourzone>exit
global# zoneadm -z yourzone install
...it takes several minutes...
global# zoneadm -z yourzone boot
...goes through to configure the Solaris Zone's environment...
global# zlogin -C youzone
   .......Enter User ID ad Password ....
#
```

2. Install WebSphere Application Server, as described in 4.1, "Installing WebSphere Application Server V6.1 Network Deployment" on page 52.
3. At this point, if you do not need more zones, skip to step 4. If you need to create more zones, clone the first created Non-Global Zone, as described in "Zone cloning" on page 131.
4. Create a profile, as described in 4.2.2, "Creating profiles with Profile Management Tool" on page 60, in each created zone.

Tip: To make the zone creation and configuration process more efficient, you can use a tool called the Zone Manager.

The Zone Manager is an open source sub-project of OpenSolaris. Its purpose is to simplify the creation and management of Solaris zones. With it, you can create a zone configured the way you want with a single command-line invocation. There are many real world use cases and examples of using the Zone Manager at:

http://thezonemanager.com

To install and configure a WebSphere Application Server instance using the Zone Manager, you need to decide upon the desired configuration attributes of the zone. Then use the corresponding Zone Manager arguments to create your WebSphere Application Server instance.

There are many Zone Manager configuration properties. However, for this example, we will use only the following configuration properties. To see the full set of properties, run the Zone Manager script with the -h flag or see the documentation at the open source project site at:

http://opensolaris.org/os/project/zonemgr/

More specific examples and updates, such as for WebSphere Application Server deployment in a zone using the Zone Manager, can be found at:

http://thezonemanager.com

5.3.3 Scenario 3: WebSphere Application Server in a Sparse Root Zone (independent installation)

To install WebSphere Application Server into a Sparse Root Zone, you need to:

1. Create a Sparse Root Zone with the zone's name.
2. Log into the zone.
3. Install WebSphere Application Server in the desired location.

Example 5-2 on page 137 shows the command needed to accomplish this installation.

Example 5-2 Configuring a Sparse Root Zone

```
global#zonecfg -z yourzone
zonecfg:yourzone>create
zonecfg:yourzone>set zonepath=/export/zones/yourzone
zonecfg:yourzone>set autoboot=true
zonecfg:yourzone>add net
zonecfg:yourzone:net>set physical='zone_network_interface'
zonecfg:yourzone:net>set address='zone_ip_address'
zonecfg:yourzone:net>end
zonecfg:yourzone>verify
zonecfg:yourzone>commit
zonecfg:yourzone>exit
global# zoneadm list -cv
global# zoneadm -z yourzone install
...it takes several minutes...
global# zoneadm -z yourzone boot
...goes through to configure the Solaris Zone's environment...
global# zlogin -C yourzone
yourzone console login: userid
Password:
   .......Enter User ID ad Password ....
#
```

4. Install the WebSphere Application Server in the desired location, as described in 4.1, "Installing WebSphere Application Server V6.1 Network Deployment" on page 52.

5.3.4 Scenario 4: Share the WebSphere Application Server installation with zones from the Global Zone

With shared binaries, you can create isolated WebSphere Application Server environments. While the base WebSphere binaries reside in the Global Zone, your profiles reside in each Non-Global Zone. In this manner, you can create large application server farms with centralized maintenance.

Global Zone preparations and actions

To achieve this goal, follow these steps to install the WebSphere Application Server product in the Global Zone, prepare the WebSphere base installation to allow profiles to reside in zones, configure and install a Non-Global Zone, and create new profile(s) in the zone. To prepare the WebSphere base installation for sharing in zones, you must edit the wasprofiles.properties file:

1. Install the WebSphere Application Server binaries, as described in 4.1, "Installing WebSphere Application Server V6.1 Network Deployment" on page 52, without creating any profiles.

2. In each local zone, assume the WebSphere Application Server profiles will reside in the /opt3/IBMSW/WASprofiles directory. Back up and edit the wasprofiles.properties shown in Example 5-3 and modify the ${was.install.root} parameter with the new directory path /opt3/IBMSW/WASprofiles in three places, as highlighted in the example: WS_CMT_LOG_HOME, WS_PROFILE_REGISTRY, and WS_WSPROFILE_DEFAULT_PROFILE_HOME.

Example 5-3 Editing the wasprofile.properties file

```
global# cd /opt/IBM/WebSphere/AppServer/properties
global# cp -p wasprofile.properties wasprofile.properties.orig
global# vi wasproperties.properties
....
WS_CMT_LOG_HOME=/opt3/IBMSW/WASprofiles/logs/manageprofiles
....
WS_PROFILE_REGISTRY=/opt3/IBMSW/WASprofiles/properties/profileRegistry.xml
....
WS_WSPROFILE_DEFAULT_PROFILE_HOME=/opt3/IBMSW/WASprofiles/profiles
"wasprofile.properties" 83 lines, 4000 characters
```

3. Save the properties file that you just edited.

Non-Global Zone preparations

This section describes the detailed procedure to configure a new zone and its WebSphere profile:

1. Create a script to create a zone. This can be a Sparse Root or Whole Root Zone. Here we use a Sparse Root Zone. Assume that the zone will use the system's network interface e1000g2 with an IP address of 192.168.1.202 and a netmask 255.255.255.0. To share the WebSphere Application Server installation in /opt/IBM/WebSphere from the Global Zone, we need to configure the necessary inheritance of this directory path in the zone creation script, as shown in Example 5-4.

Example 5-4 Zone Creation Script: zoneconfig.txt

```
#
# Script to create WebSphere Application Server Node Zone
create
set zonepath=/export/zones/waszone1
set autoboot=true
add net
set physical=e1000g2
set address=192.168.1.202/24
end
#
# Share the WebSphere Application Server installation from the Global Zone
#
add inherit-pkg-dir
set dir=/opt/IBM/WebSphere
end
verify
commit
# End of Script
```

2. Create, install, and configure the zone:

```
global# zonecfg -z waszone1 -f zoneconfig.txt
```

```
global# zoneadm -z waszone1 install
global# zoneadm -z waszone1 boot
```

Note: At the end of the zone installation, you will receive a message indicating that the installation of two WebSphere packages were skipped. You can ignore this message, as it will not affect your operation.

3. Log into the zone to create the WebSphere Application Server profile:

```
global# zlogin -C waszone1
.......Enter User ID ad Password ....
bash# mkdir -p /opt3/IBMSW/WASprofiles/properties
bash# export DISPLAY=your_workstation_display:N
bash# cd /opt/IBM/WebSphere/AppServer/bin/ProfileManagement
bash# ./pmt.sh
```

Important: Prior to invoking the PMT tool, you must create the properties directory in the same path as you defined for WS_WSPROFILE_DEFAULT_PROFILE_HOME in Example 5-3 on page 138. If you do not create this directory, the PMT tool will fail in this shared installation environment.

For more information about creating a profile with a graphical user interface, refer to 4.2.2, “Creating profiles with Profile Management Tool” on page 60. That topic has all the steps needed to create a profile.

Once the installation is completed, you can run the WebSphere Application Server installation’s First steps application to verify your newly configured environment and start the server. You can also start the server manually from the command line, as shown in Example 5-5.

Example 5-5 Starting the Application Server from the command line

```
bash#cd /opt3/IBMSW/WASprofiles/profiles/AppSrv01/bin/
bash#./startServer.sh server1
```

To verify that your server is up and running, you can go to the WebSphere Application Server admin console through your Web browser (assuming you use the default port 9060) using the following address:

```
http://<appserver_host>:9060/admin
```

To created additional zones, repeat the steps in this section for ““Non-Global Zone preparations” on page 138”.

Important: During the application deployment in this scenario, the **ejbdeploy.sh** tool will attempt to write in the <WAS_HOME>/deploytool/itp/configuration directory. If that occurs, the deployment in the local zone will fail because it has no write permission to this directory, because it is in the global zone. An IBM Technote describes a similar issue and a workaround solution:

http://www-1.ibm.com/support/docview.wss?uid=swg21232460

Based on this Technote's recommendations, modify your local profile's **setupCmdLine.sh** with the ITP_LOC variable for your custom profile directory path. For example, modify the file /opt3/IBMSW/WASprofiles/profiles/AppSrv01/bin/setupCmdLine.sh to produce ITP_LOC=/opt3/IBMSW/WASprofiles/itp.

By setting ITP_LOC, **ejbdeploy.sh** will have write permission to your local profile's deploytool/itp/configuration directory. Information about the WebSphere Application Server maintenance update for this environment is addressed in 6.3.3, "Installing a Fix Pack to the Global Zone" on page 182.

5.3.5 Scenario 5: IBM HTTP Server in a Non-Global Zone to front-end WebSphere Application Server

When installing the IBM HTTP Server, you must consider the fact that IHS relies on the IBM Global Security Kit (GSKit). The GSKit installer requires the creation of symbolic links to a number of required GSKit's binary and shared library files in the /usr/bin and /usr/lib system directories, as shown in Example 5-6, where GSKit is installed in the default location /opt/ibm.

Example 5-6 The GSKit's symbolic linked files

```
/usr/bin/gsk7capicmd -> /opt/ibm/gsk7/bin/gsk7capicmd
/usr/bin/gsk7cmd -> /opt/ibm/gsk7/bin/gsk7cmd
/usr/bin/gsk7ikm -> /opt/ibm/gsk7/bin/gsk7ikm
/usr/bin/gsk7ver -> /opt/ibm/gsk7/bin/gsk7ver

/usr/lib/libgsk7acmeidup.so -> /opt/ibm/gsk7/lib/libgsk7acmeidup.so
/usr/lib/libgsk7cms.so -> /opt/ibm/gsk7/lib/libgsk7cms.so
/usr/lib/libgsk7dbfl.so -> /opt/ibm/gsk7/lib/libgsk7dbfl.so
/usr/lib/libgsk7drld.so -> /opt/ibm/gsk7/lib/libgsk7drld.so
/usr/lib/libgsk7iccs.so -> /opt/ibm/gsk7/lib/libgsk7iccs.so
/usr/lib/libgsk7kicc.so -> /opt/ibm/gsk7/lib/libgsk7kicc.so
/usr/lib/libgsk7kjni.so -> /opt/ibm/gsk7/lib/libgsk7kjni.so
/usr/lib/libgsk7km.so -> /opt/ibm/gsk7/lib/libgsk7km.so
/usr/lib/libgsk7krnc.so -> /opt/ibm/gsk7/lib/libgsk7krnc.so
/usr/lib/libgsk7krrb.so -> /opt/ibm/gsk7/lib/libgsk7krrb.so
/usr/lib/libgsk7krsw.so -> /opt/ibm/gsk7/lib/libgsk7krsw.so
/usr/lib/libgsk7msca.so -> /opt/ibm/gsk7/lib/libgsk7msca.so
/usr/lib/libgsk7p11.so -> /opt/ibm/gsk7/lib/libgsk7p11.so
/usr/lib/libgsk7ssl.so -> /opt/ibm/gsk7/lib/libgsk7ssl.so
/usr/lib/libgsk7sys.so -> /opt/ibm/gsk7/lib/libgsk7sys.so
/usr/lib/libgsk7valn.so -> /opt/ibm/gsk7/lib/libgsk7valn.so
```

If you create a Whole Root Zone to install IHS, the installation procedure is straightforward, as defined in the IBM IHS installation documentation, because Whole Root Zone, by design, has its own /usr directory. The root user of that zone has all the read and write privileges in the /usr within the zone to create the required symbolic links.

If you create a Sparse Root Zone to install IHS, the root user of the zone has no write privileges in the /usr directory; thus, the IHS installer cannot create the symbolic links for the GSKit binary and shared library files. The best known solution today is to first install GSKit in the Global Zone to create the symbolic links in /usr before installing IHS or the WebSphere plug-in in the Sparse Root Zone. This is accomplished by running gskit.sh from within the IHS or WebSphere plug-in installation image.

The maintenance level of GSKit in use is determined by what is installed in the Sparse Root Zone, since no real code exists in /usr, allowing you to control the level of GSKit on a zone by zone basis. If you need more assistance or have any issues, you can open a PMR with IBM Technical Support.

Detailed procedure

Here are the highlights of the IHS configuration in the Sparse Root Zone.

Global Zone

Install the GSKit in Global Zone as follows:

1. Log in to Global Zone as the root user.
2. Locate your WebSphere Application Server installation media, or mount the CD-ROM or DVD-ROM to your system.
3. Go to the GSKit directory and issue the command:

   ```
   global# ./gskit.sh
   ```

   The script installs the GSKit on your Solaris machine. The GSKit is needed by IHS for SSL purposes. It installs library files and binary files into a /opt/ibm/gsk7/lib and /opt/ibm/gsk7/bin directories. It also creates symbolic links into /usr/lib and /usr/bin directories, as shown in Example 5-6 on page 140.
4. As the installation proceeds, you will see information about the propagation of the installation to other zones.
5. When the installation is over, you can proceed with the installation of IHS to the Sparse Root Zone.

Sparse Root Zone

This process assumes that you have the installation media at hand:

1. Create, configure, install, and boot the zone, as described in Example 5-2 on page 137.
2. Log into the zone as root.
3. Locate the WebSphere Application Server install binaries and go to the IHS directory.
4. To start the installation, either:
   - Use silent installation by editing the responsefile and issue the command:

     ```
     bash# ./install -options "response_file_name" -silent
     ```
   - Use the graphical user interface and issue the command:

     ```
     bash# ./install
     ```

   For more information about the installation of IBM HTTP Server, refer to the IBM InfoCenter at:

   http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.ihs.doc/info/ihs/ihs/tihs_installihs.html

Attention: To use the graphical user interface for the installation, you must set the DISPLAY environment variable to point to your X Window System display.

5. If you installed IHS in silent mode, you do not get any messages about whether the installation was successful or unsuccessful. After the installation is complete, you can verify a successful installation by looking the log.txt file in <ihs-root>/logs/install directory. Look for the `INSTCONFSUCCESS` indicator at the end of the log.txt file.

   You can also verify the installation by running the **`verifyinstallver.sh`** script in the <ihs-root>/bin directory.

From here, you can continue to configure the WebSphere plug-in within IHS to front-end WebSphere Application Server as needed.

Attention: If you install the IHS plug-in for the "WebSphere Application Server machine (local)" scenario in a shared WebSphere Application Server installation from the Global Zone, as described in "Shared WebSphere Application Server binary installation for zones" on page 130, you will get an error message that there is no write privilege. The IHS plug-in installer is not aware of your custom profile directory, but rather it is looking for the profile directory in the ${was.install.root} directory (for example, /opt/IBM/WebSphere/AppServer/profiles). This directory is in the Global Zone and you are running the IHS plug-in installer in a local zone.

A workaround solution is to select **Web server machine (remote)** even though your IHS plug-in installation scenario has a local app server in the zone.

5.4 Resource management

It is essential to manage various workloads that exist on the system to better utilize the available resources. A workload consists of one or more processes to perform a business function. For example, a WebSphere Application Server workload includes Java processes and SSL processing. A resource can be defined as any part of the underlying system's facility to support the computing activities, such as CPU and memory. Resource management ensures that these workloads utilize the system's resources in a more controlled and predictable manner by enforcing limits or partitioning resources for different workloads.

The resource management capabilities in Solaris have evolved from simple processor partitioning and binding to sophisticated virtualized operating environments. In Solaris 2.6. Sun introduced the concept of isolating process execution to a specific processor or a group of processors known as *processor sets*. In Solaris 9, Sun introduced the Solaris Resource Manager, which provides finer granularity of resource controls, including Fair-Share Scheduler (FSS). In Solaris 10, Sun added more capabilities, such as Dynamic Resource Pools, which are an integral part of Solaris Containers (Zone + Resource Management). As of the Solaris 10 08/07 release, zone configuration has more capabilities for binding a zone to a set of dedicated CPUs, capping memory (physical or swap), and selecting FSS as the default scheduler.

In 2.1.4, “Tunable parameters in Solaris 10” on page 20, we discuss Solaris tuning parameters in relation to resource controls for project and process, which are the minimum resource allocations that you need to set as a baseline requirement in compliance with the IBM recommended parameters for WebSphere Application Server. Detailed information can be found at:

http://docs.sun.com/app/docs/doc/817-1592/6mhahuoh2

Figure 5-17 shows a system where multiple processes are processing different workloads without resource management. When “App 3” starts using more system resources, such as increased CPU utilization or requiring more memory as the application continues to have memory leak, it begins to cause resource starvation in other processes on the system. On the right side of the figure, the system operates more steadily when resource boundaries are imposed on these processes for fair utilization. Therefore, when a process attempt to utilize system resources, it cannot go beyond the permitted threshold.

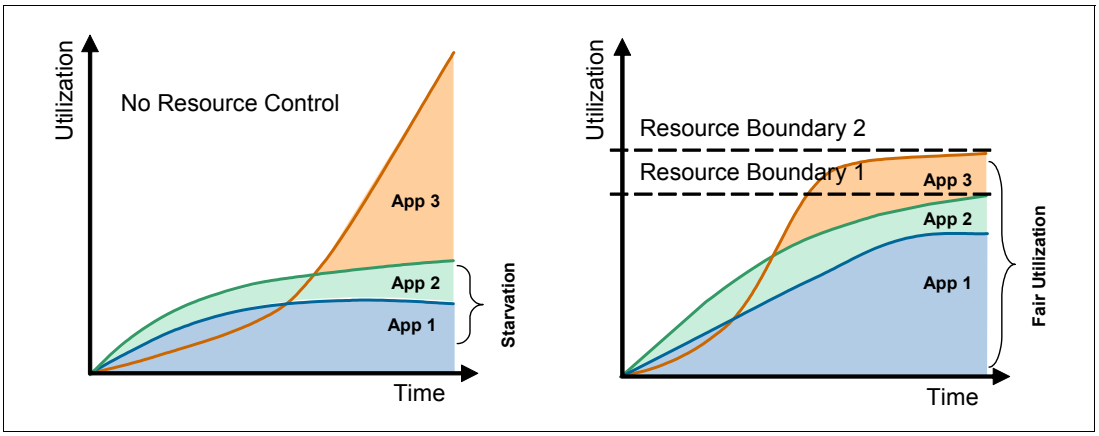


Figure 5-17 Before and after applying resource management

5.4.1 Scope of resource control

Resource controls can be assigned to different resources at different level of scopes within a system: system wide, zone wide, task wide, project wide, and on a local process. A resource control can be put on a resource at each level, as shown in Figure 5-18. When these resource controls are active at each level, the lowest level's resource control has precedence over the higher level. For example, if a resource control on CPUs are created at the process level and the zone level, the process level resource control is enforced before the zone level resource control.

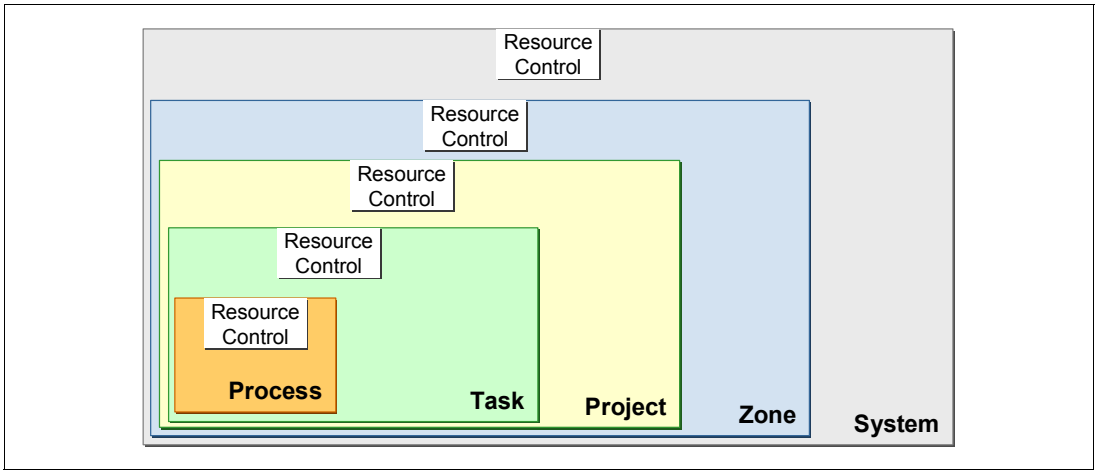


Figure 5-18 Scopes of resource control in Solaris

5.4.2 Resource control mechanisms

The current Solaris system has three types of resource management control mechanisms:

1. *Partitioning mechanisms* allow available system resources to be divided among various workloads. The result is a workload that can use only a subset of a resource at any given time. For example, you can allocate four out of 32 processors to a workload, such as WebSphere Application Server. This is called binding a workload to a subset of the system's available resources.
2. *Constraint mechanisms* allow setting limitations on available system resources to prevent certain workloads from unwanted or excessive consumption. An example is to limit the core file size of an application so that when it terminates abnormally, it could dump a core file that is no bigger than the defined value.
3. *Scheduling mechanisms* allow Solaris to make dynamic allocations of resources based upon a predictable algorithm. The dynamic allocations can be based on minimum and maximum bounds as well as certain policy. Solaris has various scheduling classes based on different algorithms: real time, interactive, fixed priority, timesharing, and fair share scheduling (FSS) class.

   For more information about these Solaris resource control mechanisms, go to:

   `http://docs.sun.com/app/docs/doc/817-1592/rmintro-15?a=view`

5.4.3 Applying resource management

In this section, we explain how to apply resource controls to manage your WebSphere Application Server workload and to maximize the utilization of the system.

Solaris processor sets

Processor sets are a Solaris partitioning mechanism that allows one or more processes to be assigned to a specific group of processors.

Some possible uses for processor sets:

- To dedicate a group of CPUs to a particular workload as a resource guarantee to meet a service level agreement (SLA).
- To efficiently allocate a system's resources when using an application that does not intrinsically scale beyond a certain processor count, by limiting an individual application instance to a small number of number of CPUs. For example, if a WebSphere Application Server process has diminishing scalability beyond four CPUs, you can create two WebSphere Application Server instances and bind each to its own processor set of four CPUs each.
- To dynamically reallocate processors among processor sets, to handle a particular workload at a specific time, such as spikes at trade closing time.

More information about processor sets is available at:

http://developers.sun.com/solaris/articles/solaris_processor.html

We provide you with some examples here to demonstrate how a processor set can be applied in your WebSphere Application Server deployment environment.

First, you must determine the processor's availability on your system. Example 5-7 shows the **psrinfo** command, which queries the status of all available processors on your Solaris system. A processor can be in one of several conditions, including on-line, off-line, or no-intr.

Example 5-7 Examining the processor status on a system

```
global# psrinfo
0       on-line   since 07/23/2007 17:35:21
1       on-line   since 07/23/2007 17:35:21
2       on-line   since 07/23/2007 17:35:21
3       on-line   since 07/23/2007 17:35:20
16      on-line   since 07/23/2007 17:35:21
17      on-line   since 07/23/2007 17:35:21
18      on-line   since 07/23/2007 17:35:21
19      on-line   since 07/23/2007 17:35:21
global# psrset -p
.... (on an initial set up, it will report no processor set)...
```

The **psrinfo** query shows the available processors with their status (on-line) and the **psrset -p** command shows that no processor sets exist on the system. In Example 5-8, we create a new processor set with processors 16 and 17. Each processor set is automatically given a unique identifier number. In this case, the ID is 1.

Example 5-8 Create a processor set with processor 16 and 17

```
# psrset -c 16 17
created processor set 1
processor 16: was not assigned, now 1
processor 17: was not assigned, now 1
```

Example 5-9 examines the status of the processor sets on the system and shows that processors 16 and 17 are bound to processor set 1.

Example 5-9 Query the status of processor sets

```
# psrset -p
processor 16: 1
processor 17: 1
```

Example 5-10 shows how to bind a process, such as a WebSphere Application Server Java process, to a processor set. You can bind one or more processes to a processor set.

Example 5-10 Bind WebSphere process to processor set 1

```
# ps -ef | grep WebSphere | grep -v grep
    root 15114     1   0   Aug 14 ?         278:30
/opt/IBM/WebSphere/AppServer/java/bin/java -XX:MaxPermSize=256m -Dwas.status
# psrset -b 1 15114
process id 15114: was not bound, now 1
```

If you query the processor status, as shown in Example 5-11, you will find that processor 16 and 17 are dedicated to this process and will no longer process system I/O or network interrupts until they are removed from the processor set and return back to the system.

Example 5-11 Verify the processor status

```
# psrinfo
0       on-line   since 07/23/2007 17:35:21
1       on-line   since 07/23/2007 17:35:21
2       on-line   since 07/23/2007 17:35:21
3       on-line   since 07/23/2007 17:35:20
16      no-intr   since 12/05/2007 18:54:02
17      no-intr   since 12/05/2007 18:54:02
18      on-line   since 07/23/2007 17:35:21
19      on-line   since 07/23/2007 17:35:21
```

If the WebSphere Application Server process needs two additional processors, you can add them to the processor set, as shown in Example 5-12.

Example 5-12 Add two more processors (18 and 19) to processor set 1

```
# psrset -a 1 18 19
processor 18: was not assigned, now 1
processor 19: was not assigned, now 1
```

Attention: As you add more processors to a processor set, the processes bound to this processor set will dynamically be able to recognize and use the added processors. The reverse is true for removing processors.

Example 5-13 shows how to remove processors from a processor set.

Example 5-13 Remove two processors (16 and 17) from processor set 1

```
# psrset -r 16 17
processor 16: was 1, now not assigned
processor 17: was 1, now not assigned
```

Example 5-14 shows how to delete the processor set. Any processes bound to this processor set will automatically begin to utilize the available processors on the system.

Example 5-14 Delete processor set 1

```
# psrset -d 1
removed processor set 1
```

A processor set created using the method shown in this section does not persist after a system reboot. Alternatively, you can use resource pools or zone configuration for persistent sets, which we discuss in the following sections.

Resource pools

Resource pools provide you with the ability to work with processor sets and Solaris scheduling classes, as discussed in 5.4.2, “Resource control mechanisms” on page 144. When defining processor sets through the resource pool facility, you no longer have to use the physical ID of the processors, as shown in “Solaris processor sets” on page 145. You just have to define how many processors you wish to allocate. You can also choose a specific scheduler, such as Fair Share or Fixed Priority scheduler (FSS or FX), in conjunction with the processor sets for the resource pool.

For more information about Resource Pools, go to:

http://docs.sun.com/app/docs/doc/817-1975/6mhlv6vvk?a=view

Example 5-15 demonstrates how you can apply the resource pools to manage the processor set for your WebSphere Application Server application process.

1. Configure a processor set, called pset4_zone1, with between two and four processors.
2. Create a resource pool called was_pool1.
3. Make the processor set a part of this resource pool.

Example 5-15 Creating a processor set and associating it with a created resource pool

```
global# pooladm -e
global# pooladm -s
global# poolcfg -c ‘create pset was_pset1 (uint pset.min = 2; uint pset.max = 4)’
global# poolcfg -c ‘create pool was_pool1’
global# poolcfg -c ‘associate pool was_pool1 (pset was_pset1)’
```

4. Instantiate the newly created resource by issuing the following command:

   ```
   global# pooladm -c
   ```

   Tip: To verify the resource, you can query your pool configuration with the following command:

   ```
   global# poolcfg -dc info
   ```

5. Once you have this resource pool created, you can apply it to various purposes:
   - You can bind one or more processes to utilize this pool.
   - You can associate the pool to a project that owns a group of processes.
   - You can associate the pool to a zone (shown in step 7).

   More information is available at:

   http://docs.sun.com/app/docs/doc/817-1592/rmconfig-3

6. At this point, one way to apply the resource pool is to bind it to a process or a project.

```
global# poolbind -p was_pool1 ${WAS_PID}
```

Tip: If you no longer wish to associate the process to this pool, you simply bind it to the system's default pool:

```
global# poolbind -p pool_default ${WAS_PID}
```

7. You can also tie this resource pool to a zone. Assume you have already created a zone called was_zone1. You can now assign the resource pool to this zone, as shown in Example 5-16.

Example 5-16 Assigning a resource pool to a zone

```
global# zonecfg -z was_zone1
zonecfg:was_zone1> set pool=was_pool1
zonecfg:was_zone1> verify
zonecfg:was_zone1> commit
zonecfg:was_zone1> exit
```

Tip: If you no longer wish to associate the zone to this pool, you need to change the zone configuration as follows:

```
global# zonecfg -z was_zone1
zonecfg:was_zone1> set pool=""
zonecfg:was_zone1> verify
zonecfg:was_zone1> commit
zonecfg:was_zone1> exit
```

Dynamic Resource Pool

In Solaris 10, a new capability called Dynamic Resource Pool has been introduced. We discuss how resource pools can be created and applied. You also need to note that the subset of these system's resources allocated to the resource pools can be changed dynamically. Traditionally, it is the system administrator's responsibility to transfer resources from one resource pool to another one. This can be time consuming and prone to errors. Dynamic Resource Pool automates the resource transfer process by enabling a conditional mechanism. Assume you have two resource pools named was_pool1 and db_pool1 with processor sets. You need to ensure that you keep the processor utilization below 60%. When a WebSphere Application Server process assigned to was_pool1 exceeds the resource boundary, the system borrows the necessary processors from db_pool1, providing it has enough capacity. This can be achieved with the following set of commands:

```
# poolcfg -dc 'modify pset was_pool1 (string pset.poold.objectives="utilization<60")'
# poolcfg -dc 'modify pset db_pool1(string pset.poold.objectives="utilization<60")'
```

Resource control for Solaris Zones

In Solaris 10 8/07, new features were added to the zone configuration command. If you need to define resource control at the zone level, you can accomplish this in a much simpler way by using the **zonecfg** command during zone creation or to modify the zone's configuration at a later time. Example 5-17 on page 149 shows how you can dedicate a number of CPUs to a zone. This mechanism hides all the commands that you have to use in "Solaris processor sets" on page 145 and "Resource pools" on page 147.

Example 5-17 Setting dedicated CPUs to a zone

```
global# zonecfg -z yourzone
zonecfg:yourzone>add dedicated-cpu
zonecfg:yourzone:dedicated-cpu>set ncpus=2
zonecfg:yourzone:dedicated-cpu>set importance=2
zonecfg:yourzone:dedicated-cpu>end
zonecfg:yourzone>verify
zonecfg:yourzone>commit
zonecfg:yourzone>exit
```

Example 5-18 shows how you can cap physical and virtual memory to certain limit.

Example 5-18 Setting capped memory to a zone (physical to 1 GB and virtual to 1 GB)

```
global# zonecfg -z yourzone
zonecfg:yourzone> add capped-memory
zonecfg:yourzone:capped-memory> set physical=1g
zonecfg:yourzone:capped-memory> set swap=1g
zonecfg:yourzone:capped-memory> end
zonecfg:yourzone> exit
```

Example 5-19 shows how to select Fair-Share Scheduler (FSS) with 10 "CPU" shares for a zone. FSS is a Solaris scheduling class and is described in 5.4.2, “Resource control mechanisms” on page 144. It allocates and distributes the available CPU resources among workloads based on their importance. The "CPU" shares are used to define the importance where the more the shares, the higher the importance is. In this case, the zone's processes will have 10 shares of "CPU" of the system.

Example 5-19 Setting capped memory to a zone (Physical to 1GB and Virtual to 1GB)

```
global# zonecfg -z yourzone
zonecfg:yourzone> set scheduling-class=FSS
zonecfg:yourzone> set cpu-shares=10
zonecfg:yourzone> set max-lwps=800
zonecfg:yourzone> end
zonecfg:yourzone> exit
```

More information about the **zonecfg** command’s new features is available at:

http://docs.sun.com/app/docs/doc/817-1592/6mhahuooo

Note: These are just few examples of Solaris Resource Management capabilities. There are many other possibilities that can be created by applying different combinations of this technology so that you can better manage your workloads and meet your business service level agreements (SLAs).

5.5 Service Management Facility (SMF)

The Service Management Facility (SMF) introduced with Solaris 10 both unifies and extends the traditional method of starting services in UNIX. Not only does it provide automated recovery from software and hardware failures, SMF can specify user attributes, privileges, dependencies, and resource limits as a part of the definition of the service. The result is better security, more complete resource accounting, and higher availability.

5.5.1 SMF core concepts

SMF is comprised of various core components that are described in Figure 5-19. They are:

- Service states
- SMF repository
- SMF daemons, utilities, and commands
- Service manifest

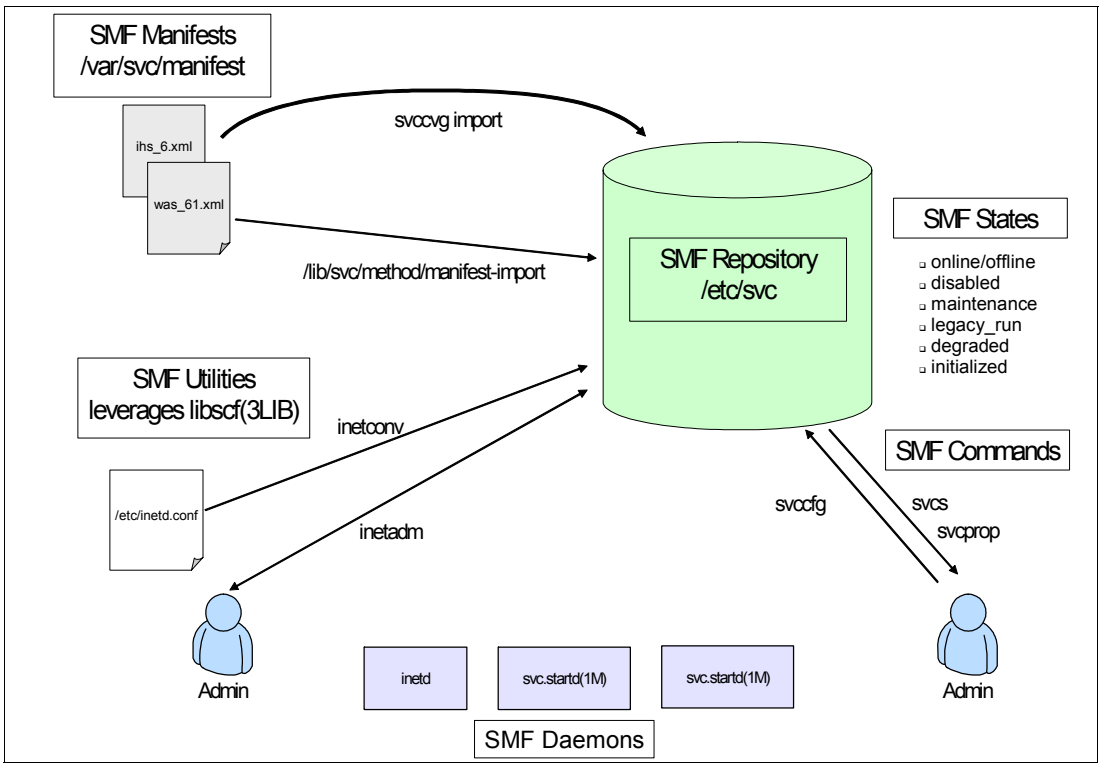


Figure 5-19 Solaris SMF core components

Service states

SMF defines seven services states as follows:

1. Online: An enabled service that has all of its dependencies satisfied. The service has started successfully.
2. Offline: An enabled service that has some dependencies that are not satisfied. Use **`svcs -x`** or **`svcs -l`** to determine which dependency or dependencies are not satisfied.
3. Disabled: A service that has been turned off.
4. Maintenance: A service that has either failed a method (returned a non-zero exit status), started and stopped too quickly, exceeded a method time out, or had insufficient

credentials to run the method. Use **svcs -x**, **svcs -l**, or look in the service log file to determine why the service transitioned into the maintenance state.

5. Legacy_run: The state of a System V Release 4 (SVR4) earlier RC script. These scripts are run as root by svc.startd when the system transitions between run levels.
6. Degraded: A service method has exited with the SMF_DEGRADE exit status.
7. Initialized: **svc.startd** cannot access the repository. This happens normally at zone boot time while the sysidtool and manifest_import services are running.

SMF repository

The SMF repository is a copy of all of the services and their associated properties. The repository is currently kept on disk in /etc/svc/repository.db (persistent properties) and /etc/svc/volatile (transient service properties). Users can access the repository using the commands **svcs(1)**, **svcprop(1)**, and **svccfg(1m)**. Applications can use the public APIs in libscf(3C).

SMF daemons

svc.startd performs most of the functions formerly performed by **init**. Rather than being driven by a set of scripts in a directory, **svc.startd** manages the service dependencies in the repository. **svc.startd** stops and starts services based on changes in each of the service dependencies. **svc.startd** also provides a common view of service states.

Unlike **init**, **svc.startd** is completely restartable.

svc.configd manages the repository. It is started by **svc.startd** and is restarted automatically if SMF is restarted.

Service manifest

A service manifest is a complete description of the service. This includes the method used to start, stop, and refresh the service as well as more advanced properties, such as dependencies, authorizations, resource control, and service specific properties.

Manifests are written in XML and generally placed in /var/svc/manifest. Manifests are compiled into a binary format and placed into the repository. This can be done manually by running **svccfg import** or automatically at boot time by the manifest_import service.

Solaris defines seven default categories of services, each corresponding to a directory in /var/svc/manifest:

1. application: One time boot services or standalone daemons. Formerly SVR4 RC scripts.
2. network: inetd converted plus some transient network services.
3. milestone: Similar to SVR4 run levels, but the transition happens at the end of the run level.
4. device: Used primarily for resolution of device specific dependencies.
5. platform: Platform specific system services.
6. system: Platform independent system services.
7. site: Reserved for customer use.

5.5.2 Creating an SMF service definition

Since WebSphere Application Server is a stand-alone daemon, we will place it in the application category. The name of the service will be svc://localhost/application/was61:was_server1, and by convention the corresponding service manifest will be named /var/svc/manifest/application/was61.xml. See Example 5-20 for more information.

Example 5-20 The Service Manifest for WebSphere Application Server 6.1 was61.xml

```
<?xml version='1.0'?>
<!DOCTYPE service_bundle SYSTEM '/usr/share/lib/xml/dtd/service_bundle.dtd.1'>
<service_bundle type='manifest' name='export'>
  <service name='application/was61' type='service' version='1'>
    <instance name='was_server1' enabled='false'>
      <dependency name='network' grouping='require_all' restart_on='error' type='service'>
        <service_fmri value='svc:/milestone/network:default'/>
      </dependency>
      <dependency name='filesystem-local' grouping='require_all' restart_on='none'
type='service'>
        <service_fmri value='svc:/system/filesystem/local:default'/>
      </dependency>
      <dependency name='autofs' grouping='optional_all' restart_on='error' type='service'>
        <service_fmri value='svc:/system/filesystem/autofs:default'/>
      </dependency>
<!-- "svcadm enable was61" to start WebSphere Application Server using a custom-defined
method -->
      <exec_method name='start' type='method' exec='/lib/svc/method/svc-was61 start'
timeout_seconds='60'>
        <method_context/>
      </exec_method>
<!-- "svcadm disable was61" to stop WebSphere Application Server using a custom-defined
method -->
      <exec_method name='stop' type='method' exec='/lib/svc/method/svc-was61 stop'
timeout_seconds='60'>
        <method_context/>
      </exec_method>
<!-- "svcadm refresh was61" to bounce WebSphere Application Server using a custom-defined
method -->
      <exec_method name='refresh' type='method' exec='/lib/svc/method/svc-was61 refresh'
timeout_seconds='60'>
        <method_context/>
      </exec_method>
    </instance>
    <stability value='Evolving'/>
    <template>
      <common_name>
        <loctext xml:lang='C'>WebSphere Application Server v6.1</loctext>
      </common_name>
      <documentation>
        <doc_link name='ibm.com'
uri='http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp'/>
      </documentation>
    </template>
  </service>
</service_bundle>
```

You can also add other dependencies in the service manifest that the WebSphere Application Server services rely upon (for example, another stand-alone application that your application requires to be co-located on the same system or a JMS provider (for example, WebSphere MQ) that must be started before the WebSphere Application Server services, and so on).

See the **service_bundle(5)** man page for more information about the contents of a service manifest.

Note: If you have multiple instances of the application server, you can have separate manifests or a single manifest for all instances. Separate manifests are convenient if you have multiple administrators making changes. In that case, the recommended file name would be /var/svc/manifest/application/was61-<*instance*>.xml. For example, you can use a name like /var/svc/manifest/application/was61-was_server1.xml.

The next step is to provide a set of methods that start, stop, and optionally refresh the application server. These are derived from the SVR4 earlier RC scripts with very few modifications needed.

Example 5-21 is an example of a single method used for both starting and stopping the application server. By convention, the methods are in /lib/svc/method. In a sparse root Non-Global Zone, this would not be writable; therefore, you must choose another location.

See the **smf_method(5)** man page for more information about the execution environment for SMF methods.

Example 5-21 The Service Method file "svc-was61"

```
#!/usr/bin/sh
#
#  SMF Method file for WebSpehre Application Server
#  This file should reside in /lib/svc/method. This method
#  need to be included in the service manifest for WebSphere Application Server and
#  invoked through SMF.
#       (e.g. /lib/svc/method/svc-was61)
#
#  --   Albert Leigh April 10, 2007
#

. /lib/svc/share/smf_include.sh

#
# Replace these where your WebSphere Application Server profile is located
WAS_DIR="/opt/WASProfiles/profiles/AppSrv01"
#
# Replace this server name if it is different
SERVER_NAME="server1"
#
#
WAS_BIN="${WAS_DIR}/bin"
START_WAS="${WAS_BIN}/startServer.sh"
STOP_WAS="${WAS_BIN}/stopServer.sh"


case $1 in
'start')
   if [ -x "$START_WAS" ]; then
            $START_WAS $SERVER_NAME
        else
```

```
        echo "'$START_WAS' is missing or not executable"
        exit $SMF_EXIT_ERR_CONFIG
   fi
     ;;
'stop')
   if [ -x "$STOP_WAS" ]; then
          $STOP_WAS $SERVER_NAME
       else
       echo "'$STOP_WAS' is missing or not executable"
       exit $SMF_EXIT_ERR_CONFIG
   fi
   ;;
'refresh')
     stop
     sleep 30
     start
     ;;
*)
       echo "Usage: $0 { start | stop | refresh }"
       exit 1
       ;;
esac

exit $SMF_EXIT_OK
#
#-------- End of Script --------
```

Make sure the method is executable by running the following command:

```
# chmod 755 /lib/svc/method/svc-was61
```

Import the service manifest into the SMF repository, and enable and verify the service as follows:

```
# svccfg import /var/svc/manifest/application/was61.xml
# svcadm enable was61
# svcs -l was61
fmri            svc:/application/was61:was_server1
name            WebSphere Application Server v6.1
enabled         true
state           online
next_state      none
state_time      Thu Jun 14 18:23:17 2007
logfile         /var/svc/log/application-was61:was_server1.log
restarter       svc:/system/svc/restarter:default
contract_id 1819
dependency      require_all/error svc:/milestone/network:default (online)
dependency      require_all/none svc:/system/filesystem/local:default (online)
dependency      optional_all/error svc:/system/filesystem/autofs:default (online)
```

Once the service is enabled, WebSphere Application Server should start if the configuration is valid. The log file is located at /var/svc/log/application-was61:was_server1.log.

5.5.3 Running WebSphere Application Server as a non-root user

One important feature of SMF is the ability to start a service as a non-privileged user. This provides greater security and is a recommended practice. To run WebSphere Application Server as a regular user, perform a non-root installation of WebSphere Application Server and change the exec_method tags in the manifest to include a method_context specification. Example 5-22 is a complete sample manifest for starting WebSphere Application Server as user wasuser1.

Example 5-22 The Service Manifest for WebSphere Application Server V6.1 was61.xml with non-root user

```
<?xml version='1.0'?>
<!DOCTYPE service_bundle SYSTEM '/usr/share/lib/xml/dtd/service_bundle.dtd.1'>
<service_bundle type='manifest' name='export'>
  <service name='application/was61' type='service' version='1'>
    <instance name='was_server1' enabled='false'>
      <dependency name='network' grouping='require_all' restart_on='error' type='service'>
        <service_fmri value='svc:/milestone/network:default'/>
      </dependency>
      <dependency name='filesystem-local' grouping='require_all' restart_on='none'
type='service'>
        <service_fmri value='svc:/system/filesystem/local:default'/>
      </dependency>
      <dependency name='autofs' grouping='optional_all' restart_on='error' type='service'>
        <service_fmri value='svc:/system/filesystem/autofs:default'/>
      </dependency>
<!-- "svcadm enable was61" to start WebSphere Application Server using a custom-defined
method -->
      <exec_method name='start' type='method' exec='/home1/wasuser1/SMF/svc-was61 start'
timeout_seconds='60'>
      <method_context working_directory='/home1/wasuser1'>
              <method_credential user='wasuser1' group='wasgrp1'
privileges='proc_fork,proc_exec'  />
      </method_context>
      </exec_method>
<!-- "svcadm disable was61" to stop WebSphere Application Server using a custom-defined
method -->
      <exec_method name='stop' type='method' exec='/home1/wasuser1/SMF/svc-was61 stop'
timeout_seconds='60'>
      <method_context working_directory='/home1/wasuser1'>
              <method_credential user='wasuser1' group='wasgrp1'
privileges='proc_fork,proc_exec'  />
      </method_context>
      </exec_method>

<!-- "svcadm refresh was61" to bounce WebSphere Application Server using a custom-defined
method -->
      <exec_method name='refresh' type='method' exec='/home1/wasuser1/SMF/svc-was61
restart' timeout_seconds='60'>
      <method_context working_directory='/home1/wasuser1'>
              <method_credential user='wasuser1' group='wasgrp1'
privileges='proc_fork,proc_exec'  />
      </method_context>
      </exec_method>
    </instance>
    <stability value='Evolving'/>
    <template>
      <common_name>
        <loctext xml:lang='C'>WebSphere Application Server v6.1</loctext>
```

```
      </common_name>
      <documentation>
        <doc_link name='ibm.com'
uri='http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp'/>
      </documentation>
    </template>
  </service>
</service_bundle>
```

You can find more information about this topic in *Restricting Service Administration in the Solaris 10 Operating System* by Brunette, found at:

http://sun.com/blueprints/0605/819-2887.pdf

Tip: For collaboration with others, you can join the OpenSolaris community for SMF at:

http://www.opensolaris.org/os/community/smf

5.5.4 Enabling autostart of WebSphere Application Server after system boot

If you wish to have the WebSphere Application Server service started automatically upon system boot, you can set the service instance to `enabled='true'`. For example, when you are creating a new WebSphere Application Server service, you need to change the 5th line in the service manifest file, `was61.xml`, as shown in Example 5-21 on page 153 (root user) or Example 5-22 on page 155 (non-root user) while following the procedure in 5.5.2, "Creating an SMF service definition" on page 152.

Before:

```
<instance name='was_server1' enabled='false'>
```

After:

```
<instance name='was_server1' enabled='true'>
```

If you have already created the WebSphere Application Server service with `enabled='false'`, do the following command to enable the service instance named `was61`:

```
 # svcadm enable was61
```

You can verify the status of the service to make sure that enabled is `true` and the state is `online` as follows:

```
# svcs -l was61
fmri         svc:/application/was61:was_server1
name         WebSphere Application Server v6.1
enabled      true
state        online
next_state   none
state_time   Mon Jan 24 13:31:25 2008
logfile      /var/svc/log/application-was61:was_server1.log
restarter    svc:/system/svc/restarter:default
contract_id  2748
dependency   require_all/error svc:/milestone/network:default (online)
dependency   require_all/none svc:/system/filesystem/local:default (online)
dependency   optional_all/error svc:/system/filesystem/autofs:default (online)
```

5.5.5 Removing a service from SMF

To remove the service, stop it and delete it from the repository. If your manifest is in /var/svc/manifest, then it should be removed as well or it will be re-imported upon the next boot:

```
# svcadm disable was61
# svccfg delete was61
# rm -i /var/svc/manifest/application/was61.xml
```

5.6 Process Rights Management

Process Rights Management is based on the Least Privilege model and is available in Solaris 10 OS. Most services in Solaris run as root (user ID 0) because they need to perform a few privileged operations, such as creating a listener on a port less than 1024. Some of these services may fork child processes using a non-root user, but the parent process, which normally runs from a shell environment at boot time, can still be compromised and the results can be catastrophic.

Solaris 10 introduces a set of fine grained privileges that can be granted to processes needing to perform tasks normally reserved for user ID 0. It is even possible to configure Solaris such that the root user is a role (a non-login account) and has restricted capabilities.

Currently, Solaris 10 features more than 60 privileges. A regular user is granted with the following privileges by default by Solaris:

- file_link_any: Can create a link to an object not owned by the user
- proc_fork: Can create additional processes (subject to resource management limits)
- proc_exec: Can use the exec(2) system call to execute a different program than is currently executing
- proc_info: Can see the status of any process other than its own children
- proc_session: Allows a process to see other processes outside of its own session

These privileges were previously granted to any process; thus in Solaris 10, they are part of the basic set that all processes are granted by default. Additional privileges may be added or taken away as necessary. The defaults for a specific user can be specified in the extended user attributes database of /etc/user_attr. The defaults for a specific application can be specified as a execution profile in /etc/security/exec_attr.

Consider the example in Example 5-23. You can use the **ppriv** command to examine the effective process privileges of the current shell session denoted as $$ in Solaris.

Example 5-23 Examining the user's shell effective process privileges

```
% ppriv $$
E: basic
     I: basic
     P: basic
     L: all

% cat /etc/shadow
cat: cannot open /etc/shadow: Permission denied
```

You can also use **ppriv(1)** in debug mode to determine which privileges are required to perform the requested operation, as shown in Example 5-24.

Example 5-24 Debugging with the ppriv command

```
% ppriv -eD cat /etc/shadow
   cat[9089]: missing privilege "file_dac_read" (euid = 23234, syscall = 225)
needed at ufs_iaccess+0xe1
   cat: cannot open /etc/shadow: Permission denied
```

If we were to grant the requesting processes file_dac_read (which bypasses discretionary file access control for reading), the process could read the shadow password file (and any other file on the system). While still a powerful privilege, the requesting process would not be able to do other privileged operations, such as changing process execution classes, bypassing file write permissions, or interacting with processes that it does not own.

You can grant file_dac_read in /etc/user_attr, as shown in Example 5-25.

Example 5-25 The /etc/user_attr file

```
% grep fred /etc/user_attr
fred::::type=normal;defaultpriv=basic,file_dac_read

% ppriv $$
9118:   -bash
flags = <none>
     E: basic,file_dac_read
     I: basic,file_dac_read
     P: basic,file_dac_read
     L: all

% grep bin /etc/shadow
bin:NP:6445::::::
```

Debugging individual privileges for a network service can be rather tedious, although there are some general guidelines that can help.

Most start methods will require at the very least proc_fork and proc_exec. They may require other privileges from the default basic set. If the service wants to create a listener on a port less than 1024, it will also require net_privaddr. The start and refresh methods will typically require proc_fork, proc_exec, proc_session and proc_info.

File permissions (write for log files, execute for scripts) can also be a problem. Fortunately, these are not a problem with a non-root installation of the application server.

To determine what privileges are actually required by a method, you can look at the privdebug.pl Perl script available at the OpenSolaris Security Community Web site at http://opensolaris.org/os/community/security/file. This Perl script was developed by Glenn Brunette of Sun Microsystems. It produces a list of all of the privileges a process requests along with the status (granted, failed). Use **privdebug.pl** to run the service as root to get a list of all of the privileges that are needed for the method.

In Example 5-26 on page 159, the WebSphere Application Server is installed and owned by the user wasuser1 in the Non-Global Zone named waszone8. You can use **privdebug.pl** to determine which privileges are really required to start and stop the WebSphere Application Server.

Example 5-26 Using the privdebug.pl script to examine the start method

```
# ./privdebug.pl -f -v -e  "zlogin -l wasuser1 waszone8 /lib/svc/method/svc-was61
start"
STAT TIMESTAMP          PPID   PID    UID    PRIV           CMD
USED 15790348592172000  5344   5345   0      proc_fork      zlogin
USED 15790348656904000  5347   5348   100    proc_exec      startServer.sh
USED 15790348661423100  5347   5348   100    proc_fork      startServer.sh
USED 15790348675359800  5347   5348   100    proc_fork      startServer.sh
USED 15790348677628000  5347   5348   100    proc_fork      startServer.sh
USED 15790348682510900  5347   5348   100    proc_fork      startServer.sh
USED 15790348691073400  5347   5348   100    proc_fork      startServer.sh
USED 15790348697784300  5347   5348   100    proc_fork      startServer.sh
USED 15790348650569000  5346   5347   100    proc_exec      svc-was61
USED 15790348655554700  5346   5347   100    proc_fork      startServer.sh
USED 15790348670542100  5347   5348   100    proc_fork      startServer.sh
USED 15790348692779800  5348   5355   100    proc_exec      startServer.sh
USED 15790348699352300  5348   5356   100    proc_exec      startServer.sh
USED 15790348596052000  5345   5346   0      proc_exec      zlogin
USED 15790348605130200  5345   5346   0      sys_audit      su
USED 15790348605139200  5345   5346   0      sys_audit      su
USED 15790348668087400  5347   5348   100    proc_fork      startServer.sh
USED 15790348703717100  5347   5348   100    proc_fork      startServer.sh
USED 15790348622866300  5345   5346   0      sys_audit      su
USED 15790348622876000  5345   5346   0      sys_audit      su
USED 15790348622955200  5345   5346   0      sys_audit      su
USED 15790348622962000  5345   5346   0      sys_audit      su
USED 15790348625927600  5345   5346   0      proc_taskid    su
USED 15790348628039600  5345   5346   0      proc_setid     su
USED 15790348628399600  5345   5346   0      proc_setid     su
USED 15790348628420800  5345   5346   0      proc_setid     su
USED 15790348628429500  5345   5346   0      proc_setid     su
USED 15790348629763000  5345   5346   100    proc_exec      su
USED 15790348642003500  5345   5346   100    proc_exec      bash
USED 15790348649028000  5345   5346   100    proc_fork      svc-was61

#...Note: the actual service method starts here.....

USED 15790348662933200  5348   5349   100    proc_exec      startServer.sh
USED 15790348671951300  5348   5351   100    proc_exec      startServer.sh
USED 15790348678863000  5348   5353   100    proc_exec      startServer.sh
USED 15790348683745700  5348   5354   100    proc_exec      startServer.sh
USED 15790348705377900  5348   5357   100    proc_exec      startServer.sh
USED 15790348710999200  5347   5348   100    proc_fork      startServer.sh
USED 15790348712984000  5348   5358   100    proc_exec      startServer.sh
USED 15790348718707900  5348   5358   100    proc_exec      java
ADMU0116I   Tool information is being logged in file       contract_event
/home1/wasuser1/IBM/WebSphere/AppServer/profiles/AppSrv01/logs/server1/startServer
.log                                        contract_event
ADMU0128I Starting tool with the AppSrv01 profile contract_event
ADMU3100I server1  Reading configuration for server contract_event
USED 15790359433038600  5348   5358   100    sys_net_config  java
USED 15790359526897100  5348   5358   100    proc_fork       java
ADMU3200I Server launched. Waiting for initialization status.
contract_event
USED 15790359746491500  5358   5359   100    proc_exec       java
```

```
ADMU3000I Server server1 open for e-business;
process id is 5359        contract_event
USED 15790404267579600  5348   5362   100    proc_exec      startServer.sh
USED 15790404265525600  5347   5348   100    proc_fork      startServer.sh
```

Notice in this example that the start method only requires proc_fork and proc_exec. The privilege sys_net_config was also used in this example, but an examination of the **privileges(5)** man page shows that it is not needed. In fact, it silently fails without impact later in the application execution.

Similarly, running the stop method produces the output in Example 5-27.

Example 5-27 Output of the stop method

```
# ./privdebug.pl -f -v -e  "zlogin -l wasuser1 waszone8 SMF/svc-was61 stop"
STAT TIMESTAMP          PPID   PID    UID    PRIV                 CMD
USED 16359596049040700  12720  12721  100    proc_exec            stopServer.sh
USED 16359596058267800  12720  12723  100    proc_exec            stopServer.sh
USED 16359596066248700  12720  12725  100    proc_exec            stopServer.sh
USED 16359596079437900  12719  12720  100    proc_fork            stopServer.sh
USED 16359596085926300  12719  12720  100    proc_fork            stopServer.sh
USED 16359596093911600  12720  12729  100    proc_exec            stopServer.sh
USED 16359595958009600  12716  12717  0      proc_fork            zlogin
USED 16359596042624000  12719  12720  100    proc_exec            stopServer.sh
USED 16359596047276100  12719  12720  100    proc_fork            stopServer.sh
USED 16359596054667200  12719  12720  100    proc_fork            stopServer.sh
USED 16359596056835700  12719  12720  100    proc_fork            stopServer.sh
USED 16359596062429700  12719  12720  100    proc_fork            stopServer.sh
USED 16359596071655900  12720  12726  100    proc_exec            stopServer.sh
USED 16359596087632400  12720  12728  100    proc_exec            stopServer.sh
USED 16359596099548400  12719  12720  100    proc_fork            stopServer.sh
USED 16359596036485700  12718  12719  100    proc_exec            svc-was61
USED 16359596041290700  12718  12719  100    proc_fork            stopServer.sh
USED 16359596081106400  12720  12727  100    proc_exec            stopServer.sh
USED 16359595962358700  12717  12718  0      proc_exec            zlogin
USED 16359595972367400  12717  12718  0      sys_audit            su
USED 16359595972376100  12717  12718  0      sys_audit            su
USED 16359596005195900  12717  12718  0      sys_audit            su
USED 16359596005204600  12717  12718  0      sys_audit            su
USED 16359596005273100  12717  12718  0      sys_audit            su
USED 16359596005280400  12717  12718  0      sys_audit            su
USED 16359596008126800  12717  12718  0      proc_taskid          su
USED 16359596010251100  12717  12718  0      proc_setid           su
USED 16359596010555700  12717  12718  0      proc_setid           su
USED 16359596010577800  12717  12718  0      proc_setid           su
USED 16359596010585800  12717  12718  0      proc_setid           su
USED 16359596011770000  12717  12718  100    proc_exec            su
USED 16359596028292100  12717  12718  100    proc_exec            bash
USED 16359596034882100  12717  12718  100    proc_fork            svc-was61
USED 16359596064812300  12719  12720  100    proc_fork            stopServer.sh
USED 16359596070167400  12719  12720  100    proc_fork            stopServer.sh
USED 16359596092285300  12719  12720  100    proc_fork            stopServer.sh
USED 16359596101273200  12720  12730  100    proc_exec            stopServer.sh
USED 16359596106255100  12720  12730  100    proc_exec            java
ADMU0116I                          Tool information is being logged in file
contract_event
```

```
/home1/wasuser1/IBM/WebSphere/AppServer/profiles/AppSrv01/logs/server1/stopServer.
log                                          contract_event
ADMU0128I                           Starting tool with the AppSrv01 profile
contract_event
ADMU3100I                      server1  Reading configuration for server
contract_event
ADMU3201I                             Server stop request issued. Waiting for stop
status.        contract_event
ADMU4000I                             Server server1 stop completed.
contract_event
```

Since our service manifest, as discussed in Example 5-20 on page 152, maps the refresh method to a service restart (stop followed by start), the refresh method requires the same privileges: proc_fork and proc_exec.

These method credentials can be added to our existing SMF manifest in the method_context for each of the three exec_methods. These entries would now look like the ones shown in Example 5-28.

Example 5-28 Modified SMF for start and stop methods with non-root user and restricted privileges

```
<exec_method name='start' type='method'
           exec='/lib/svc/method/svc-was61 start' timeout_seconds='60'>
      <method_context working_directory='/home1/wasuser1'>
              <method_credential user='wasuser1'
           group='wasgrp1' privileges='proc_fork,proc_exec'  />
      </method_context>
      </exec_method>

    <exec_method name='stop' type='method'
           exec='/lib/svc/method/svc-was61 start' timeout_seconds='60'>
      <method_context working_directory='/home1/wasuser1'>
              <method_credential user='wasuser1'
           group='wasgrp1' privileges='proc_fork,proc_exec'  />
      </method_context>

      </exec_method>

<exec_method name='refresh' type='method'
           exec='/lib/svc/method/svc-was61 retart' timeout_seconds='60'>
      <method_context working_directory='/home1/wasuser1'>
              <method_credential user='wasuser1'
           group='wasgrp1' privileges='proc_fork,proc_exec'  />
      </method_context>

      </exec_method>
```

You need to import the updated manifest and start the service as discussed in 5.5, “Service Management Facility (SMF)” on page 150. You can then examine the processes to show this least privilege configuration. The steps are shown in Example 5-29.

Example 5-29 Working with Service Manifest Facility

```
# svccfg import /var/svc/manifest/application/was61.xml
# svcadm enable was61
# svcs -p was61
STATE          STIME    FMRI
online         11:52:55 svc:/application/was61:was_server1
               11:52:10     5945 java

# ppriv 5945
5945:   /home1/wasuser1/IBM/WebSphere/AppServer/java/bin/java -XX:MaxPermSize=
flags = <none>
        E: basic,!file_link_any,!proc_info,!proc_session
        I: basic,!file_link_any,!proc_info,!proc_session
        P: basic,!file_link_any,!proc_info,!proc_session
        L:
basic,contract_event,contract_observer,file_chown,file_chown_self,file_dac_execute
,file_dac_read,file_dac_search,file_dac_write,file_owner,file_setid,ipc_dac_read,i
pc_dac_write,ipc_owner,net_bindmlp,net_icmpaccess,net_mac_aware,net_privaddr,proc_
audit,proc_chroot,proc_owner,proc_setid,proc_taskid,sys_acct,sys_admin,sys_audit,s
ys_mount,sys_nfs,sys_resource
```

Not only is WebSphere Application Server running as a non-root user, but with less privileges than a regular user. If the service becomes compromised, the attacker can only use fork and exec. And if the service is running in a Non-Global Zone, then the max-lwps resource control will limit the impact of the compromise.

This technique can be applied to more complicated service examples, including server side applets or scripts that require additional privileges that normally run as root.

For complete details on Process Rights Management, refer to *Limiting Service Privileges in the Solaris 10 Operating System* by Brunette, found at:

http://sun.com/blueprints/0505/819-2680.pdf

5.7 Solaris ZFS

ZFS is a new file system introduced in Solaris 10 6/06. It is designed to relieve the constraints and limitations of previous file systems, and provide new capabilities needed for modern data centers. ZFS discards design decisions based on assumptions that were valid 10 to 20 years ago when most contemporary file systems were written, but are no longer true. For example, it was previously thought that doing end-to-end checksums were too expensive to do in a file system. Now, data is too important to not protect, and the hardware based solutions miss too many failure scenarios. Innovative technology in ZFS makes it possible to provide these new, important functions without sacrificing performance or usability.

ZFS is designed to provide:

- Data integrity that far exceeds the integrity and reliability provided by other file systems, by using end-to-end checksums on both data and metadata contents. ZFS automatically detects and recovers from media errors, cable problems, phantom or misdirected writes, driver errors, and other sources of disk subsystem error. ZFS also ensures that on-disk contents are always self-consistent, eliminating the need for the time-consuming "fsck" (file system check) traditionally needed after a system outage. ZFS does this by using "copy on write" algorithms that always write new data to a different location on disk. Related changes either succeed or fail to be written to disk as a whole. This ensures that the on-disk copy of data is consistent with itself.
- Scaling for today's massive applications and data requirements, without the use of complex, brittle, and expensive volume managers. ZFS does this by providing pooled storage with 128 bits of data addressability for limitless logical scale.
- Advanced capabilities like point-in-time snapshots, clones, on-disk encryption, and replication. For example, ZFS permits instantaneous data snapshots, which can be used to preserve the entire contents of a file system, whether they represent a particular software build's contents, or the contents of a stock market data repository at the moment the market closes. Snapshots are space-efficient, as only new, altered contents need to be written to different disk locations.
- Simpler administration, making it possible to create data resources, mirrors, and stripes without learning obscure command options and flags. In particular, ZFS removes administrator burden by providing self-tuning, removing onerous and error-prone activities such as setting stripe sizes or calculating space needed for inodes. ZFS automatically tunes itself to best match the data characteristics for which it is being used. It is easy for an administrator to set quotas or space reservations, to disable or enable compression, and other common tasks.
- High performance. Current file systems are slow in numerous common situations that ZFS addresses by eliminating linear-time file system creation, fixed block sizes, excessive locking, and naive prefetch.

These features make ZFS one of the most advanced file systems, bringing new levels of reliability and protection for critical data assets. A free feature of Solaris, it is so popular that it has been adapted from OpenSolaris to be ported to other operating systems, such as Linux and MacOS 10.5.

ZFS provides a pooled storage model that no longer requires the user to deal with the intricacies of storage management, such as volumes, partitions (or slices), format, mounting, and underutilization. You can easily build RAID volumes and file systems without needing expensive volume management software. In this section, we provide an introduction to ZFS and show how it can be applied to your WebSphere Application Server deployment.

On the Solaris system, you can examine the available disks using the **format** command:

```
global# format
Searching for disks...done

AVAILABLE DISK SELECTIONS:
       0. c0t0d0 <DEFAULT cyl 8894 alt 2 hd 255 sec 63>
          /pci@0,0/pci1022,7450@2/pci1000,3060@3/sd@0,0
       1. c0t1d0 <SEAGATE-ST973401LSUN72G-0556-68.37GB>
          /pci@0,0/pci1022,7450@2/pci1000,3060@3/sd@1,0
       2. c0t2d0 <FUJITSU-MAY2073RCSUN72G-0501-68.37GB>
          /pci@0,0/pci1022,7450@2/pci1000,3060@3/sd@2,0
       3. c0t3d0 <FUJITSU-MAY2073RCSUN72G-0501-68.37GB>
          /pci@0,0/pci1022,7450@2/pci1000,3060@3/sd@3,0
Specify disk (enter its number): ^C
```

To create a file system on Solaris, you would typically use the **format** command to select a disk, create partitions/slices, and do **mkfs**. ZFS eliminates all those steps and simplifies the process with the **zpool** and **zfs** commands. The following example demonstrates that a ZFS file system, named pool1, is created using the device c0t1d0 with 68 GB:

```
global# zpool list
no pools available
global# zpool create pool1 c0t1d0
global# zpool list
NAME                    SIZE    USED   AVAIL    CAP  HEALTH     ALTROOT
pool1                    68G     88K   68.0G     0%  ONLINE     -
global# zpool status
  pool: pool1
 state: ONLINE
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        pool1       ONLINE       0     0     0
          c0t1d0    ONLINE       0     0     0

errors: No known data errors
```

If you need to make this volume bigger, you can simply concatenate one or more additional devices to pool1, as shown in the following example. You will find that the pool size has grown from 68 GB to 136 GB after adding another device (c0t2d0) with 68 GB:

```
global# zpool add pool1 c0t2d0
global# zpool list
NAME                    SIZE    USED   AVAIL    CAP  HEALTH     ALTROOT
pool1                   136G     91K    136G     0%  ONLINE     -
global# zpool status
  pool: pool1
 state: ONLINE
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        pool1       ONLINE       0     0     0
          c0t1d0    ONLINE       0     0     0
          c0t2d0    ONLINE       0     0     0
```

```
errors: No known data errors
```

If you want to delete this pool and create another pool, called mpool1, a two-way mirrored volume with c0t1d0 and c0t2d0, do the following (mpool1 has 2x68 GB on disks in the zfs pool and is mirrored; thus, the total pool size is 68 GB.):

```
global# zpool destroy pool1
global# zpool list
no pools available
global# zpool create mpool1 mirror c0t1d0 c0t2d0
global# zpool list
NAME                    SIZE    USED   AVAIL    CAP  HEALTH     ALTROOT
mpool1                   68G     89K   68.0G     0%  ONLINE     -
global# zpool status
  pool: mpool1
 state: ONLINE
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        mpool1      ONLINE       0     0     0
          mirror    ONLINE       0     0     0
            c0t1d0  ONLINE       0     0     0
            c0t2d0  ONLINE       0     0     0

errors: No known data errors
```

Now, you can turn mpool1 from a two-way mirrored volume to a three-way mirrored volume, as shown in the following example:

```
global# zpool attach mpool1 c0t1d0 c0t3d0
global# zpool list
NAME                    SIZE    USED   AVAIL    CAP  HEALTH     ALTROOT
mpool1                   68G     90K   68.0G     0%  ONLINE     -
global# zpool status
  pool: mpool1
 state: ONLINE
 scrub: resilver completed with 0 errors on Mon Oct 29 16:39:13 2007
config:

        NAME        STATE     READ WRITE CKSUM
        mpool1      ONLINE       0     0     0
          mirror    ONLINE       0     0     0
            c0t1d0  ONLINE       0     0     0
            c0t2d0  ONLINE       0     0     0
            c0t3d0  ONLINE       0     0     0

errors: No known data errors
```

You can also turn mpool1 back into a two-way mirrored volume between c0t1d0 and c0t3d0 by taking out c0t2d0 from the pool with the **detach** command:

```
global# zpool detach mpool1 c0t2d0
global# zpool status
  pool: mpool1
 state: ONLINE
 scrub: resilver completed with 0 errors on Mon Oct 29 16:43:13 2007
config:

        NAME        STATE     READ WRITE CKSUM
        mpool1      ONLINE       0     0     0
          mirror    ONLINE       0     0     0
            c0t1d0  ONLINE       0     0     0
            c0t3d0  ONLINE       0     0     0

errors: No known data errors
```

Furthermore, you can create zfs file systems within the pool that we just created:

```
global# zfs create mpool1/IBMSW1
global# zfs create mpool1/IBMSW2

You can check the filesystem with the standard Solaris command like df.

global# df
/                  (/dev/dsk/c0t0d0s0 ):129268622 blocks  8041716 files
/devices           (/devices          ):       0 blocks        0 files
/system/contract   (ctfs              ):       0 blocks 2147483614 files
/proc              (proc              ):       0 blocks    16301 files
/etc/mnttab        (mnttab            ):       0 blocks        0 files
/etc/svc/volatile  (swap              ):17003904 blocks  1166846 files
/system/object     (objfs             ):       0 blocks 2147483494 files
/lib/libc.so.1     (/usr/lib/libc/libc_hwcap2.so.1):129268622 blocks  8041716
files
/dev/fd            (fd                ):       0 blocks        0 files
/tmp               (swap              ):17003904 blocks  1166846 files
/var/run           (swap              ):17003904 blocks  1166846 files
/mpool1            (mpool1            ):140377815 blocks 140377815 files
/mpool1/IBMSW1     (mpool1/IBMSW1     ):140377815 blocks 140377815 files
/mpool1/IBMSW2     (mpool1/IBMSW2     ):140377815 blocks 140377815 files
```

You may also change the mount point to something that we like, such as /opt2:

```
global# zfs set mountpoint=/opt2 mpool1
global# df
/                  (/dev/dsk/c0t0d0s0 ):129268624 blocks  8041717 files
/devices           (/devices          ):       0 blocks        0 files
/system/contract   (ctfs              ):       0 blocks 2147483614 files
/proc              (proc              ):       0 blocks    16301 files
/etc/mnttab        (mnttab            ):       0 blocks        0 files
/etc/svc/volatile  (swap              ):17003648 blocks  1166846 files
/system/object     (objfs             ):       0 blocks 2147483494 files
/lib/libc.so.1     (/usr/lib/libc/libc_hwcap2.so.1):129268624 blocks  8041717
files
/dev/fd            (fd                ):       0 blocks        0 files
/tmp               (swap              ):17003648 blocks  1166846 files
```

```
/var/run           (swap              ):17003648 blocks  1166846 files
/opt2              (mpool1            ):140377801 blocks 140377801 files
/opt2/IBMSW1       (mpool1/IBMSW1     ):140377801 blocks 140377801 files
/opt2/IBMSW2       (mpool1/IBMSW2     ):140377801 blocks 140377801 files
global# zfs list
NAME            USED  AVAIL  REFER  MOUNTPOINT
mpool1          156K  66.9G  27.5K  /opt2
mpool1/IBMSW1  24.5K  66.9G  24.5K  /opt2/IBMSW1
mpool1/IBMSW2  24.5K  66.9G  24.5K  /opt2/IBMSW2
```

Now, this file system is ready for IBM WebSphere deployment. You can monitor the performance of ZFS as follows:

```
global# zpool iostat -v 5
               capacity     operations    bandwidth
pool         used  avail   read  write   read  write
----------  -----  -----  -----  -----  -----  -----
mpool1       158K  68.0G      0      1    104  2.47K
  mirror     158K  68.0G      0      1    104  2.47K
    c0t1d0      -      -      0      0    417  4.31K
    c0t3d0      -      -      0      0    106  3.83K
----------  -----  -----  -----  -----  -----  -----
```

The advantages of using ZFS for WebSphere Application Server deployment are:

- Provisioning of WebSphere Application Server and Solaris Containers with higher reliability.
- Delegate file system management to a Container.
- Easier to migrate Solaris Containers from one host to another.
- Large log files can be automatically compressed within the ZFS environment.
- Simplify volume and file system management.

For more information, refer to http://sun.com/solaris/zfs_learning_center.jsp.

Tip: For collaboration with others, you can join the OpenSolaris community for ZFS at http://www.opensolaris.org/os/community/zfs.

6

Management and maintenance of WebSphere Application Server on the Solaris Operating System

This chapter discusses how to manage and maintain specific WebSphere Application Server versions, fix packs, and Service Releases, as well as Solaris Patches. We discuss how you can apply fix packs to a shared installation environment on Solaris Zones. We also describe how WebSphere Application Server deployment in Solaris containers can be moved and migrated from one system to another.

6.1 WebSphere Application Server maintenance

It is a best practice to follow IBM's recommendations for WebSphere Application Server maintenance. There are several reasons why:

- Keeps your environment up to date.
- Keeps your system safe from a security point of view.
- Eliminates environment instability due to WebSphere Application Server software bugs.
- Takes full advantage of your WebSphere Application Server environment.

This topic discusses the update strategy of your WebSphere Application Server environment, and how to install and uninstall a maintenance package to your WebSphere Application Server installation.

6.1.1 Update strategy for WebSphere Application Server V6.1

Note: The following information in this section can be accessed o line at:

http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg27009276

WebSphere Application Server Support delivers this strategy to provide a clear upgrade path that reduces the risk of installing collections of Application Server fixes.

Requests for published collections of fixes are delivered in a timely manner and tested together without new features or behavioral changes.

WebSphere Application Server V6.1 will provide fix packs containing cumulative fixes that will be updated regularly. This provides a consistent maintenance approach that you can follow as you manage your products.

Installing preventative maintenance as soon as it becomes available will save you time. As long as you test appropriately, actively applying preventative maintenance can avoid problems that could result in a service call.

Delivering updates

Fix Pack deliveries will be timely. Deliveries are approximately every 12 weeks. The date of next planned Fix Pack will be published at http://www.ibm.com/software/webservers/appserv/was/support, at least 30 days prior to availability.

Each Fix Pack delivery *can* consist of multiple fix packs for the following components:

- Application Server
- Application Client
- Web server plug-ins
- IBM HTTP Server
- Java SDK

Note:

- The preceding component fix packs are all tested together at the same level.
- In some cases, maintenance might not be delivered for every component for each new Fix Pack level (for example, there was no maintenance delivered for IBM HTTP Server for Fix Pack 1 (6.1.0.1)).
- IBM supports users who might not install all fix packs delivered at each Fix Pack level (for example, installing Fix Pack 5 (6.1.0.5) for the Application Server but not installing Fix Pack 5 for IBM HTTP Server).

There is a single list of all defects for each release that is updated for each new Fix Pack. Install a fix on top of any previous Fix Pack installed (for example, 6.1.0.5 could be applied to 6.1.0.1 or 6.1.0.2).

Note: The version numbers (6.1, 6.1.0.1, 6.1.0.2, and so on) used throughout this document are to illustrate a typical maintenance path used to provide solutions to our customers. It does not reflect actual, nor intended, deliverables.

Table 6-1 shows the types of solutions that are provided for WebSphere Application Server.

Table 6-1 Types of solution provided for WebSphere Application Server

Solutions	Characteristics of each solution
Release	☐ A new WebSphere Application Server that includes a major new function, such as V6.1. ☐ This is a separate installation that can coexist with other Application Server releases. ☐ Full testing of all applications with a new release is recommended. ☐ Requires a migration to upgrade. Distribution is done by way of Passport Advantage®.
Feature pack	☐ WebSphere Application Server V6.1 introduces Feature Packs, which are free, downloadable product extensions (on top of V6.1) that offer targeted, incremental new features. ☐ Regression testing of critical functions with new feature packs is strongly recommended.

Solutions	Characteristics of each solution
Fix Pack	☐ This is the standard delivery for updates; it has been regression tested. ☐ A Fix Pack is a cumulative package of fixes, such as Fix Pack 2 (6.1.0.2). ☐ Each Fix Pack delivery can consist of multiple fix packs for the following components: – Application Server – Application Client – Web server plug-ins – IBM HTTP Server – Java SDK ☐ Fix packs also install on top of a previous fix pack, such as applying V6.1.0.1 to V6.1.0.2. ☐ Fix packs are cumulative, so V6.1.0.2 includes all fixes in V6.1.0.1. ☐ Fix packs uninstall all interim fixes applied to the release since the last Fix Pack was installed. Therefore, it is necessary to check the list of delivered fixes to determine if an interim fix needs to be reinstalled. ☐ Brief testing of critical functions with the new Fix Pack is recommended.
Fix	☐ A single published emergency fix, such as PK31235. ☐ A fix is an interim fix or test fix that resolves one or more product defects. ☐ A fix can be applied to a release or Fix Pack where applicable. ☐ Interim Fix = IFnnnn (for example: 6.1 IF0001) ☐ Test Fix = TFnnnn (for example: 6.1 TF0002) ☐ Interim fixes are created when a stand-alone fix is required between fix packs. They are validated by at least one customer prior to being published. ☐ We recommend that you test functions affected by the WebSphere Application Server component fixed. ☐ Reference the Recommended fixes page for currently available fixes. When urgent, the fix will also be flashed on the Support page.

Maintenance planning

Choose the appropriate update solution and verify the inclusion of the needed individual fixes in the Fix Pack:

- Review the Recommended fixes and the Fix list for Version 6.1 to help you plan your updates (http://www.ibm.com/software/webservers/appserv/was/support).

- In order to ensure the quality of fix packs, individual fixes that become available late in the development cycle may *not* be included in a Fix Pack.
- The **historyInfo** utility lists the history of installed and uninstalled fixes. For more details, refer to 6.3.1, “Installing Update Installer to the Global Zone” on page 176.

Prior to installing any WebSphere Application Server solution in production, we highly recommend that appropriate testing should be done on a test system.

Recommended update path

- Update to the latest Fix Pack for cumulative fixes in order to proactively avoid problems already resolved within WebSphere Application Server.
- Update to the latest Fix Pack only when there is time in the development schedule for a full regression test of your application.
- Plan appropriate testing prior to installing any WebSphere Application Server solution in production.

The current recommended fixes for WebSphere Application Server V6.1 are available at:

http://www.ibm.com/support/dociew.wss?rs=180&uid=swg27004980#ver61

6.1.2 WebSphere Feature Packs

As part of the WebSphere Software Early Programs, WebSphere Application Server Feature Packs are available as optionally installable product extensions that offer targeted, incremental new features, such as:

- IBM WebSphere Application Server V6.1 Feature Pack for SOA - Beta Release
- IBM WebSphere Application Server Feature Pack for Web 2.0 Beta Program
- IBM WebSphere Application Server Version 6.1 Feature Pack for EJB 3

With feature packs, you can selectively take advantage of new standards and features while maintaining a more stable internal release cycle. Feature packs are offered generally available or available in either open alpha, beta, or technology preview. WebSphere Feature Packs are available at:

https://www14.software.ibm.com/iwm/web/cc/earlyprograms/websphere.shtml

6.2 Solaris maintenance

For Solaris system maintenance, you or your Solaris system administrator should follow Sun’s Solaris Patch Management *Recommended Strategy*, found at:

http://www.sun.com/blueprints/0205/819-1002.pdf

You should also check the SunSolve Web site on regular basis and the IBM recommended fixes Web site for new required patches. Solaris patches are available for download at:

http://sunsolve.sun.com

You can also subscribe to Sun Alert for weekly summary reports at:

http://sunsolve.sun.com/show.do?target=salert-notice

You can determine your system’s current patch level with the following command:

```
bash# showrev -p
```

There are different level of patches that are available for the Solaris patching mechanism that you need to be concerned with:

- General patches
- Kernel Update patches

Some Solaris patches are targeted specifically for the Java platform known as JavaSE Cluster Patches for Solaris. These patches are concerned with your WebSphere's Java Virtual Machine and related software libraries. They are available for download at:

https://sunsolve.sun.com/show.do?target=patches/JavaSE

6.2.1 Maintenance of Solaris 10 patches and zones

A software package may have a global scope, common to the entire Solaris environment (the Global Zone and all the Non-Global Zones), or may potentially be installed in certain zones or be at different patch levels in different zones. Patching of Solaris systems with zones is documented in the manual *System Administration Guide: Solaris Containers -Resource Management and Solaris Zones*, found at http://docs.sun.com/app/docs/doc/817-1592. Its rules are briefly summarized here.

Solaris packages may apply to the entire system, or to individual zones. The patch and package tools are aware of the scope and visibility of each package and it is possible for the Global Zone administrator to manage the software in all zones, subject to the following package and patch attributes:

- The SUNW_PKG_ALLZONES package parameter is a binary flag that defines the scope of a package, specifying the type of zone in which the package can be installed. If set to true, then the package can be installed and patched only from the Global Zone. If false, then it can be patched individually in a Non-Global Zone
- The SUNW_PKG_HOLLOW parameter defines the visibility of a patch, specifying whether it must be installed and identical in all zones, and typically is for kernel software shared by all zones. If true, then the patch must be installed from the Global Zone, and is propagated implicitly to the Non-Global Zones so their patch inventories can maintain the proper patch dependencies. If SUNW_PKG_HOLLOW is true, then SUNW_PKG_ALLZONES is also true.
- Finally, SUNW_PKG_THISZONE is a binary flag that, if true, indicates that the patch applies only to the currently running zone (where the **patchadd** command is running). This is typically used for application software installed in a zone.

Solaris uses these attributes to drive the behavior of the **patchadd** utility: If **patchadd** is issued from the Global Zone, the patch is added to the Global Zone and the Global Zone's patch database is updated to record the software inventory. Then the patch is added to each Non-Global Zone, and each zone's patch database is updated. This makes is possible to patch all the zones on the system with a single administrator command. For the sake of efficiency, it is a best practice to boot all the zones before applying a patch. If a zone is not booted, then it is booted once into single user mode to check dependencies (and then halted), and again to install the patch packages (and then halted). If the zone is already running, then this extra activity is skipped.

If **patchadd** is issued within a Non-Global Zone and SUNW_PKG_ALLZONES=false for all packages affected by the patch, then the patch is added to that zone, and its individual patch database is updated. This makes it possible to separately patch zone-specific software under the control of the zone's administrator, and maintain different software levels. In general, this method is especially suitable for application software installed into a zone.

6.3 Maintenance of WebSphere Application Server V6.1 in zones

We introduce the deployment concepts and strategies for installation and configuration of WebSphere Application Server V6.1 in Solaris 10 Zones in 5.1.3, “Solaris Containers” on page 118. In this section, we describe how to manage the maintenance of such an environment. First, we describe how to install and uninstall an Update Installer. Then, we describe how to install and uninstall a Fix Pack in this environment.

We look at the special case of shared binary installation among multiple zones, as this is not documented in the IBM InfoCenter for WebSphere Application Server V6.1. The shared binaries reside in the Global Zone while the profile directories reside in the local zones. We demonstrate how to apply a Fix Pack to the WebSphere Application Server binaries in the Global Zone and how this update is automatically propagated to respective local zones. This deployment strategy improves the manageability of your WebSphere Application Server environment by needing to apply fixes once in the centrally located installation base in the Global Zone.

Before applying a new Fix Pack, we examine and record the current installed version of WebSphere Application Server with the **versionInfo.sh** command in both the Global Zone and local zones. We see the output of this command in Example 6-1.

Example 6-1 Output of the versionInfo.sh command before the update

```
WVER0010I: Copyright (c) IBM Corporation 2002, 2005; All rights reserved.
WVER0012I: VersionInfo reporter version 1.15.1.13, dated 3/29/06

--------------------------------------------------------------------------------
IBM WebSphere Application Server Product Installation Status Report
--------------------------------------------------------------------------------

Report at date and time November 16, 2007 9:06:31 AM PST

Installation
--------------------------------------------------------------------------------
Product Directory        /opt/IBM/WebSphere/AppServer
Version Directory        /opt/IBM/WebSphere/AppServer/properties/version
DTD Directory            /opt/IBM/WebSphere/AppServer/properties/version/dtd
Log Directory            /opt/IBM/WebSphere/AppServer/logs
Backup Directory
/opt/IBM/WebSphere/AppServer/properties/version/nif/backup
TMP Directory            /var/tmp

Product List
--------------------------------------------------------------------------------
ND                       installed

Installed Product
--------------------------------------------------------------------------------
Name                     IBM WebSphere Application Server - ND
Version                  6.1.0.0
ID                       ND
Build Level              b0620.14
Build Date               5/16/06
```

```
--------------------------------------------------------------------------------
End Installation Status Report
--------------------------------------------------------------------------------
```

6.3.1 Installing Update Installer to the Global Zone

To update the local zones WebSphere Application Server environment, we must first install the Update Installer and desired Fix Pack in the Global Zone. This procedure can be used for any WebSphere Application Server installation in Solaris. The procedure is as follows:

1. Download the newest Update Installer and Fix Pack from the IBM support Web site:

   http://www.ibm.com/support/docview.wss?rs=180&uid=swg24012718

2. Unpack and install the Update Installer package using the commands shown in Example 6-2.

Example 6-2 Unpack Update Installer package and installation

```
bash#unzip <Update_Installer.package.name.zip>
----Output omitted----
bash#cd UpdateInstaller
bash#./install
```

3. You will see the Update Installer installation welcome window (Figure 6-1).

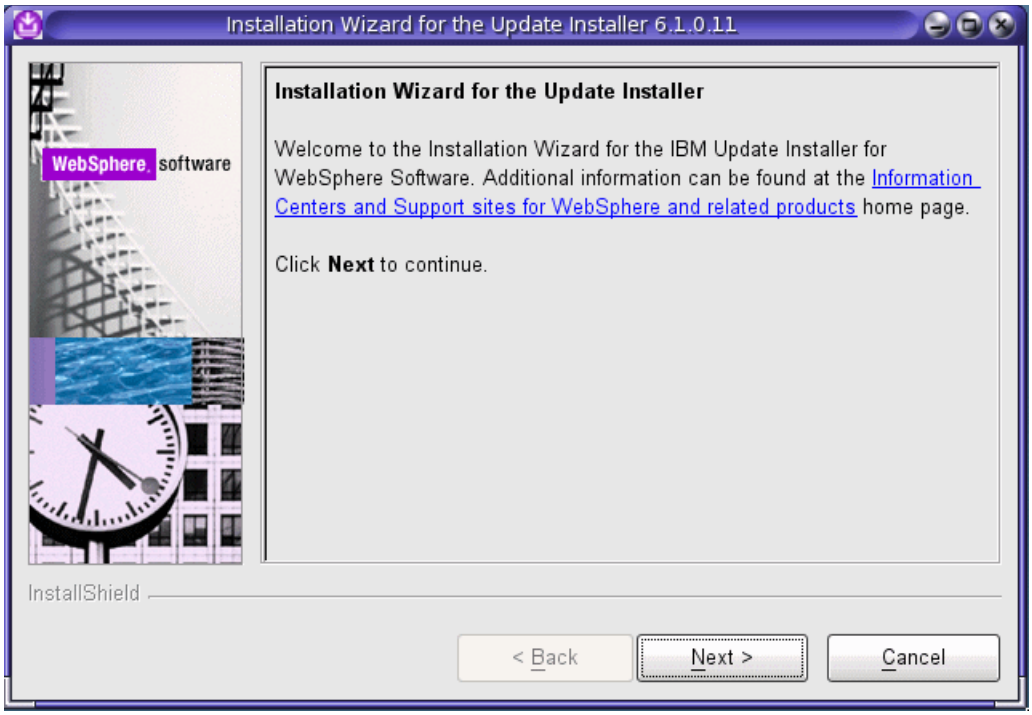


Figure 6-1 Update Installer installation welcome window

4. Click **Next**.

5. Read the licence agreement carefully, click **I accept the term in the licence agreement** radio button, and click **Next** (Figure 6-2 on page 177).

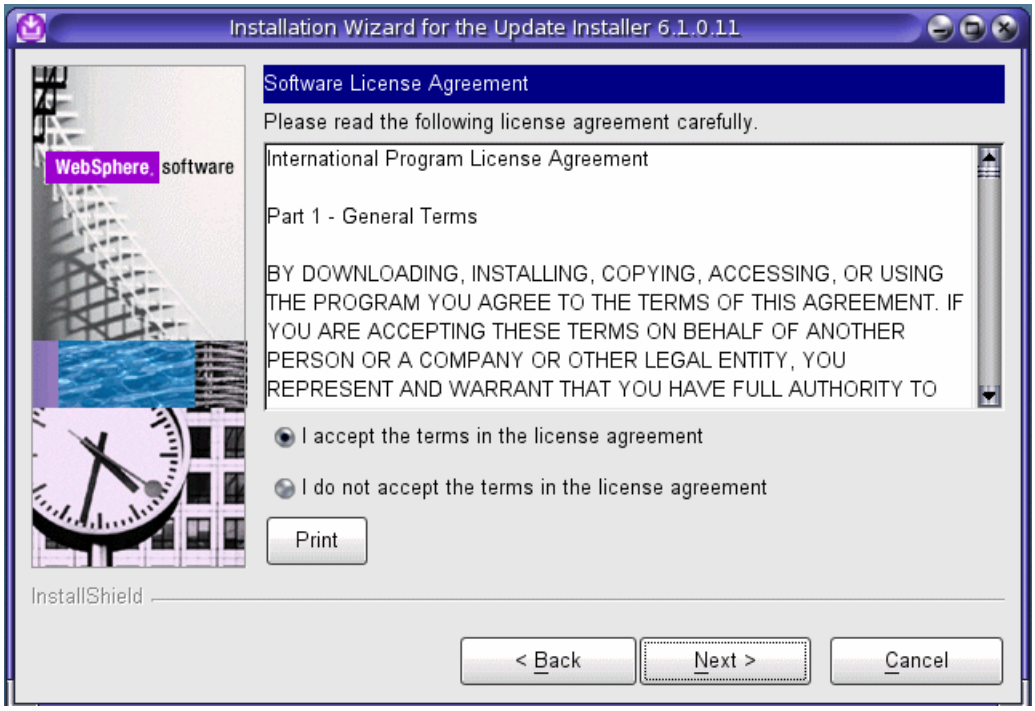


Figure 6-2 License agreement window

Installer checks that the system meets the installation requirements (Figure 6-3).

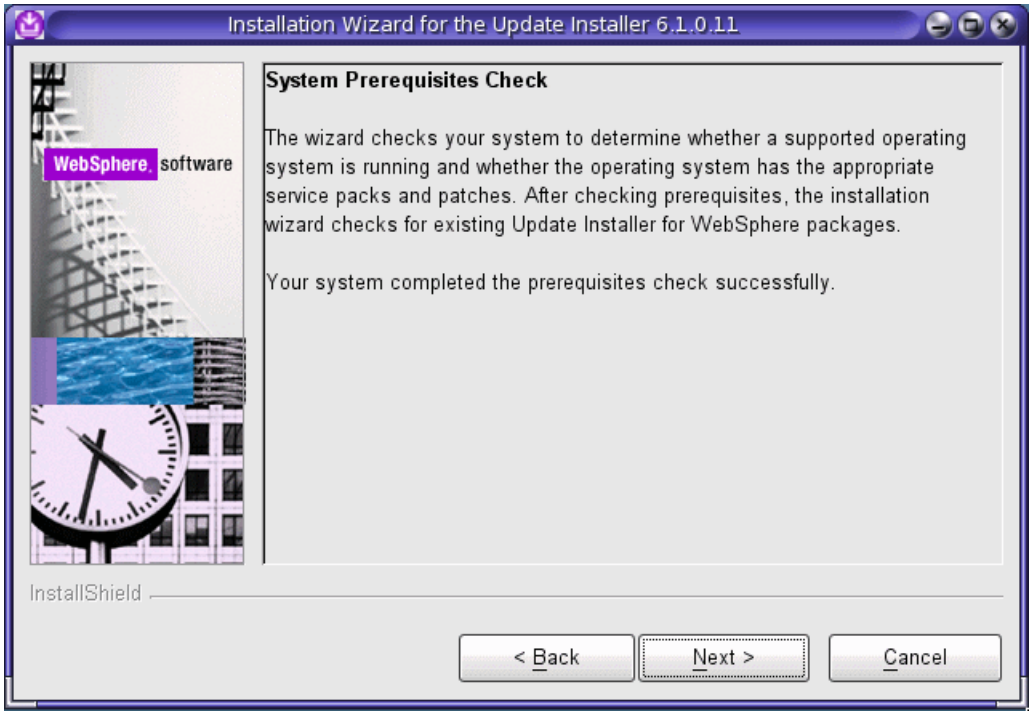


Figure 6-3 System prerequisite check window

6. Click **Next**.

   In this window (Figure 6-4), you can choose the installation directory for Update Installer. We recommend that you use the default installation directory.

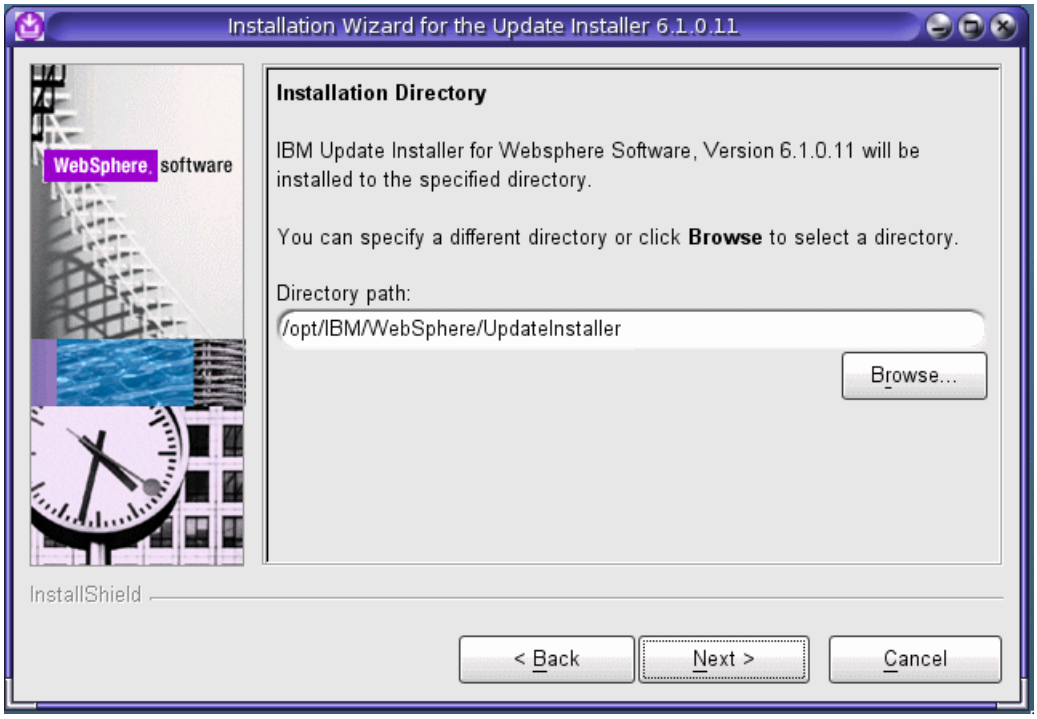


Figure 6-4 Installation directory window

Attention: Leaving the Directory path field empty prevents you from continuing. Remember that you cannot specify a directory that already contains Update Installer as a Directory path. If you have a previous version of Update Installer, you have to uninstall it prior to reinstallation.

7. Fill in the Directory path field or leave it as the default and click **Next**.
8. Review the installation information (Figure 6-5 on page 179). If it is correct, click **Next**. If it needs corrections, click **Back**, make the corrections, and proceed.

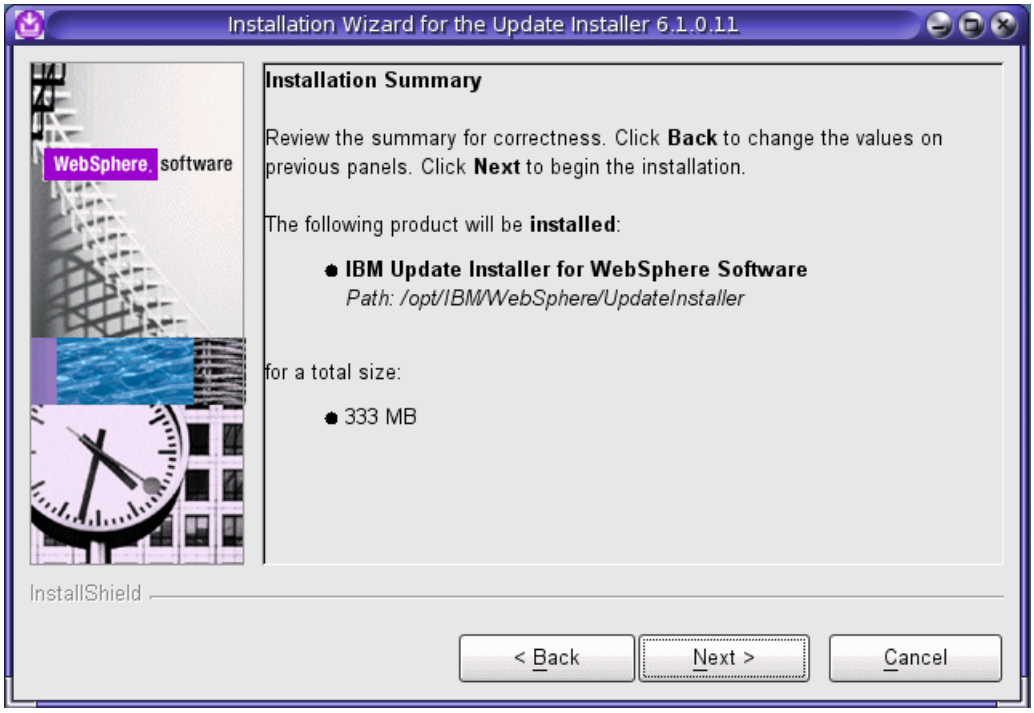


Figure 6-5 Installation summary window

The installer program creates an uninstaller first and then it installs the Update Installer package. After the successful completion of the installation, you should see the Installation Complete window (Figure 6-6).

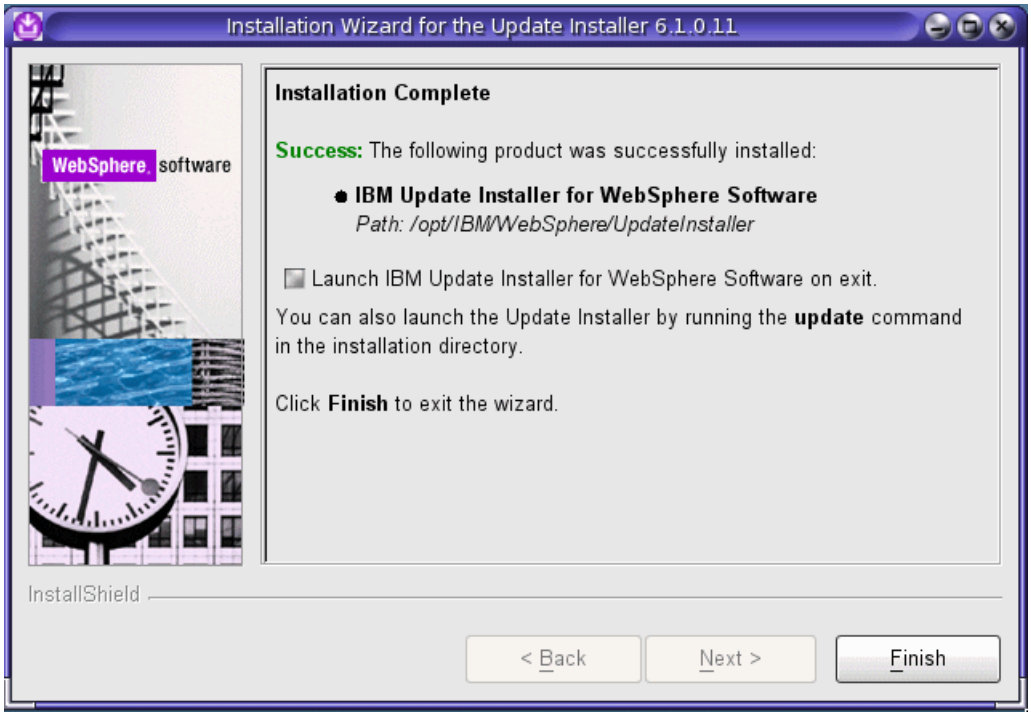


Figure 6-6 Update Installer installation successful window

You can launch the Update Installer program right after the installation of Update Installer is completed by selecting the **Launch IBM update installer for WebSphere Software on exit** check box or from the command line, as shown in Example 6-3.

Important: You must stop all WebSphere Application Server related Java processes before using the Update Installer because these active processes can interfere with product updates that it performs.

Example 6-3 Launching Update Installer from the command line

```
bash#cd /opt/IBM/WebSphere/UpdateInstaller
bash#./update.sh
```

9. To exit the install wizard, click **Finish**.

This procedure installs the Update Installer for WebSphere Software on your system. To install maintenance packs into your WebSphere Application Server environment, refer to 6.3.3, “Installing a Fix Pack to the Global Zone” on page 182 for more details of the installation of maintenance packages to your WebSphere Application Server environment.

6.3.2 Uninstalling Update Installer from the Global Zone

We describe the procedure to uninstall your Update Installer as the root user in this section:

1. Log in to the Global Zone as root.
2. Go to the Update Installers uninstall directory. Issue the **uninstall** command, as shown in Example 6-4.

Example 6-4 Start uninstaller of Update Installer program

```
bash#cd /opt/IBM/WebSphere/UpdateInstaller/uninstall
bash#./uninstall
```

3. The **uninstall** command starts the uninstallation wizard of Update Installer (Figure 6-7 on page 181). To continue, click **Next**.

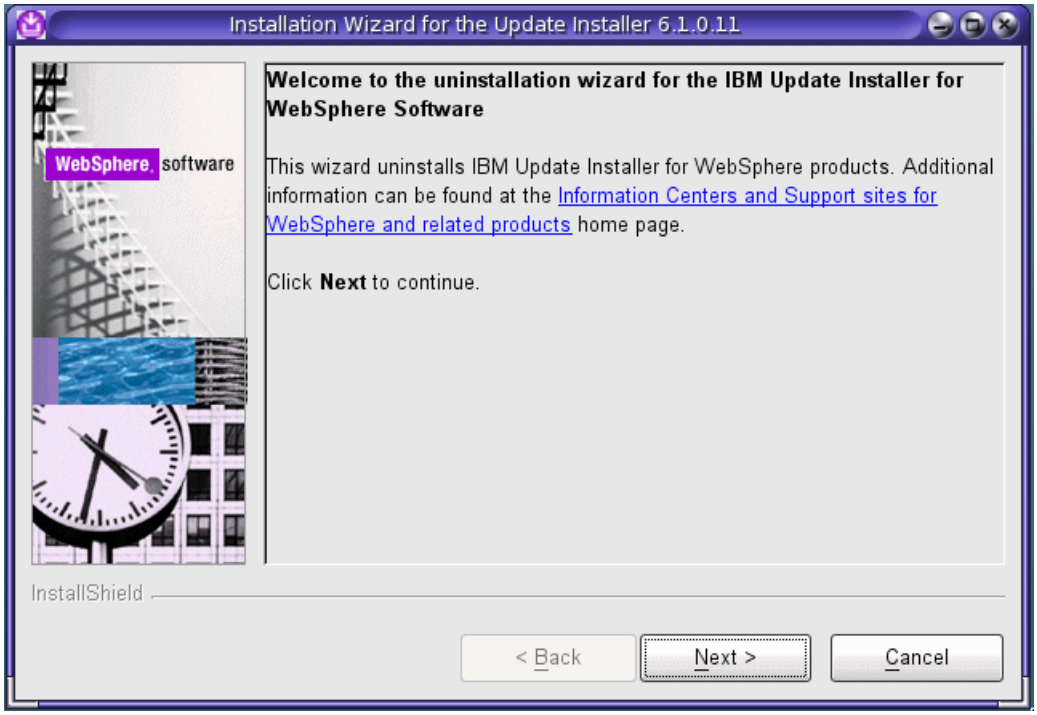


Figure 6-7 Update Installers uninstall welcome window

4. The uninstallation wizard shows the uninstallation summary (Figure 6-8). If you want to cancel uninstallation, click **Cancel**. To begin the uninstallation, click **Next**.

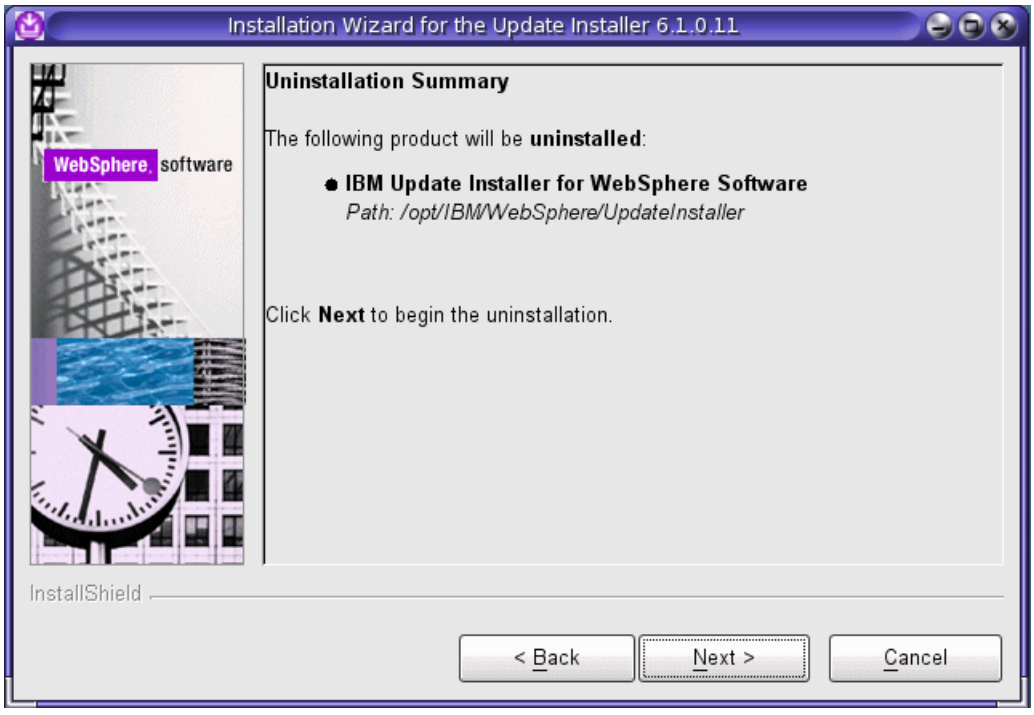


Figure 6-8 Update Installer uninstallation summary window

5. After uninstallation completes, the uninstallation wizard shows the Uninstallation Complete window (Figure 6-9). Click **Finish** to exit the uninstallation wizard.

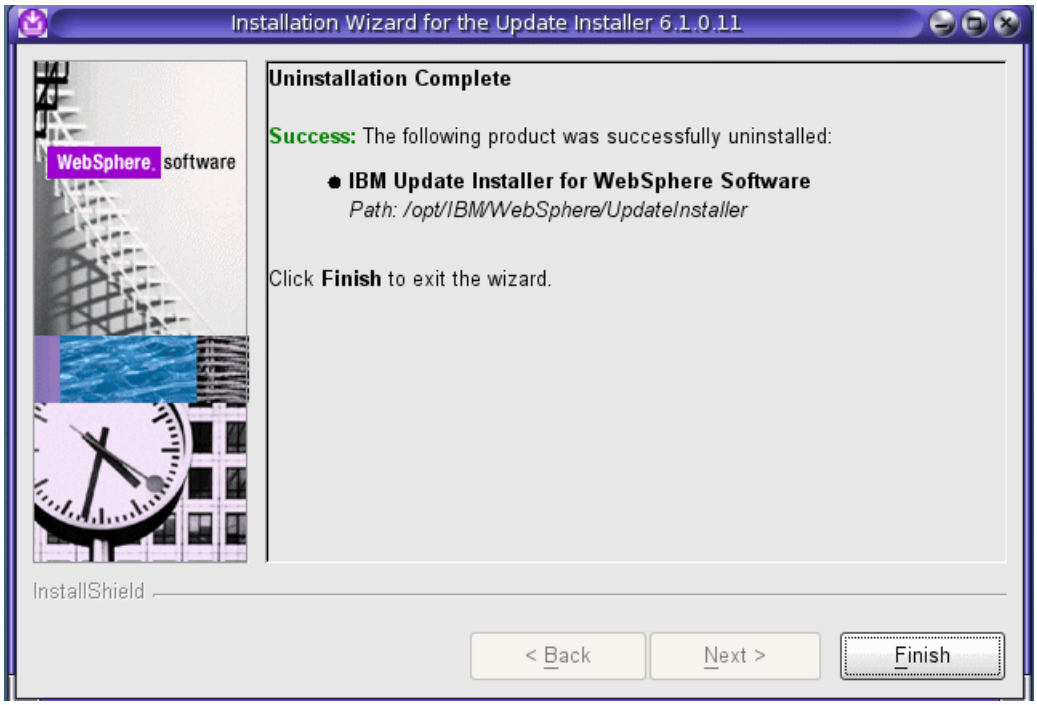


Figure 6-9 Uninstallation of Update Installer completed window

This completes the uninstallation of the WebSphere Application Server software from your system.

6.3.3 Installing a Fix Pack to the Global Zone

In this subsection, we install a fix pack to WebSphere Application Server. Assume you need to upgrade v6.1 with Fix Pack 11 taking the version from version 6.1.0.0 to version 6.1.0.11. We discuss how you can apply the Fix Pack to the WebSphere Application Server v6.1 and the Fix Pack also gets propagated automatically to each local zone that inherits the shared WebSphere Application Server installation as shown in Figure 6-10 on page 183.

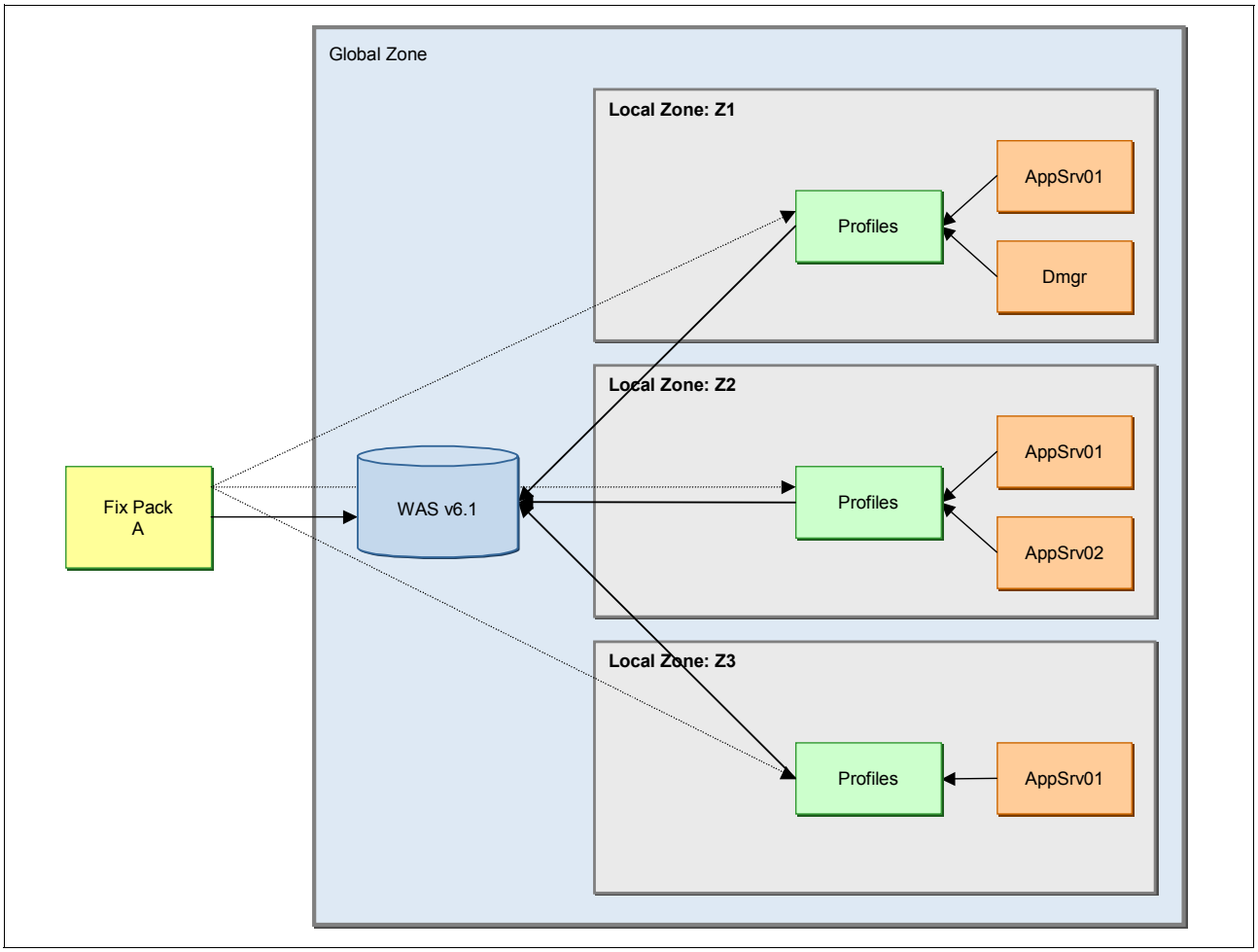


Figure 6-10 Installing a Fix Pack to the shared WebSphere Application Server binaries propagates to local zones

The Fix Pack installation process is the same as any other platform, but the significant part is the automatic propagation of the Fix Pack to each inherited local zone through the WebSphere Application Server base binary installation directory as a loop-back file system (LOFS):

1. Download the WebSphere Application Server Fix Pack from the IBM WebSphere Application Server Support Web site.
2. Put the Fix Pack into the WebSphere Application Server Update Installers maintenance directory. In our case, it is in /opt/IBM/WebSphere/UpdateInstaller/maintenance.
3. Go to Update Installer directory and issue the command shown in Example 6-5.

Example 6-5 Start update command

```
bash#./update.sh
```

4. After issuing the command, you see the IBM Update Installer for WebSphere software welcome window (Figure 6-11).

Attention: Make sure that every WebSphere and related Java processes are stopped before continuing Fix Pack installation. Otherwise, failure may occur.

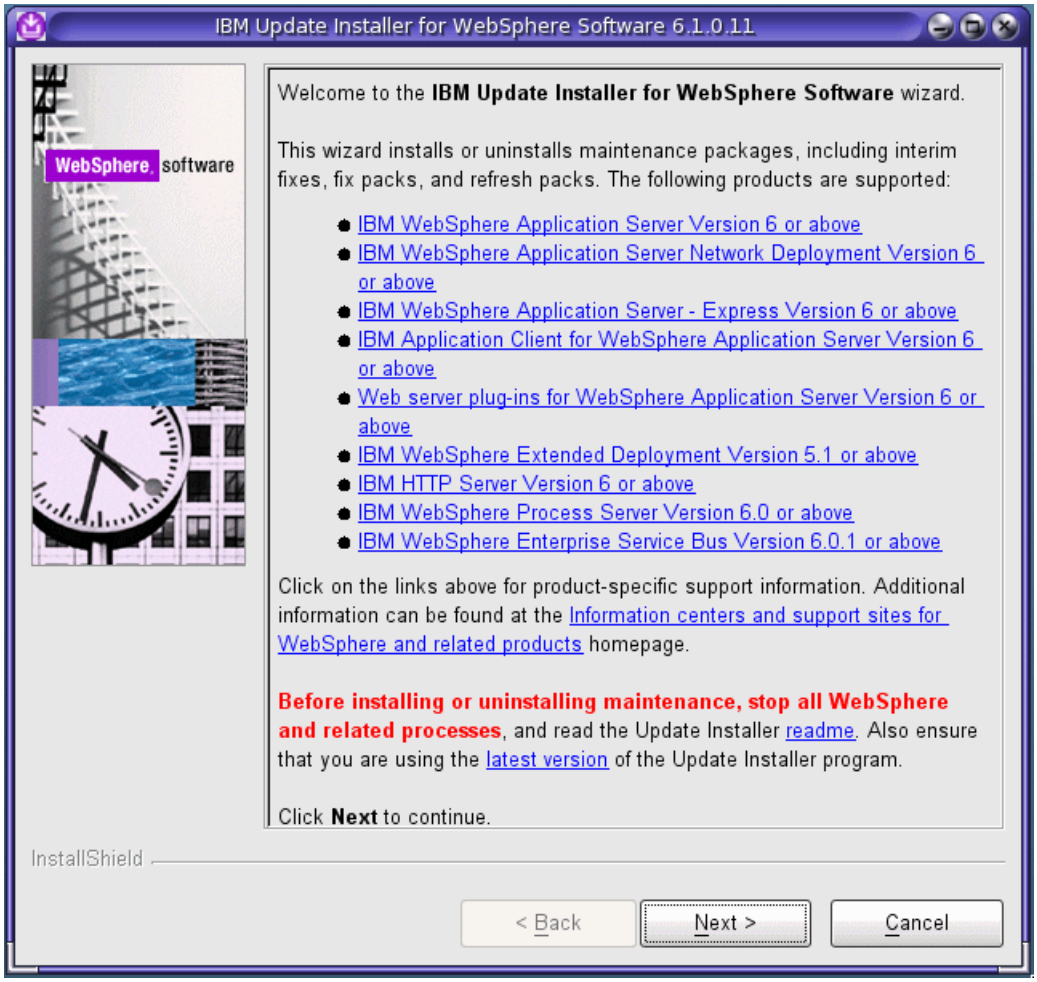


Figure 6-11 Update Installer welcome window

5. Click **Next**.
6. Type or browse to the path of the product binaries to be updated (Figure 6-12 on page 185). Typically, the field already contains the right path to the product binaries, but if your product is in another path, you can go to the location by using the **Browse** button or type in the correct path into field. After you have verified the right path, click **Next**.

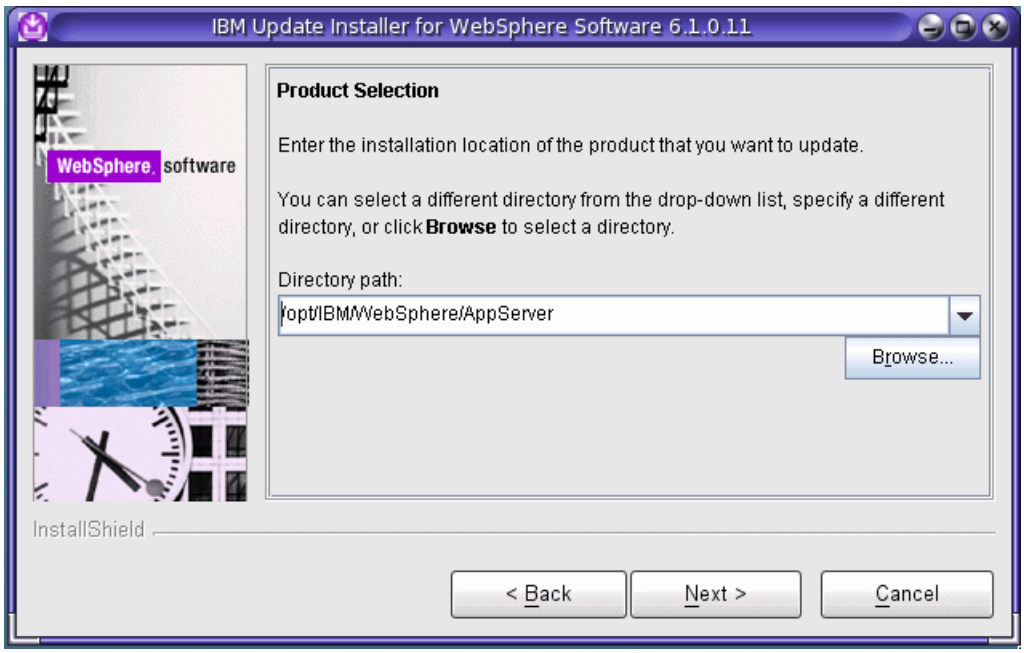


Figure 6-12 Product selection to be updated

Attention: You cannot leave the Directory path field empty because it prevents you from continuing the installation of the Fix Pack.

7. Select the maintenance operation. Click the **Install maintenance package** radio button and click **Next**, as shown in Figure 6-13.

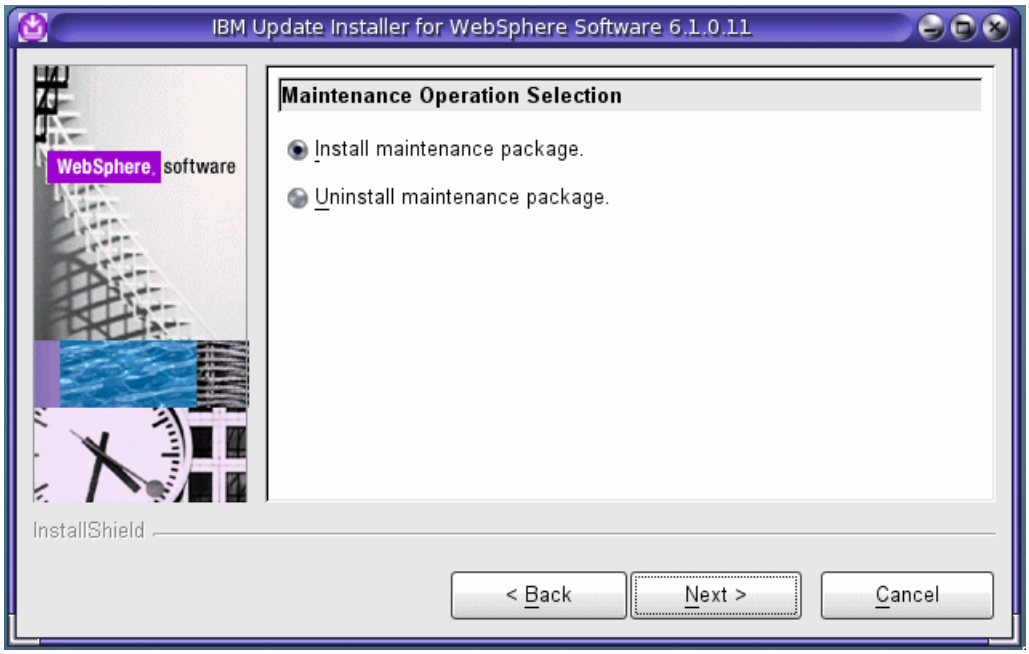


Figure 6-13 Select maintenance operation

8. Enter the path to directory where the maintenance package is available for installation (Figure 6-14). You can specify a directory or click **Browse** to select the path to the maintenance package. When you have entered the path, click **Next**.

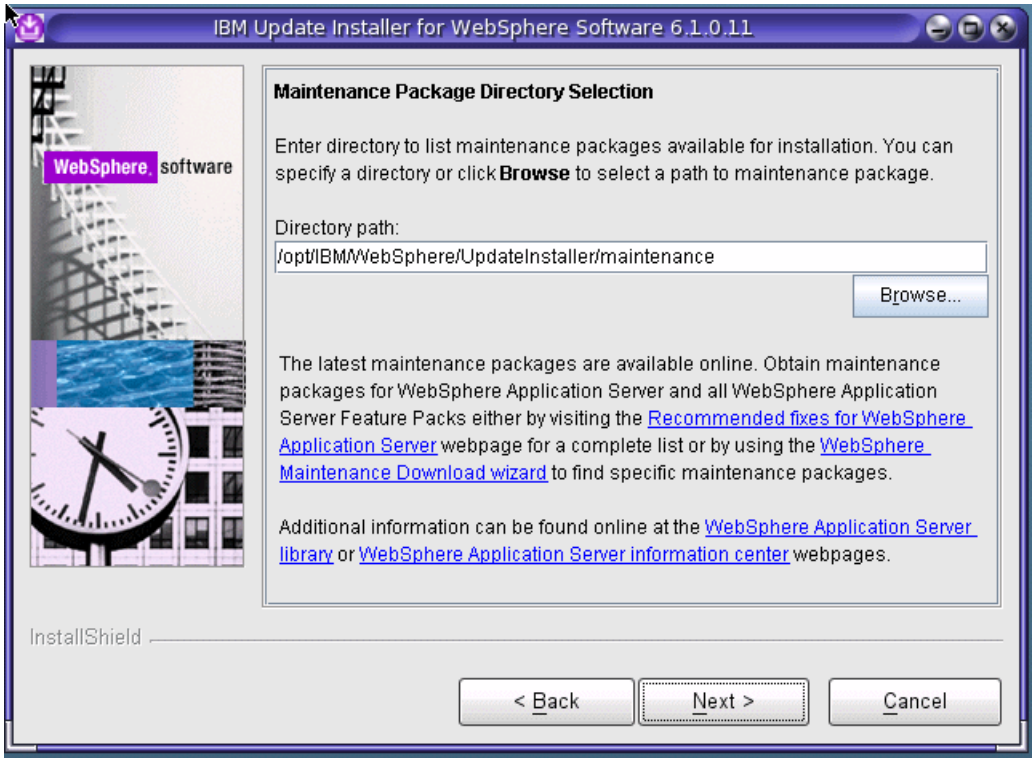


Figure 6-14 Select the directory for the maintenance package

Attention: You cannot leave the Directory path field empty because it prevents you from continuing the installation of the Fix Pack.

9. The next window shows you the available maintenance packages for installation in the given path. Click the check box if it is not already checked (Figure 6-15) and click **Next**.

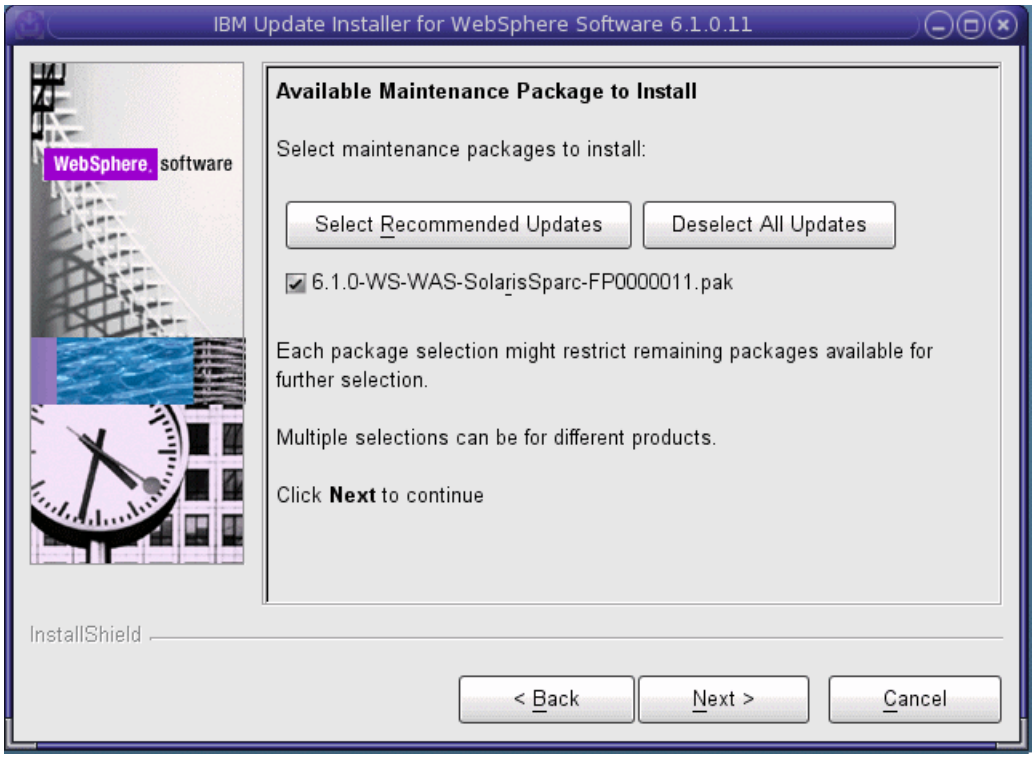


Figure 6-15 Available packages to install

10. Review the Installation Summary window for correctness. Figure 6-16 shows the maintenance package to be installed and the product to be updated. If something needs to be corrected, click the **Back** button; otherwise, click the **Next** button to begin the installation of the maintenance package.

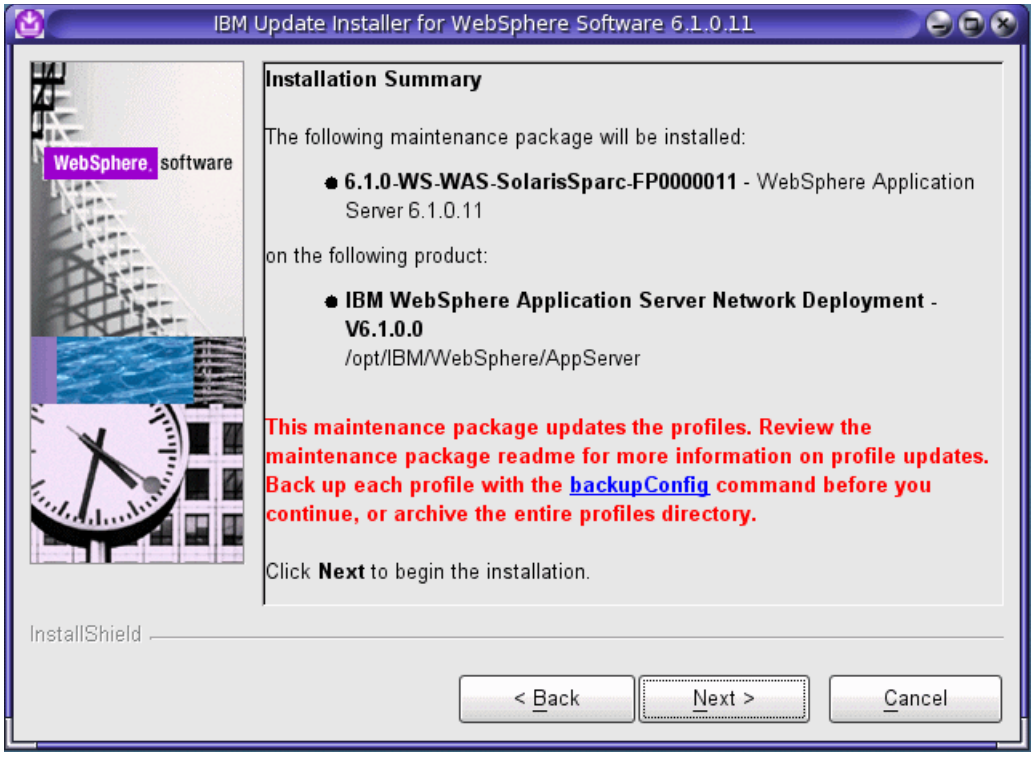


Figure 6-16 Package installation summary window

Important: Before you begin the update installation, make sure you have your application server profiles backed up. We recommend that you back up your profiles or archive the whole profiles directory for fast roll back.

11. After the installation is completed, you see the Installation Complete window (Figure 6-17 on page 189). You can click the **Relaunch** button if you are planning to install or uninstall another maintenance package. If you are not going to do any more maintenance installations, click the **Finish** button to exit the installation program.

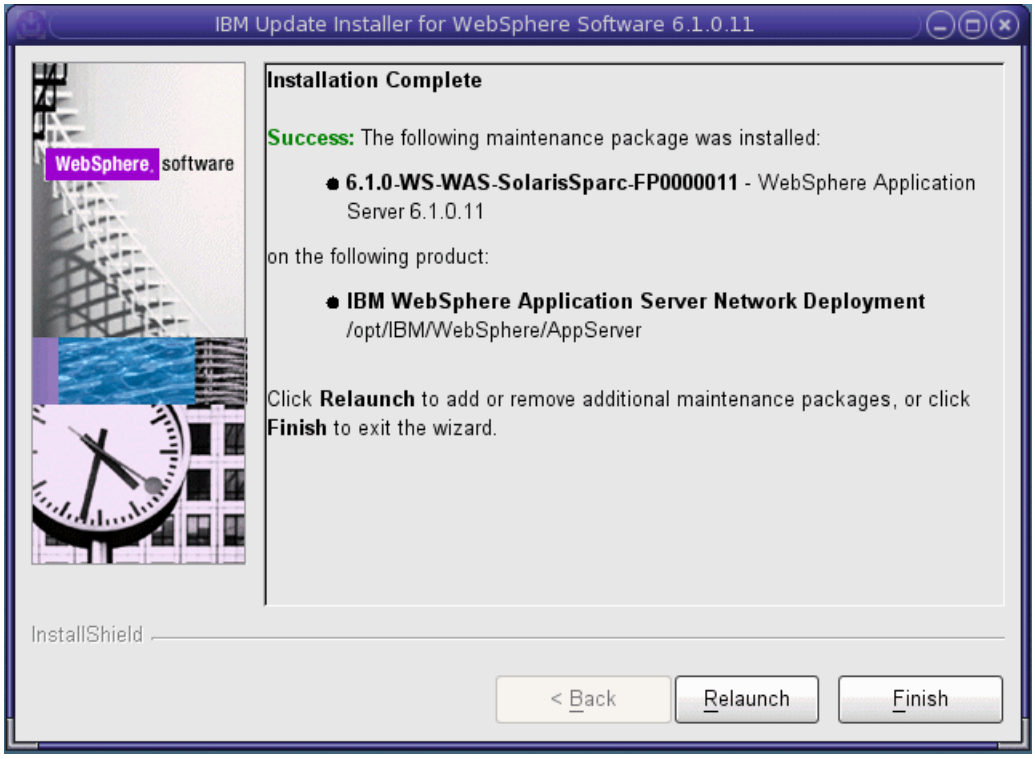


Figure 6-17 Installation Complete window

When you have finished your Fix Pack installation, log in to a local zone and verify that installation is propagated to the local zone. Use the commands in Example 6-6 to verify installation propagation.

Example 6-6 Verifying propagation

```
bash#cd /opt3/IBMSW/WASprofiles/profiles/AppSrv01/bin
bash#./versionInfo.sh
```

In this example case, the propagation took place and the profile in the local zone was updated. Observe the differences between Example 6-1 on page 175 and Example 6-7. In the latter example, the differences are bolded.

Example 6-7 Output of versionInfo.sh after update process

```
WVER0010I: Copyright (c) IBM Corporation 2002, 2005; All rights reserved.
WVER0012I: VersionInfo reporter version 1.15.1.14, dated 11/17/06

--------------------------------------------------------------------------------
IBM WebSphere Application Server Product Installation Status Report
--------------------------------------------------------------------------------

Report at date and time November 16, 2007 10:11:40 AM PST

Installation
--------------------------------------------------------------------------------
Product Directory        /opt/IBM/WebSphere/AppServer
Version Directory        /opt/IBM/WebSphere/AppServer/properties/version
DTD Directory            /opt/IBM/WebSphere/AppServer/properties/version/dtd
Log Directory            /opt/IBM/WebSphere/AppServer/logs
```

```
Backup Directory
/opt/IBM/WebSphere/AppServer/properties/version/nif/backup
TMP Directory            /var/tmp

Product List
--------------------------------------------------------------------------------
ND                       installed

Installed Product
--------------------------------------------------------------------------------
Name                     IBM WebSphere Application Server - ND
Version                  6.1.0.11
ID                       ND
Build Level              cf110734.37
Build Date               8/31/07

--------------------------------------------------------------------------------
End Installation Status Report
--------------------------------------------------------------------------------
```

This process installs the maintenance package to WebSphere Application Server V6.1. If you need to uninstall the maintenance package, Fix Pack, cumulative fix, or interim fix, see 6.3.4, “Uninstalling a Fix Pack from the Global Zone” on page 190 for more information about the uninstallation of the maintenance pack.

Important: As described in 5.3.4, “Scenario 4: Share the WebSphere Application Server installation with zones from the Global Zone” on page 137, it is important to verify the **setupCmdLine.sh** file for each profile in the local zones to ensure that ITP_LOC points to the writable directory that you selected in 5.3.4, “Scenario 4: Share the WebSphere Application Server installation with zones from the Global Zone” on page 137 after the Fix Pack installation. You should also synchronize the contents of <WAS_HOME>/deploytool/itp to $ITP_LOC by copying the updated files (for example, **ejbdeploy.sh**, the jar files, and the plug-ins' files).

6.3.4 Uninstalling a Fix Pack from the Global Zone

This topic describes the process of uninstalling a maintenance package from the WebSphere Application Server product:

1. Log into the Solaris system as the root user.
2. Start the Update Installer wizard, as shown in Example 6-8.

Example 6-8 Start Update Installer

```
bash#cd /opt/IBM/WebSphere/UpdateInstaller
bash#./setup.sh
```

3. The update command starts the IBM Update Installer for WebSphere software. You should see the welcome window (Figure 6-18 on page 191).

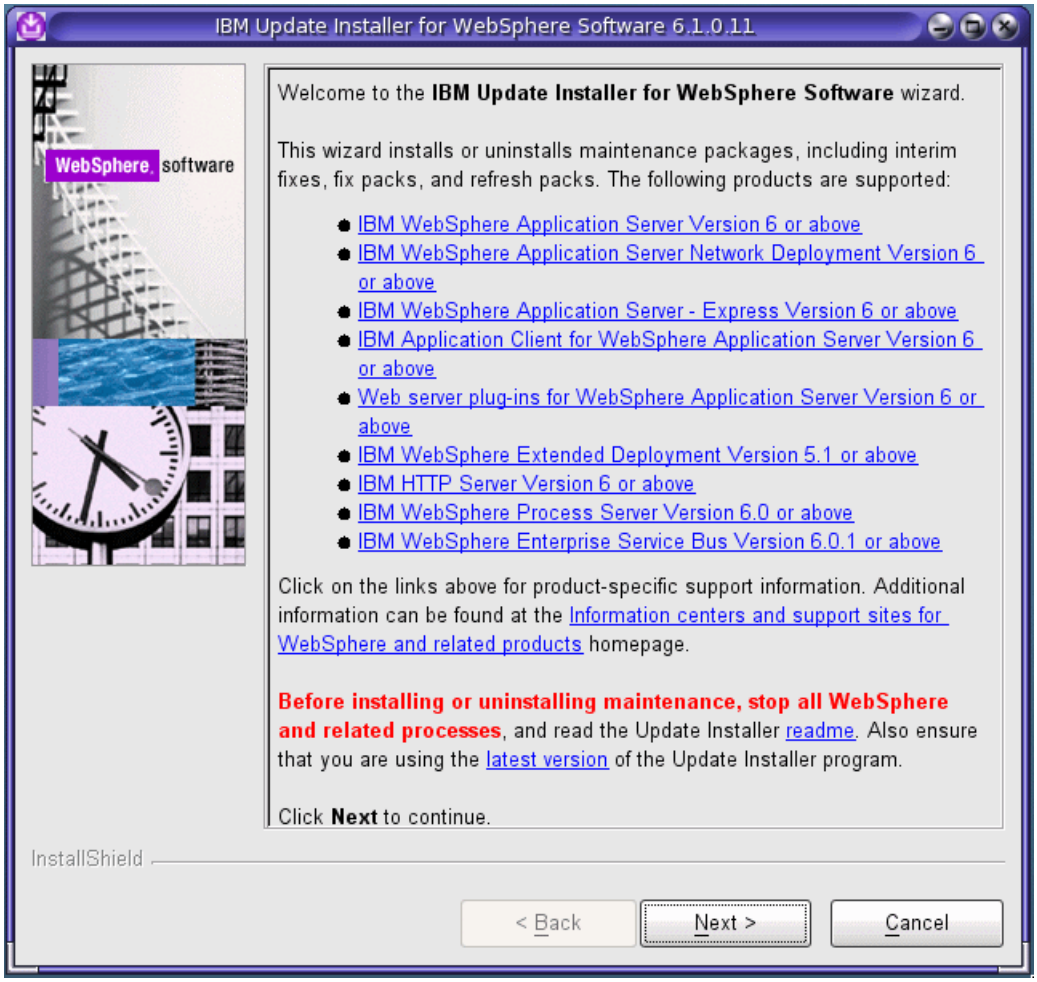


Figure 6-18 Update Installer welcome window

4. Click **Next**.

5. Type the path of the product binaries to be updated (Figure 6-19). Typically, this field already has the right path to the product binaries, but if your product resides in another path, you can go to the correct location using the **Browse** button or type the correct path in the field. After you have verified the right path, click **Next**.

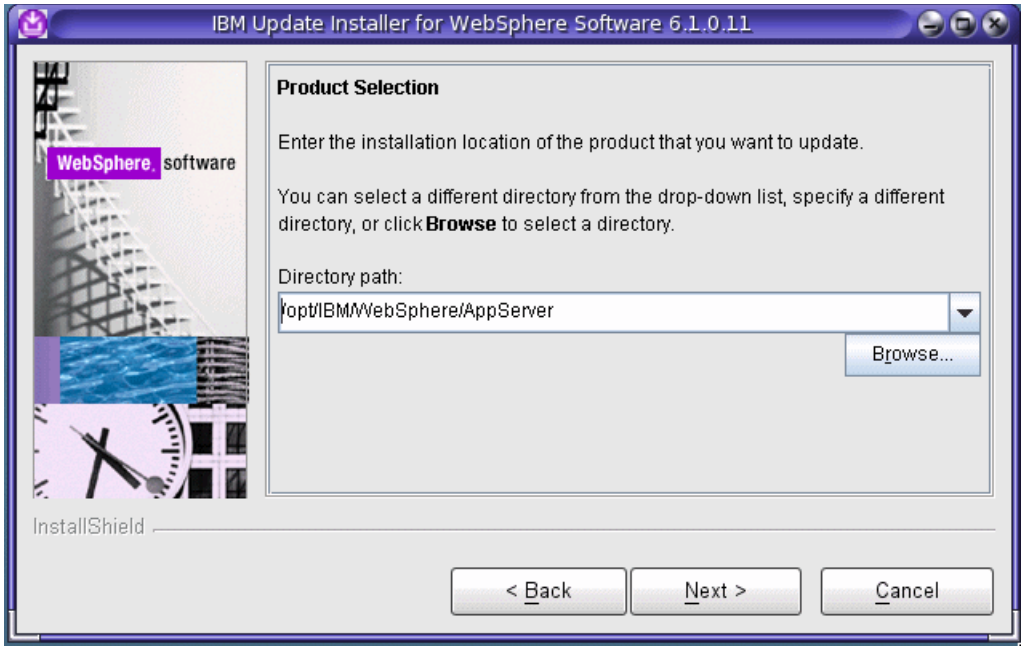


Figure 6-19 Product selection window

Attention: You cannot leave the Directory path field empty because it prevents you from continuing the installation of the Fix Pack.

6. Select the **Uninstall maintenance package** radio button (Figure 6-20 on page 193) and click **Next**.

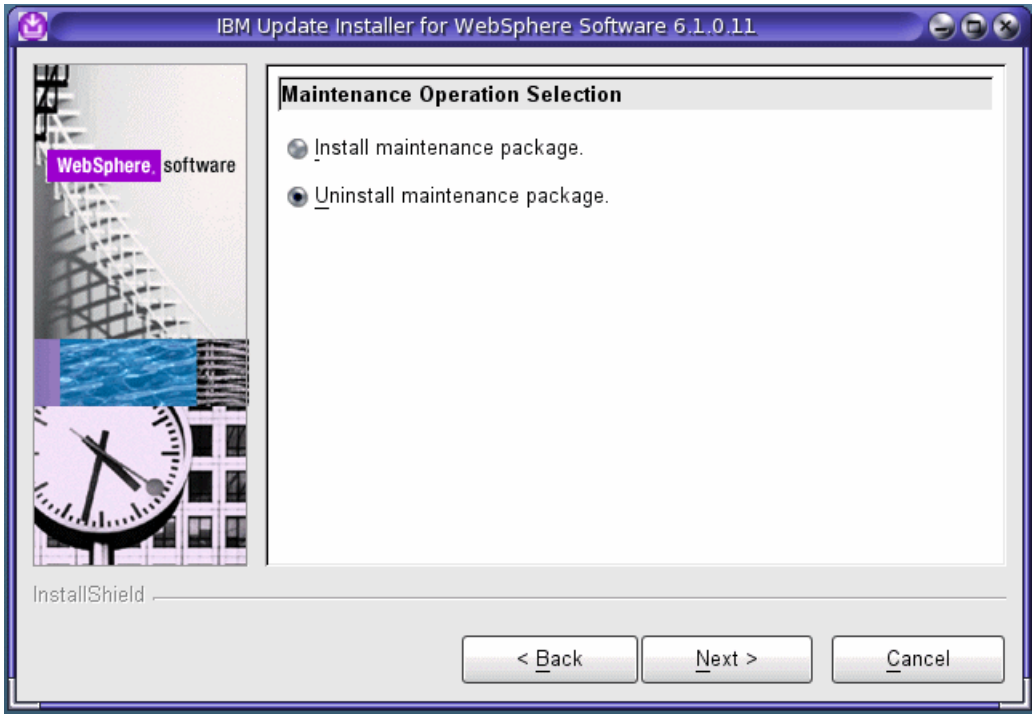


Figure 6-20 Maintenance operation selection

7. Select the maintenance package to be uninstalled and click **Next** (Figure 6-21).

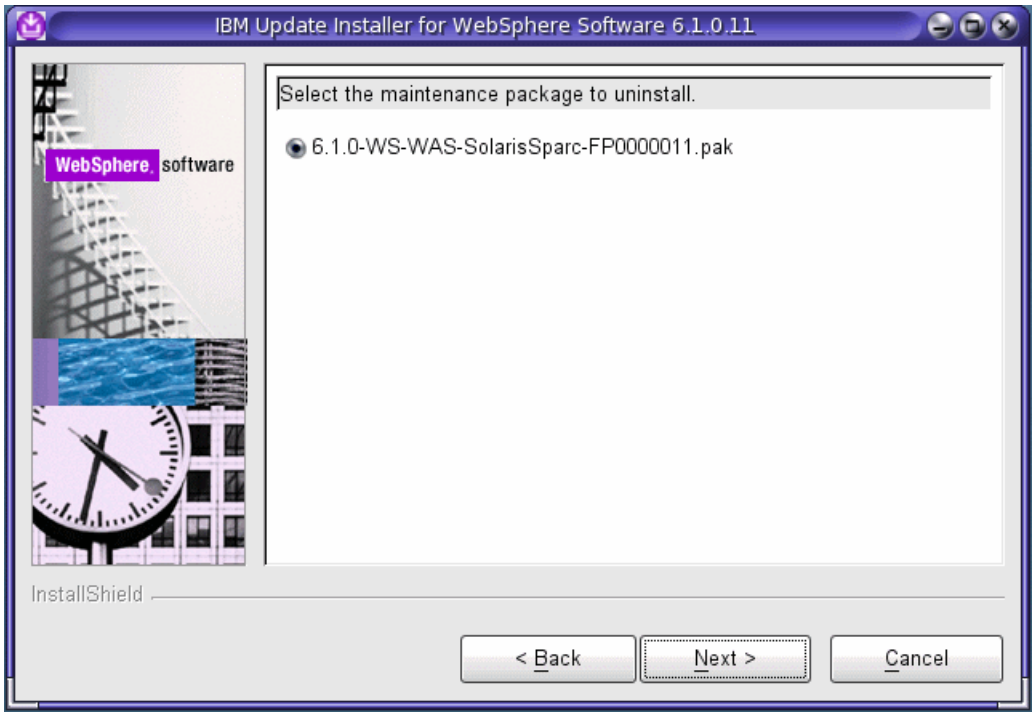


Figure 6-21 Select package to uninstall

8. Review the Uninstallation Summary window (Figure 6-22). If you want to cancel the uninstallation, click the **Cancel** button. To continue with the uninstallation, click **Next**.

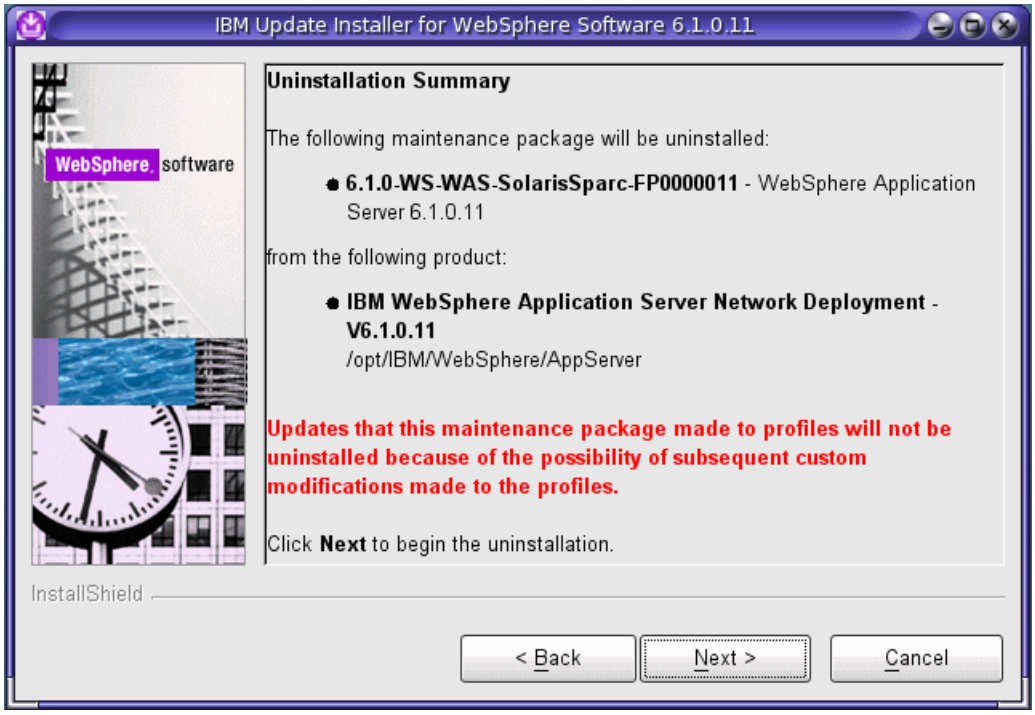


Figure 6-22 Uninstallation summary window

9. If you are planning to install or uninstall another maintenance package, click **Relaunch** in the installation completion window (Figure 6-23). Click **Finish** to exit the installation wizard.

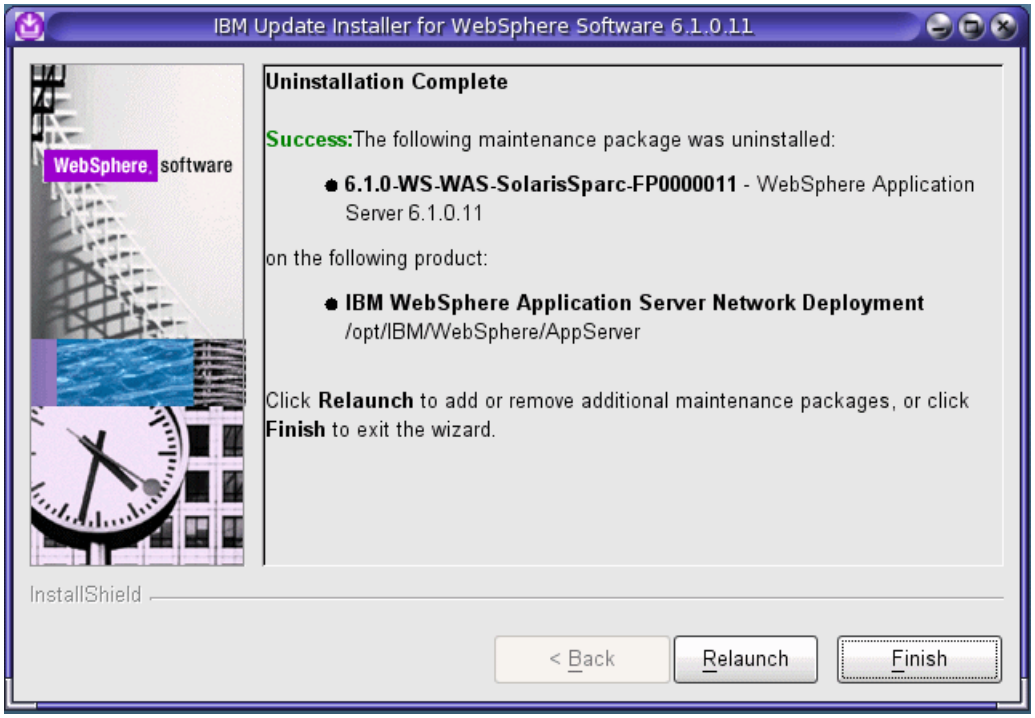


Figure 6-23 Maintenance uninstallation completed

This section describes ways to install and uninstall an Update Installer program, which is needed to update the WebSphere Application Server product. This section also describes the procedures to install and uninstall maintenance packages with the Update Installer program to keep the WebSphere Application Server installation up to date.

6.4 Rehosting of an existing WebSphere Application Server environment

A WebSphere Application Server environment on production usually has many customizations. You may have business applications deployed, WebSphere Application Server thread pools adjusted, JVM tuned, resources for the back-end system defined, and so on. Some situations can demand a rehosting of your WebSphere Application Server environment, perhaps for one or more of the following reasons:

- The current physical system is being re-purposed.
- The current WebSphere Application Server environment needs to be relocated to a newer or larger capacity system.
- The current file system where WebSphere Application Server or zones reside is being replaced with a new file system.
- A new hosting service provider has been chosen.
- Management decided to do server consolidation in your datacenter.

To plan accordingly, you can begin to categorize the various needs for the rehost and consider the best approaches to preserve the current environment and roll it out identically in the new environment. First, we clarify that this is not a migration project; thus, the underlying system infrastructure is to remain on the same vendor's platforms. In our case, the system infrastructure is Solaris based systems that are hosting the WebSphere environment. As a guideline, you can define the types of rehost that are required as follows:

- From a physical system to a physical system
- From a physical system to a virtual system (for example, Dynamic System Domains to Containers)
- From a logical system to a virtual system (for example, LDoms to Containers)
- From a physical system to a logical system (for example, Dynamic System Domains to LDoms)
- From a virtual system to a logical system (for example, Containers to LDoms)
- From a virtual system to a virtual system on different hosts (for example, Containers to Containers)

It could also be a combination of rehost scenarios, such as a move from a container on a physical to a container on an LDom. In this section, we discuss some approaches that you can take as best practices to make the rehosting effort relatively simple and reliable.

A recommended approach is to use what IBM provides in the Installation Factory, as discussed in 4.4, "Installation Factory" on page 93, to relocate the WebSphere Application Server environment from one system to another. You can also find it online at:

http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg24016062

You can use this approach in any of type of rehost situation. It requires some preparation work to create the Custom Installation Package (CIP) and so on, but once you have this package prepared, you have a very flexible solution.

Another approach is using Solaris Containers. If your WebSphere Application Server environment has bee virtualized with Solaris Containers, as discussed in Chapter 5, “Configuration of WebSphere Application Server in an advanced Solaris 10 Environment” on page 115, you can easily relocate Solaris Containers from one system to another with the zone migration capabilities. This approach is the most simple and agile solution.

We discuss details about these two approaches to relocate your WebSphere Application Server environment as part of the rehosting requirements in detail in the next two sections.

6.4.1 Relocating WebSphere Application Server environment using CIP

This procedure allows you to prepare for a move of a WebSphere Application Server installation from one system (that is, source system) to another (that is, destination system) using a feature of the WebSphere Application Server Installation Factory to create a CIP. In the following example, it is assumed that these preparations are already made:

- Installation Factory is installed on the source system.

  You can download and install Installation Factory binaries, as shown in “Creating a WebSphere Application Server CIP” on page 96.

- WebSphere Application Server binaries and WebSphere Application Server maintenance packages are downloaded to the source system. WebSphere Application Server is installed and a stand-alone application server profile exists in the source system.

  In this example, we use the WebSphere Application Server V6.1 installation image, WebSphere Application Server Fix Pack 11, and SDK Fix Pack 11. Refer to 6.1.1, “Update strategy for WebSphere Application Server V6.1” on page 170 for more information about Fix Pack installation strategies.

- The destination Solaris system is prepared and configured.

  In this example, we install CIP to the directory location /opt/IBM/WebSphere, so the destination can be a standard Solaris system, Whole Root Zone, or Sparse Root Zone.

A Customized Installation Package (CIP) created with the Installation Factory can contain a previously exported configuration archive of an existing WebSphere Application Server profile located in a custom defined location along with WebSphere Application Server installation binaries, maintenance packages, user files and folders, and scripts.

A configuration archive (CAR) file captures the configuration of a pre-existing, stand-alone WebSphere Application Server profile for later restoration on another application server node. The CAR includes a profile configuration, but also installed enterprise archives (EAR) files. The CAR can help clone the original profile to another machine or system. Therefore, creating CIP with CAR is very powerful when creating a large scale WebSphere Application Server topology.

Consideration: If you are intending to clone a WebSphere Application Server profile that uses messaging, you must also include a script to configure the service integration bus (SIB). The original SIB from the original profile is not portable and therefore it is not included in the configuration archive file.

Creating a Custom Installation Package of a pre-existing, stand-alone profile

We give an example of this task here; its topology is as follows:

The WebSphere Application Server profile is located in a custom defined location, the Fix Pack level is 6.1.0.11, and one additional “business application” named ItsoTest is deployed to the destination system. The business application is using a custom Web host name to serve Web content. We create an installation package from this scenario and install it to another system:

1. Create an archive file of your current profile on the source system, as shown in Example 6-9.

Example 6-9 Creating a profile archive for CIP using the command line

```
source_host# cd /opt/IBM/WebSphere/AppServer/bin/
source_host# ./wsadmin.sh -username <username> -password <password> -conntype NONE
WASX7357I: By request, this scripting client is not connected to any server
process. Certain configuration and application operations will be available in
local mode.
WASX7029I: For help, enter: "$Help help"
wsadmin>$AdminTask exportWasprofile { -archive /tmp/itsoTest.car }

wsadmin>exit
```

2. Go to the directory where the Installation Factory is located and issue the command **ifgui.sh**, as shown in Example 6-10.

Example 6-10 Starting Installation Factory

```
source_host# cd /opt/IBM/WebSphere/InstallationFactory/bin
source_host# ./ifgui.sh
```

3. You will see the IBM Installation Factory dialog window, as shown in Figure 6-24. Click the **Create New Customized Installation Package** icon.

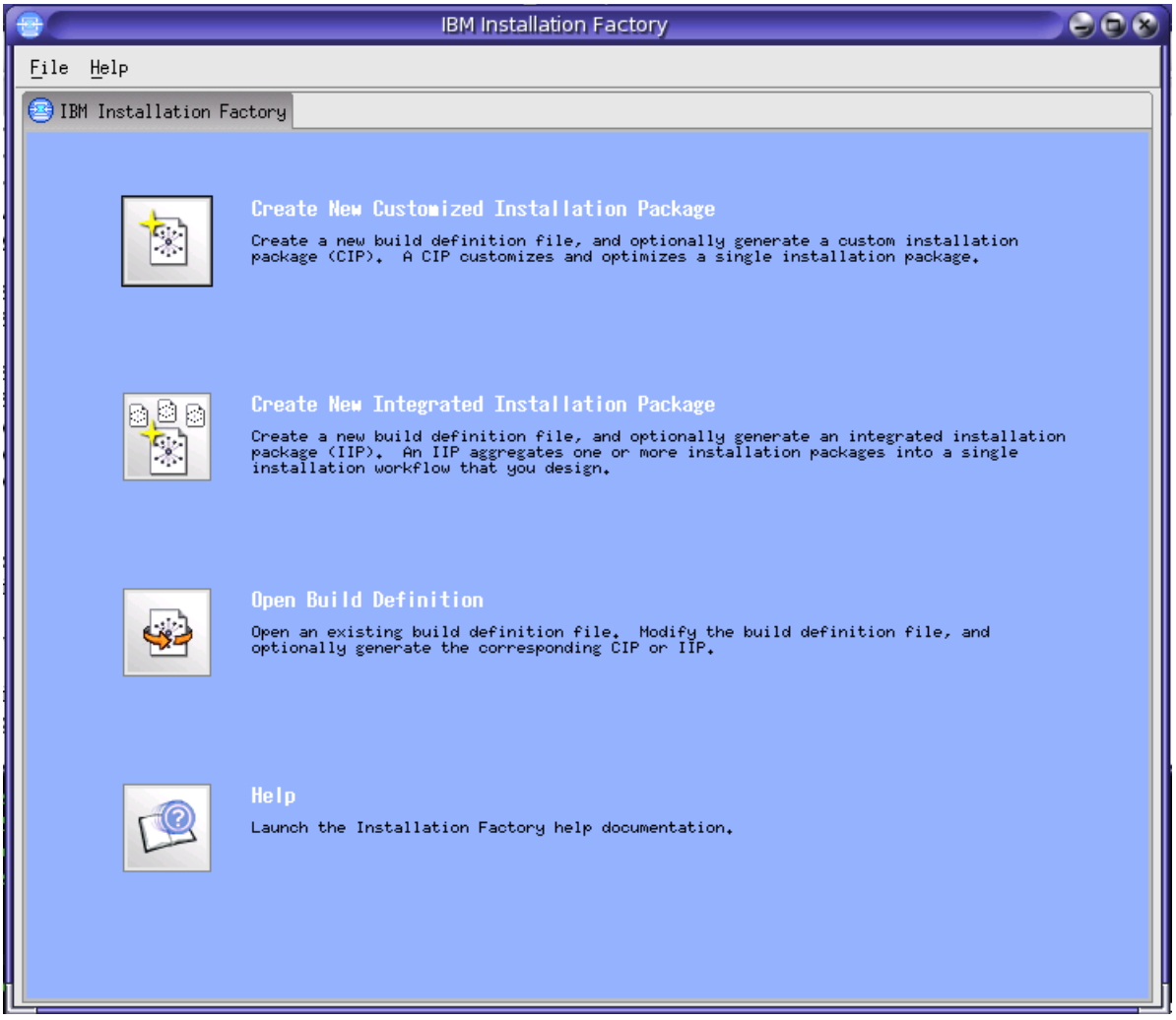


Figure 6-24 IBM Installation Factory

4. Choose **WebSphere Application Server** and **Network Deployment** edition, as shown in Figure 6-25. Click the **Finish** button.

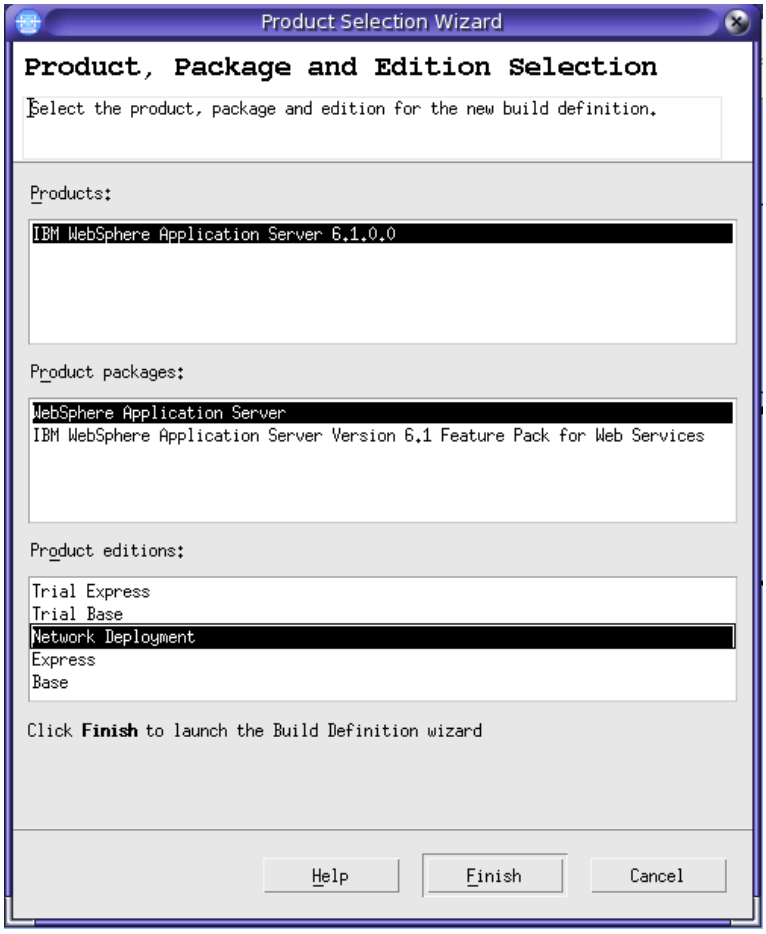


Figure 6-25 Customized Installation Factory: Select the product

5. Click the **Connected mode** radio button and choose the appropriate platform for your environment. In this example, we select the **Sun Solaris Sun Sparc 32 bit** platform, as shown in Figure 6-26. Click the **Next** button.

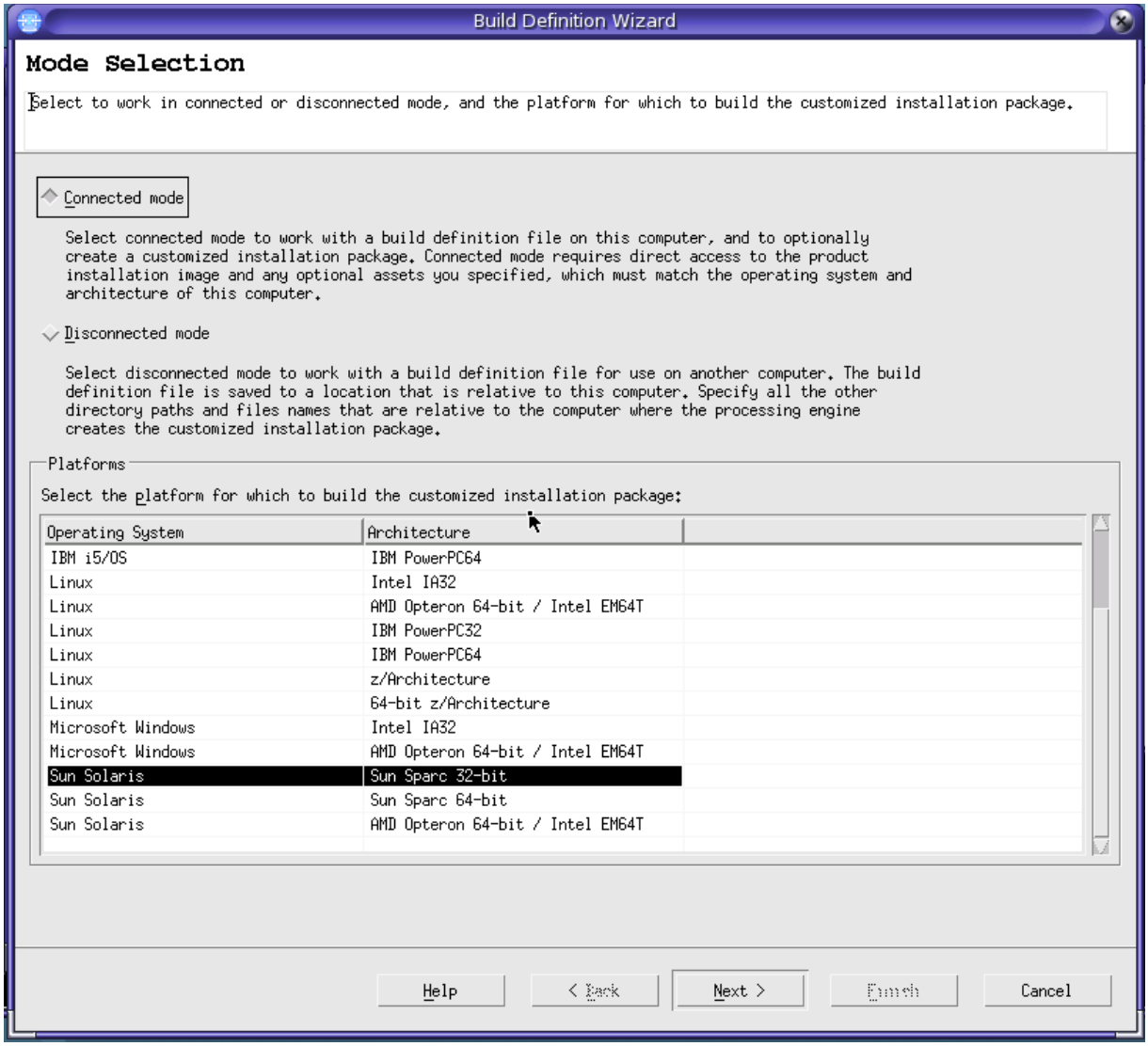


Figure 6-26 Customized Installation Factory: Mode Selection

6. Change the Identifier field to, for example, com.itso.test and keep the default value for the Version field, as shown in Figure 6-27. Click **Next**.

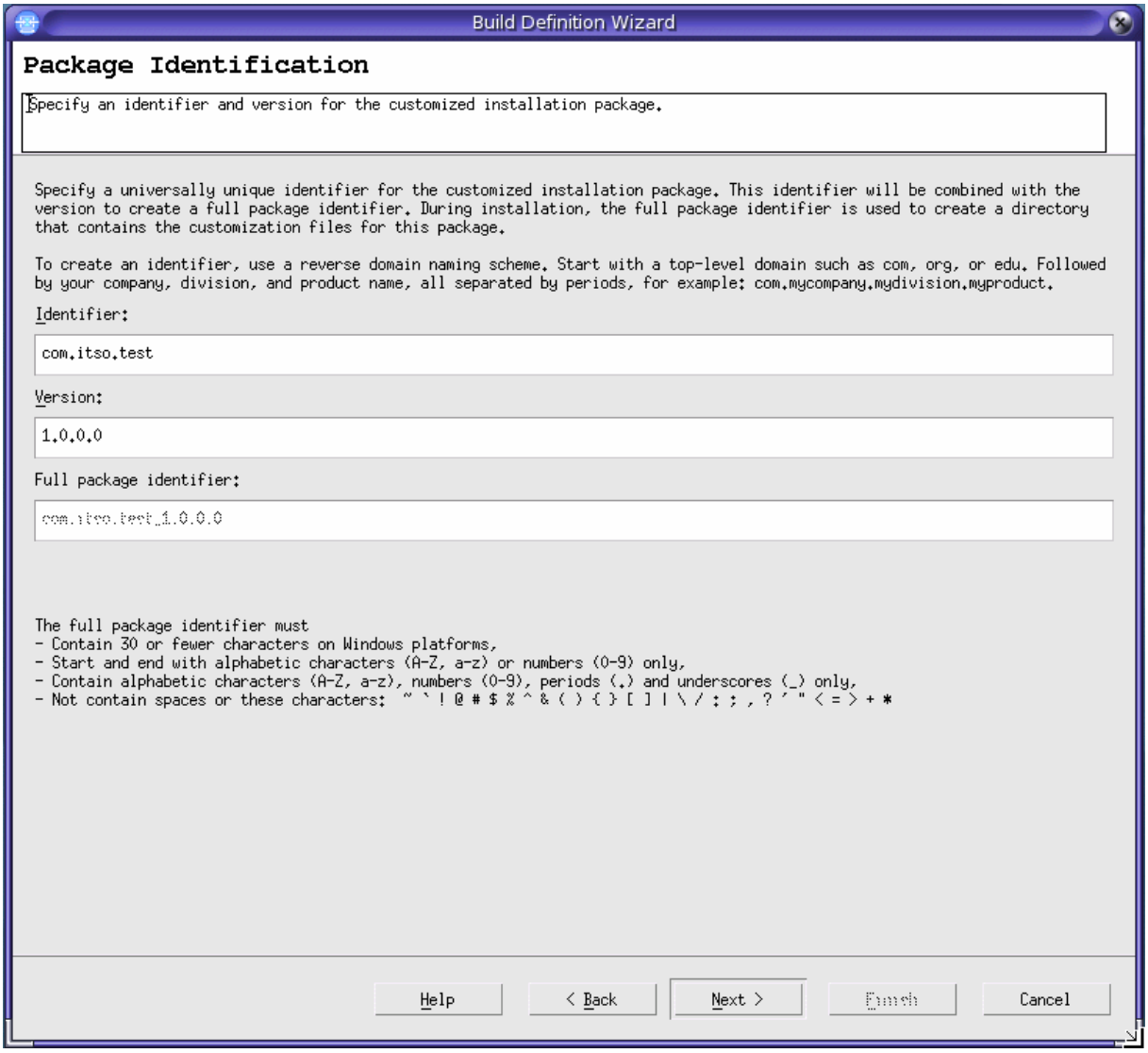


Figure 6-27 Build Definition Wizard: Package Identification

7. Keep the Build definition directory path field as default and edit the CIP build directory path field, as shown in Figure 6-28. Click **Next**.

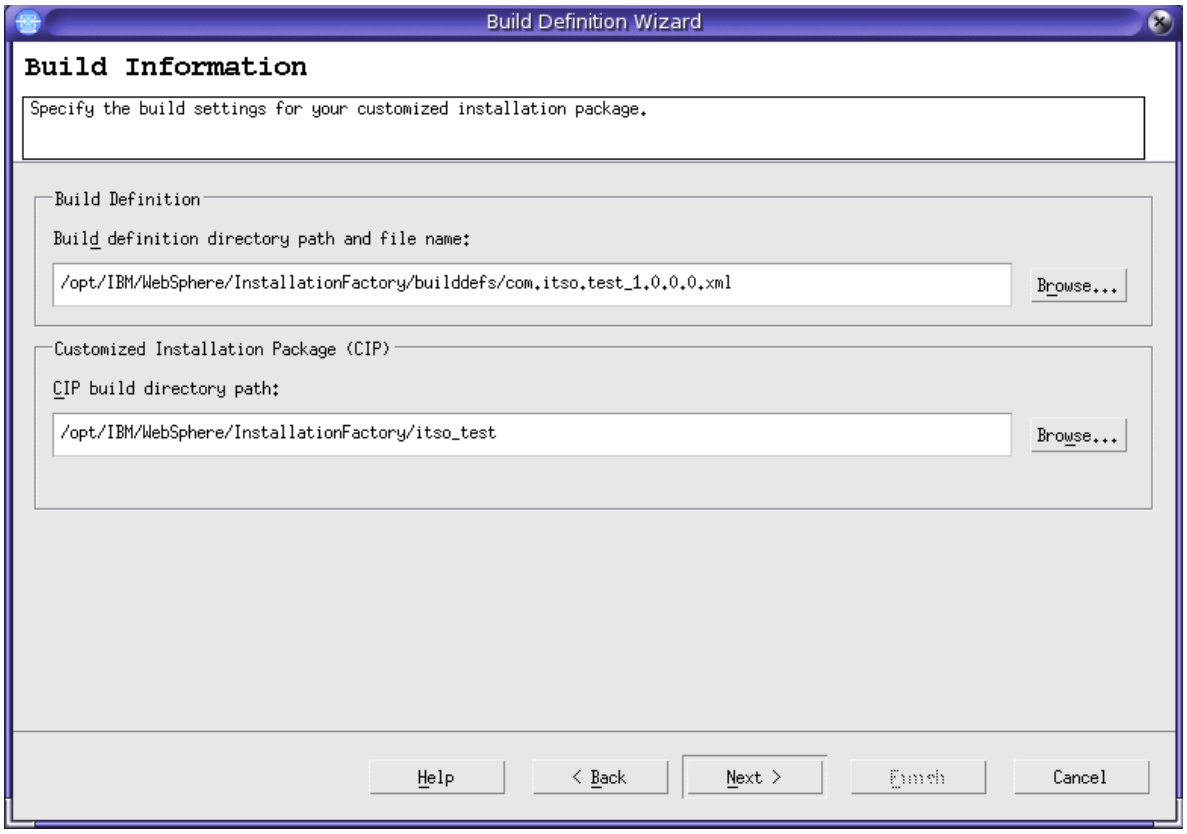


Figure 6-28 Build Definition Wizard: Build Information paths

8. Type or browse the path containing the WebSphere Application Server installation image and click **Next** (Figure 6-29).

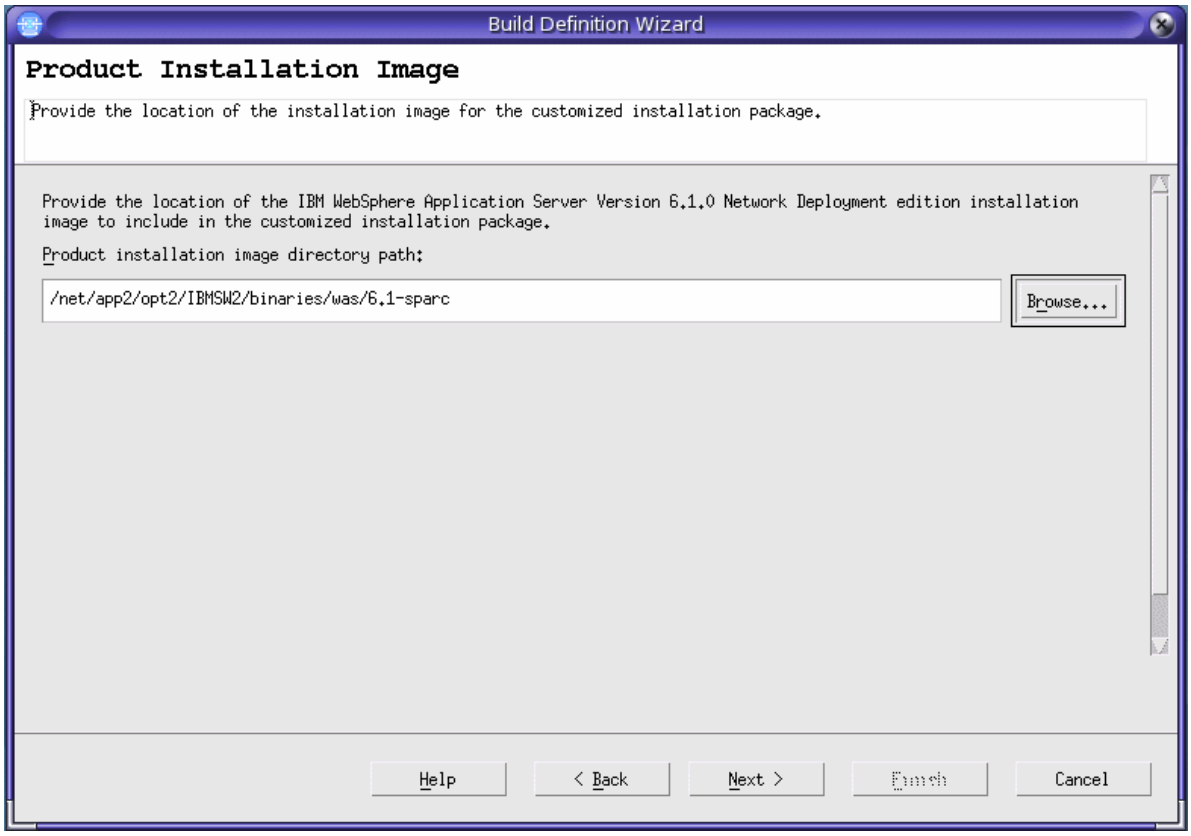


Figure 6-29 Build Definition Wizard: Installation image path

9. Select the **Core product files** check box and click **Next** (Figure 6-30).

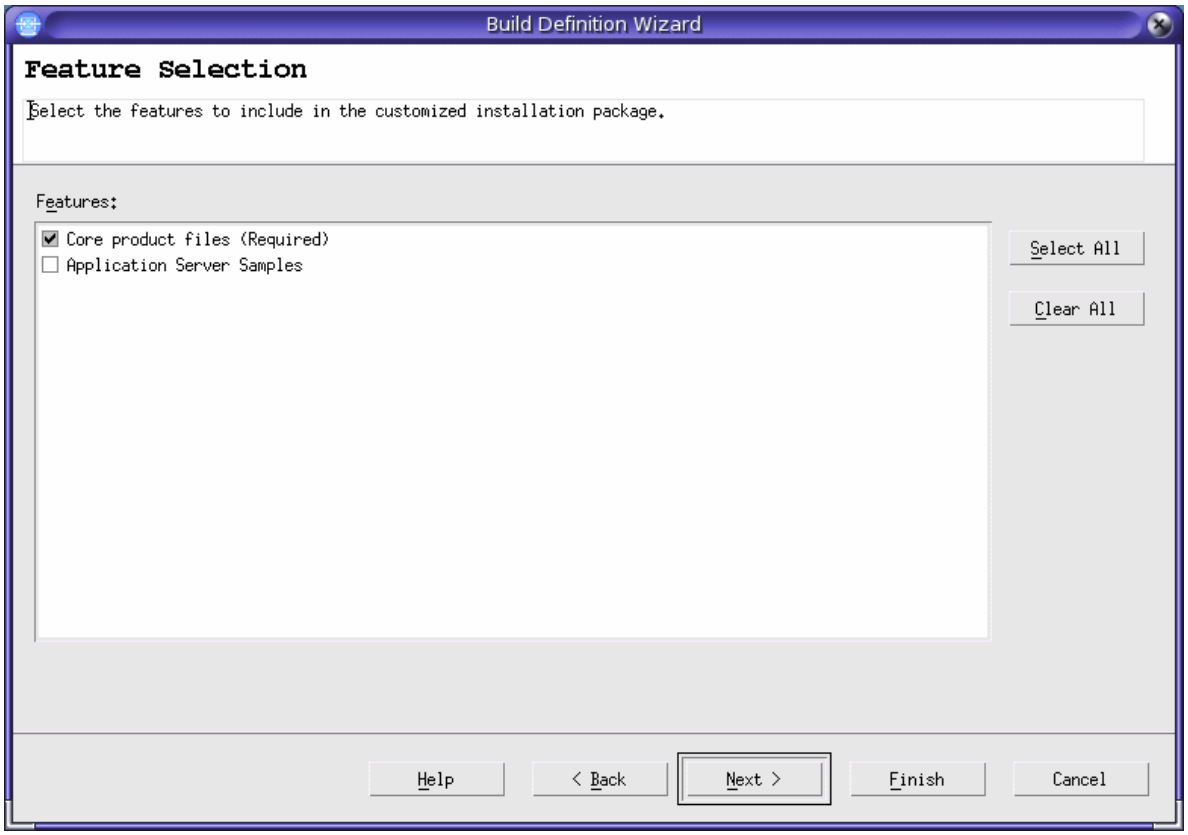


Figure 6-30 Build Definition Wizard: Feature Selection

10. Browse your Fix Pack, SDK Fix Pack, and interim fixes you want to include in the installation package and click **Next** (Figure 6-31 on page 205).

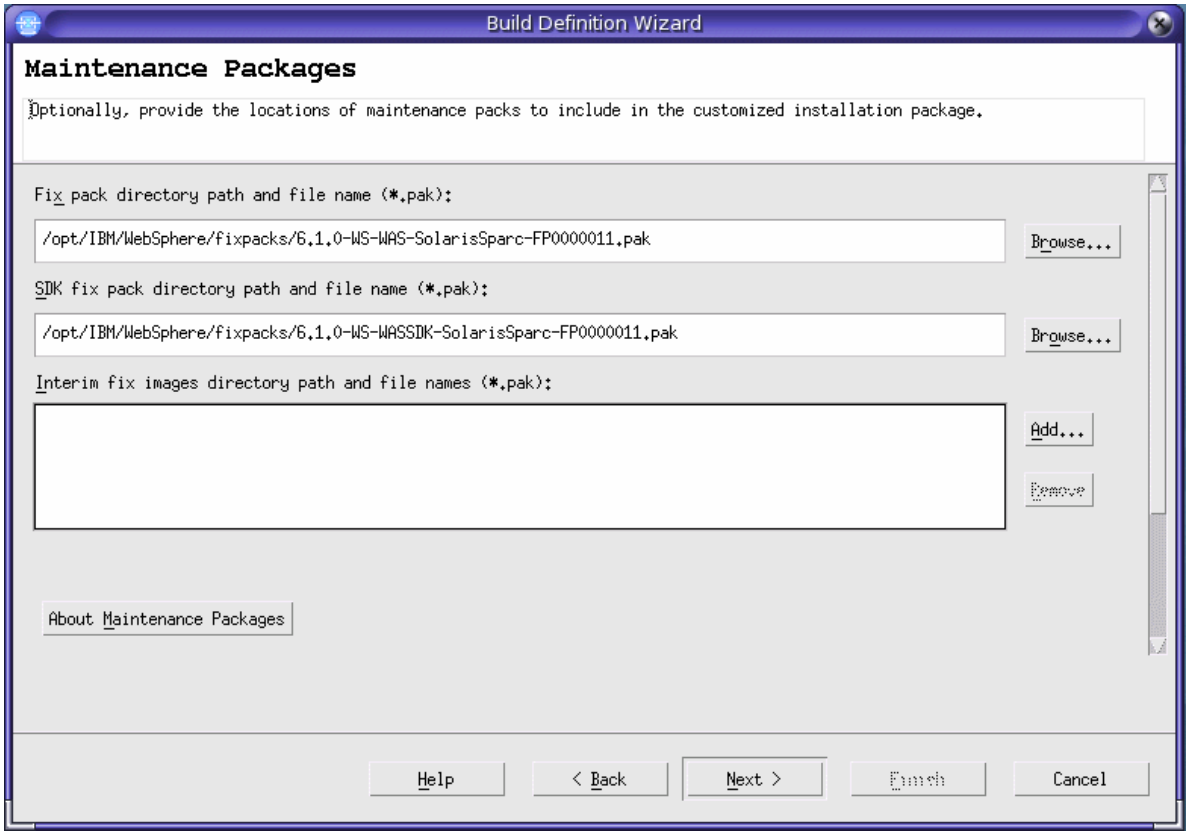


Figure 6-31 Build Definition Wizard: Maintenance Packages

11. Optional: You can choose here the scripts you want to be run after an installation. In this example, we skip this step and go to the next window by clicking **Next** (Figure 6-32).

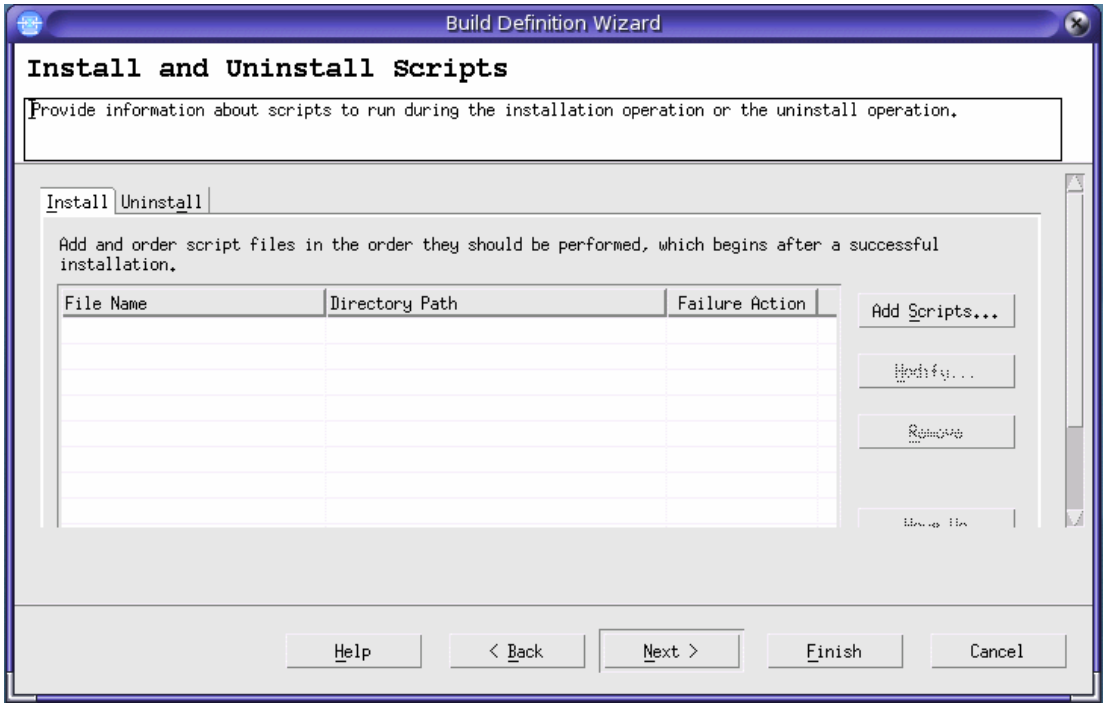


Figure 6-32 Build Definition Wizard: Install and Uninstall scripts

12. Choose the **Stand-alone Application Server** from the Profile types menu, click the **Allow creation of new profiles using the customizations** check box, and after that, click the **Add Configuration Archive** button, as shown in Figure 6-33, “Build Definition Wizard: Profile Customization main window” on page 206.

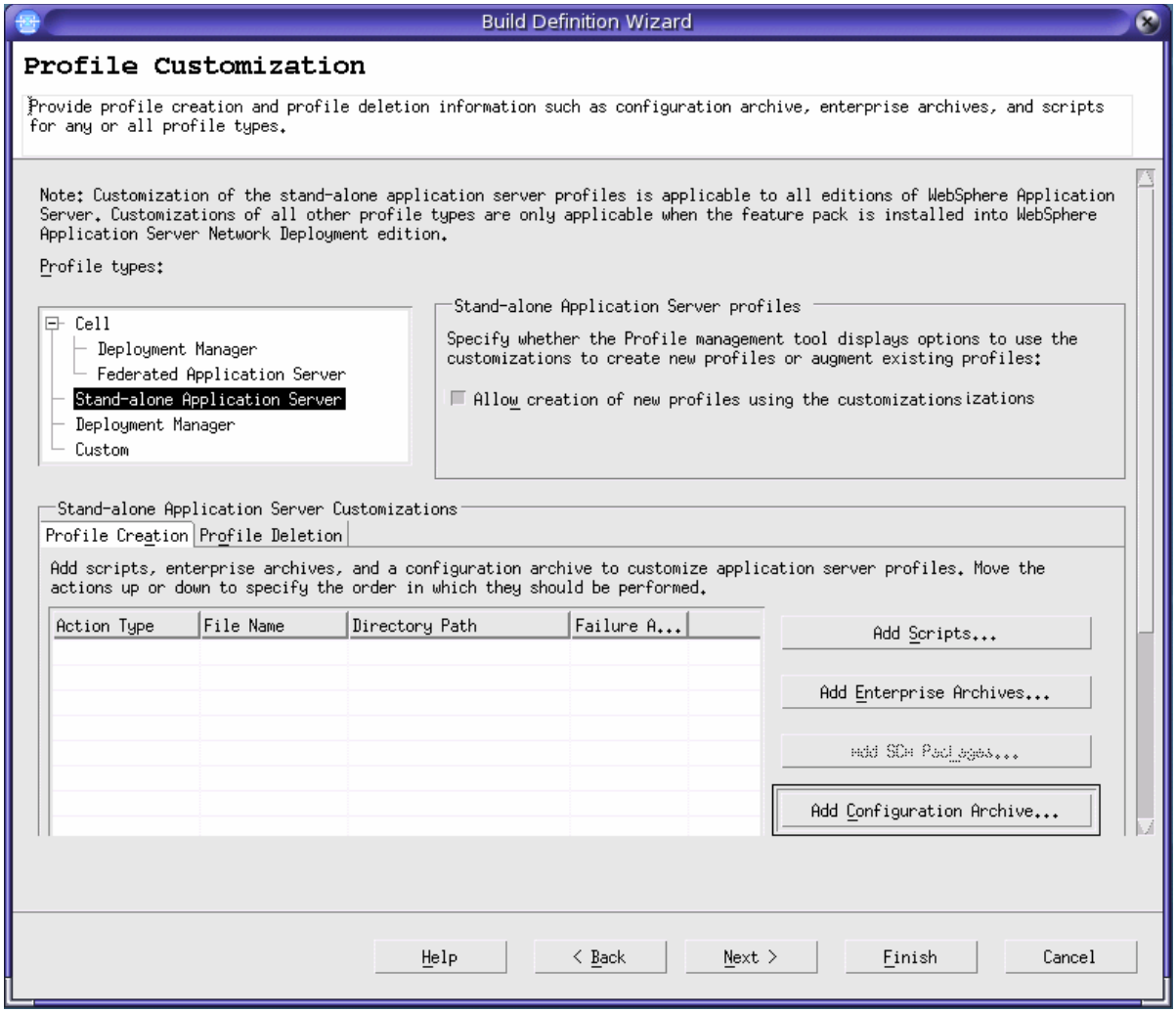


Figure 6-33 Build Definition Wizard: Profile Customization main window

13. From the Add a Configuration Archive (CAR) file window shown in Figure 6-34 on page 207, type or browse the whole path to the configuration archive file, leave the **Unrecoverable error** radio button selected, and click **OK**.

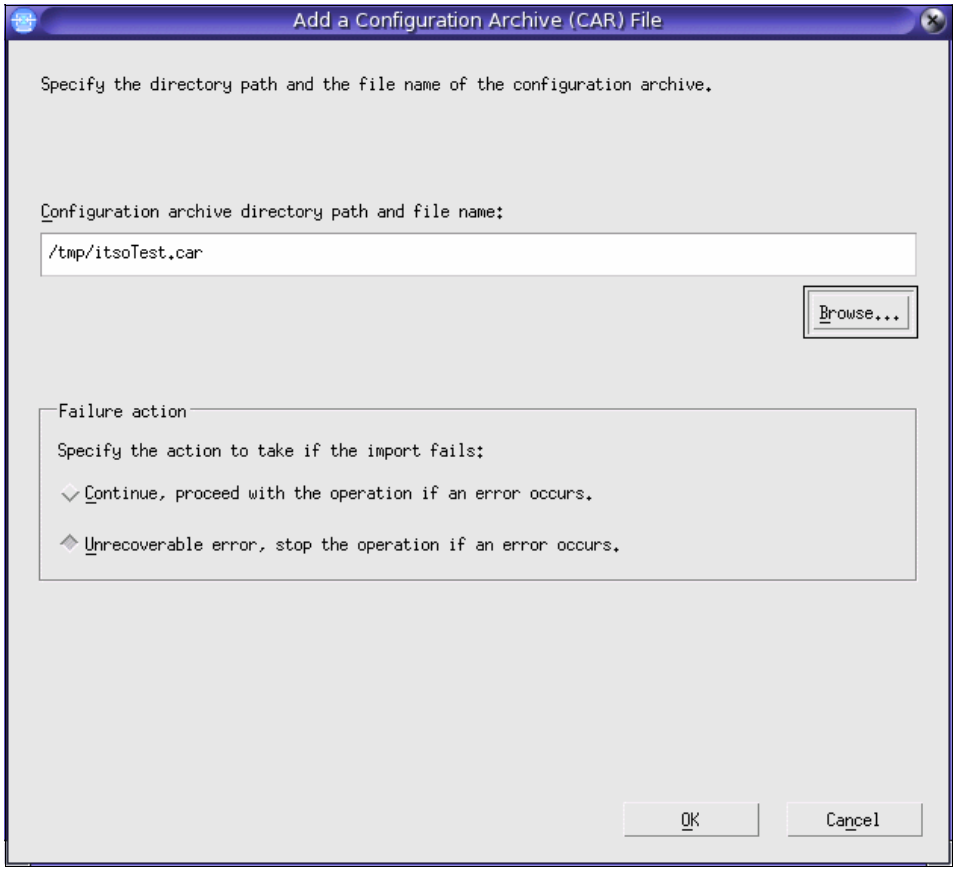


Figure 6-34 Add a Configuration Archive window

14. Click the **Add Enterprise Archive** button shown in Figure 6-33, “Build Definition Wizard: Profile Customization main window” on page 206.

15. Browse an EAR file to be included and type in the Application display name. Leave the **Unrecoverable error** radio button selected and click **OK**, as shown in Figure 6-35.

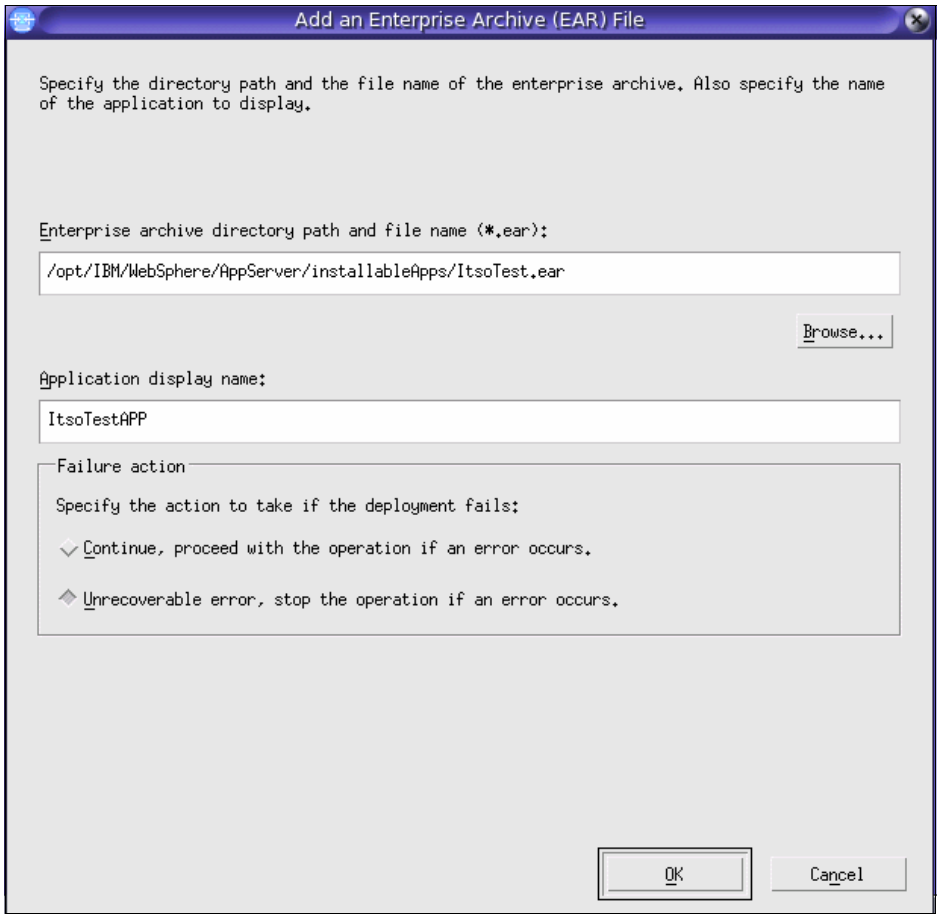


Figure 6-35 Add an Enterprise Archive window

16. Click **Next** (Figure 6-36).

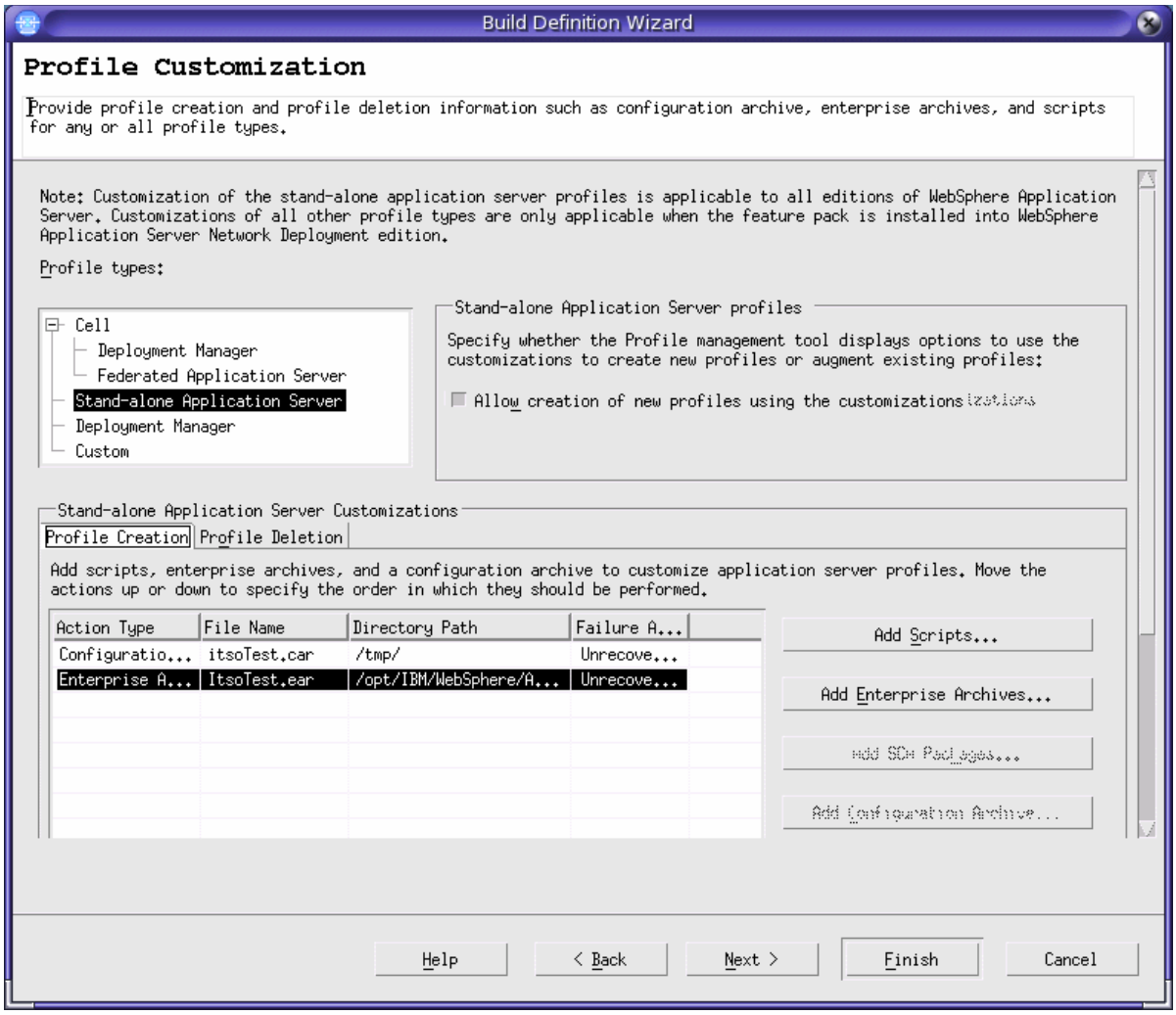


Figure 6-36 Build Definition Wizard: Profile Customization completed

17. Optional: You can add additional files or folders in the Additional Files window shown in Figure 6-37. In this example, we skip the Additional Files window by clicking **Next**.

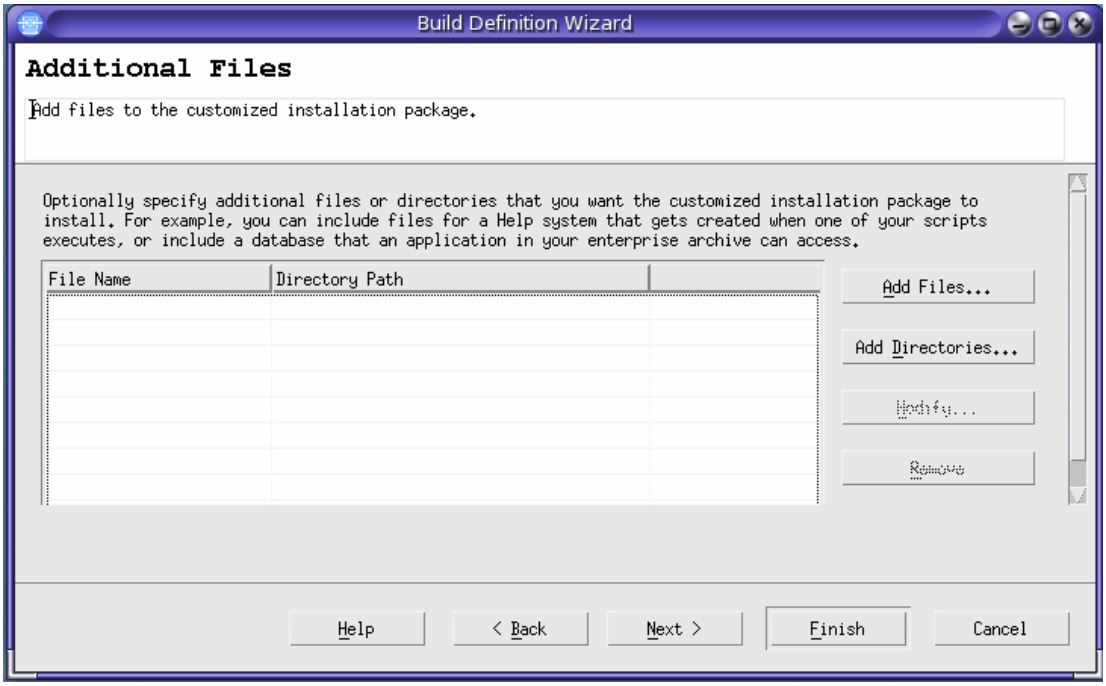


Figure 6-37 Optional: Additional files window

18. Optional: You can fill in the Organization field and Description text area of the Authorship panel with your organization's details, as shown in Figure 6-38. Click **Next** to continue.

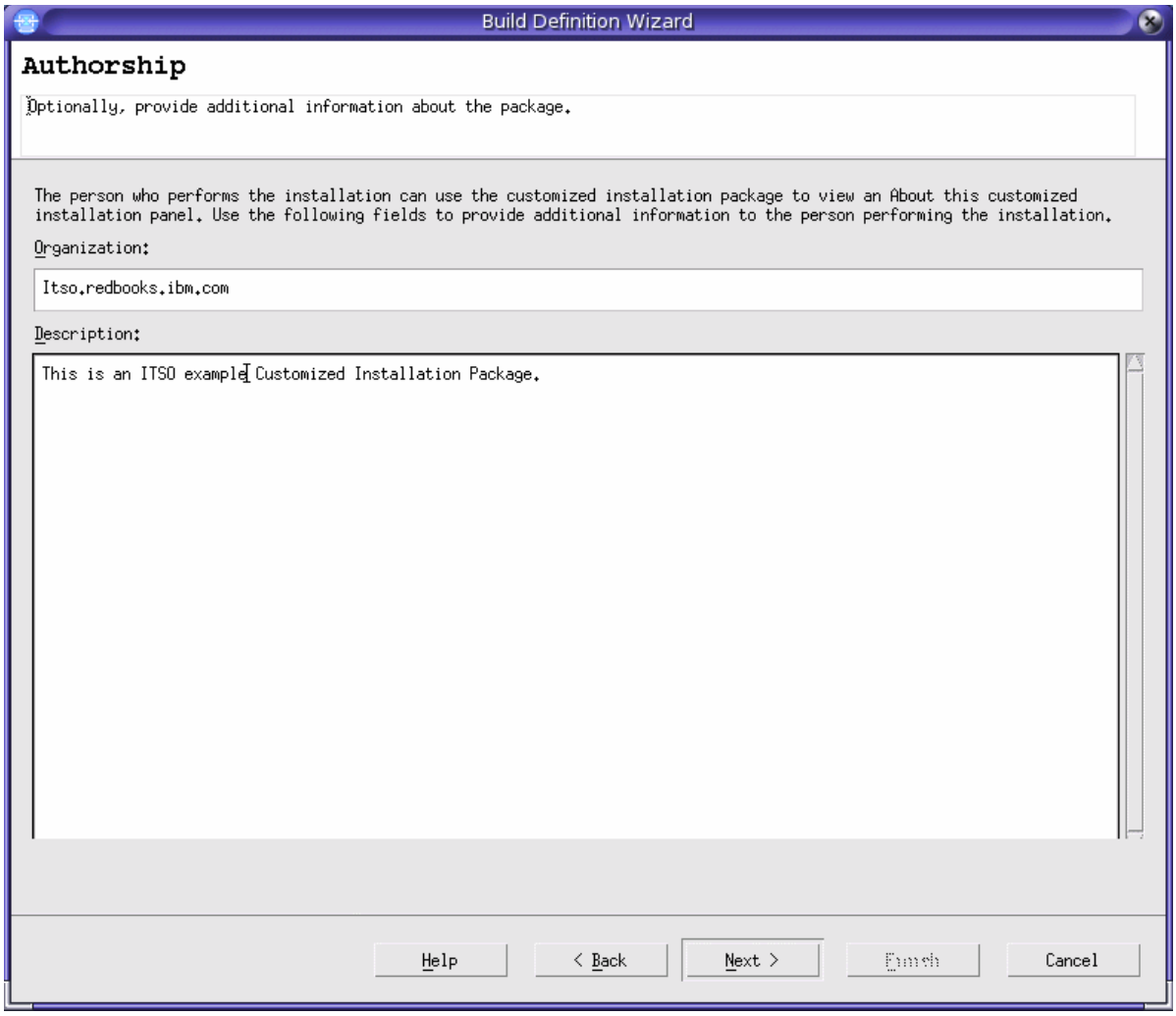


Figure 6-38 Optional: Authorship window

19. Click the **Save build definition file and generate customized installation package** radio button. Optionally, you can click **Estimate Size and Available Space** to learn about the required free space and installation package size information. Review your installation package information and click **Finish**, as shown in Figure 6-39.

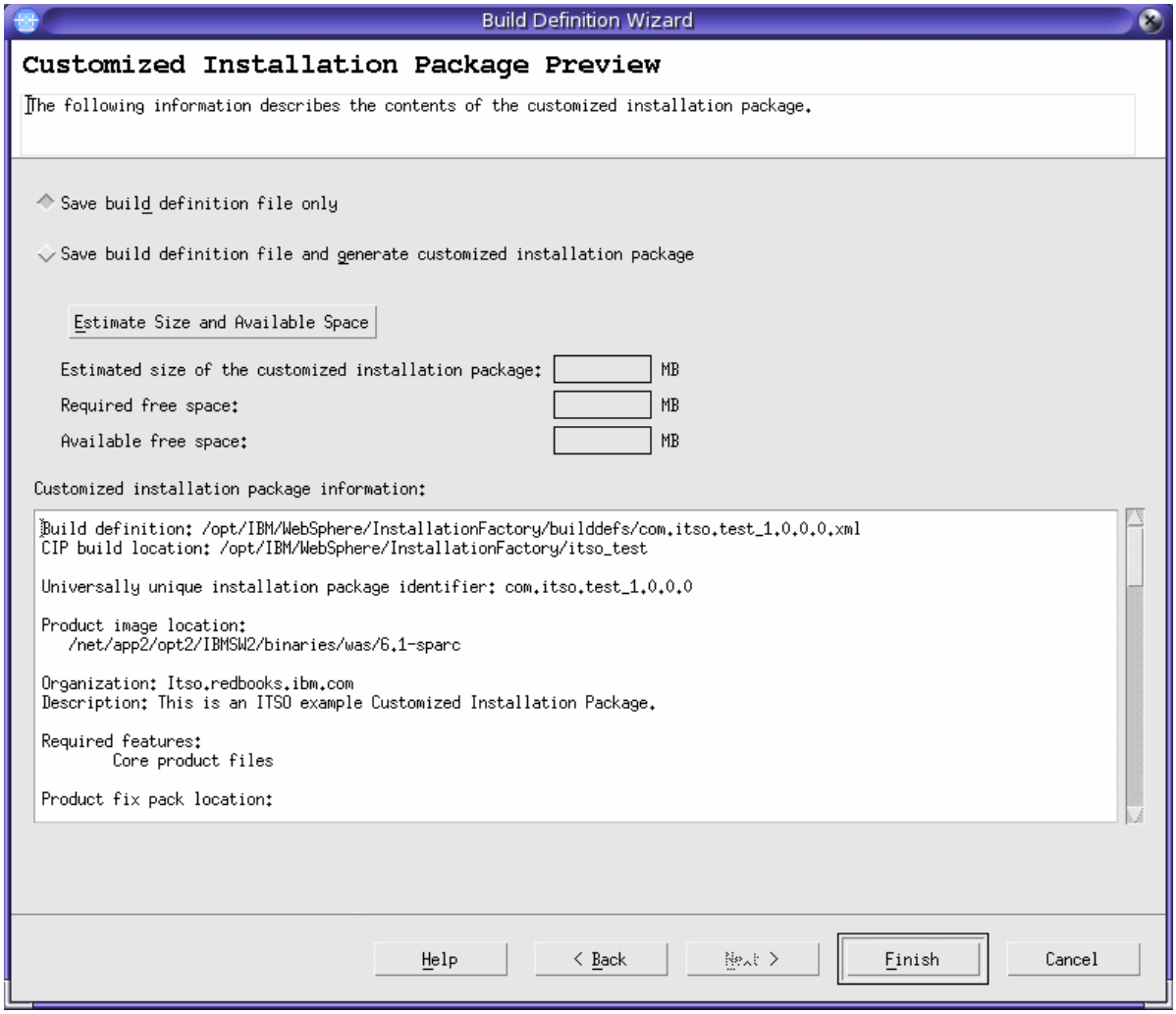


Figure 6-39 CIP Preview window

Package creation takes a little time and after it is completed, click **OK** in the Successful Build window (Figure 6-40). Select **File** → **Exit** in the IBM Installation Factory window (Figure 6-24 on page 198) to close the Installation factory application.

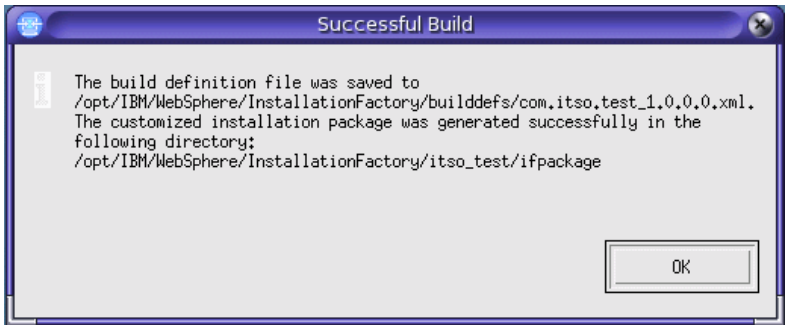


Figure 6-40 Build Successful window

As a result of this process, the Installation Factory creates a Customized installation package with a pre-existing stand-alone Application Server Profile, two fix packs, and one additional customized business application in it.

Installing Customized Installation Package

To install the newly created Customized Installation Package, follow the steps described here:

1. You can archive your installation package with a standard Solaris utility like tar, as shown in Example 6-11.

Example 6-11 Creating a tar package of installation binaries

```
source_host# cd /opt/IBM/WebSphere/InstallationFactory/itso_test/ifpackage/
source_host# tar cvf itsoTest_inst.tar *
---output omitted---
a ./WAS/panels/coexistencePanel.xml 12K
a ./WAS/readme/ 0K
a ./WAS/readme/wasstyle_nlv.css 4K
a ./WAS/readme/readme_en.html 8K
a ./WAS/responsefile.nd.txt 47K
```

2. After transferring the installation package to the destination system, unpack the tar file and start installation wizard by issuing the **install** command that is included in the installation package, as shown in Example 6-12.

Example 6-12 Unpack the installation package and start the installation

```
dest_host#mkdir -p /tmp/install
dest_host#cd /tmp/install
dest_host#tar xvf /<path>/<to>/itsoTest_inst.tar
---output omitted---
x ./WAS/panels/coexistencePanel.xml, 12268 bytes, 24 tape blocks
x ./WAS/readme, 0 bytes, 0 tape blocks
x ./WAS/readme/wasstyle_nlv.css, 3685 bytes, 8 tape blocks
x ./WAS/readme/readme_en.html, 7701 bytes, 16 tape blocks
x ./WAS/responsefile.nd.txt, 47704 bytes, 94 tape blocks
dest_host#./WAS/install
InstallShield Wizard

Initializing InstallShield Wizard...

Searching for Java(tm) Virtual Machine...
........
```

3. From the Installation wizard welcome window, you can optionally see information about the custom installation package by clicking the **About this custom installation package** button. To continue with the installation wizard, click **Next** in the Welcome window (Figure 6-41).

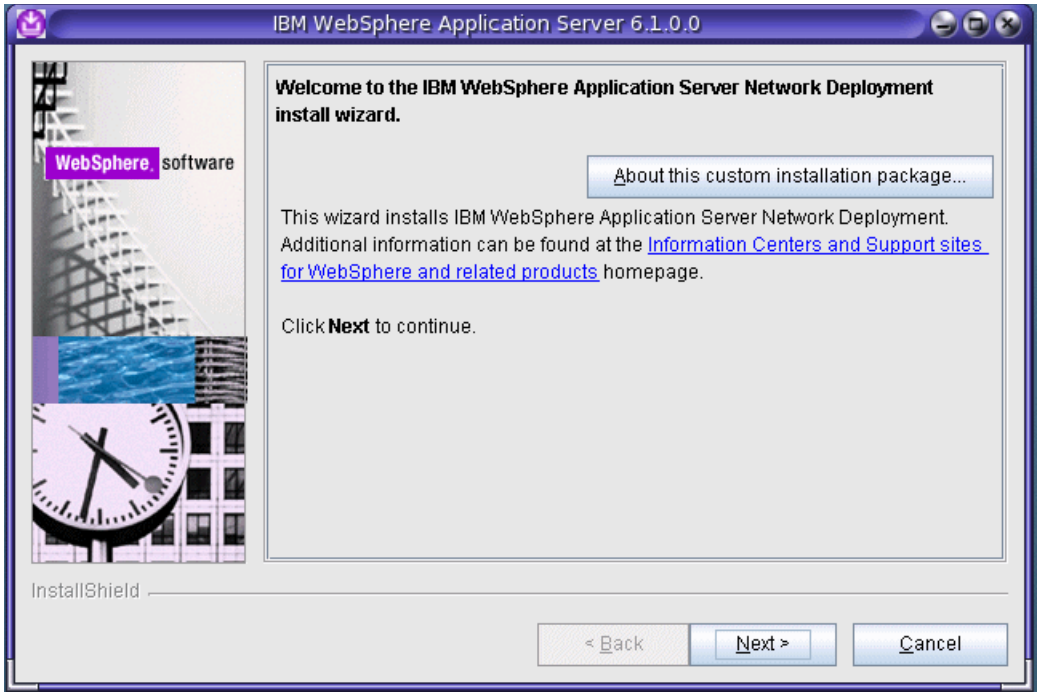


Figure 6-41 CIP installation wizard: Welcome window

4. Click to accept the license agreement and click **Next**, as shown in Figure 6-42 on page 215.

   After accepting the licensing terms, the installation verifies your system. The system must meet the prerequisites for the installation to continue. If you receive an error message indicating that your system does not meet the prerequisites, cancel the installation, make the required corrections, and restart the installation.

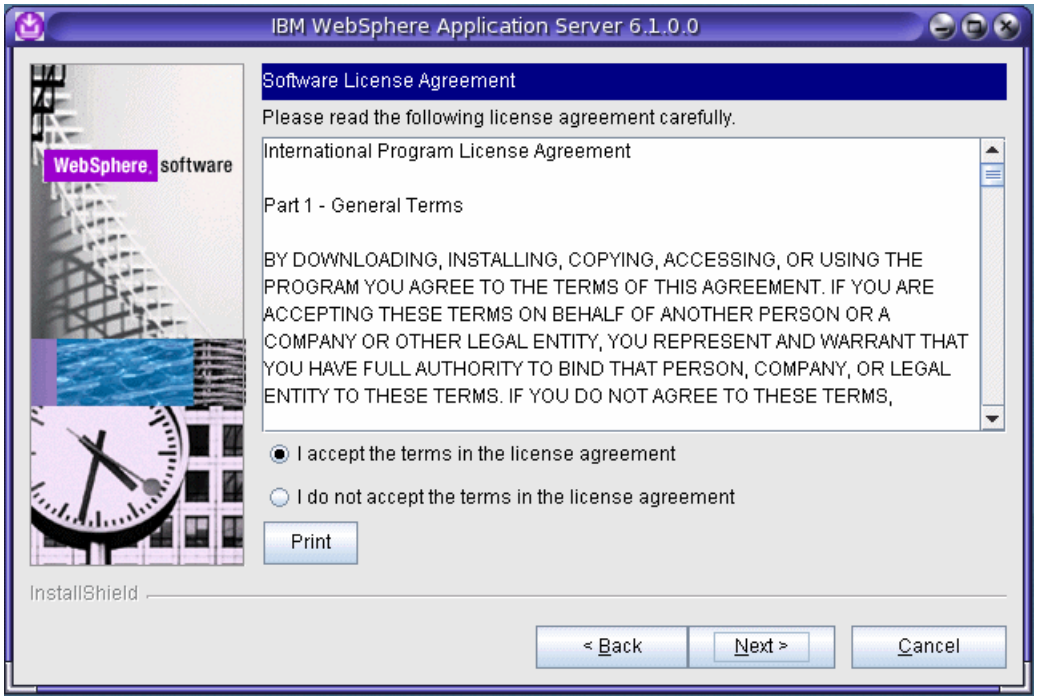


Figure 6-42 CIP installation wizard: License agreement window

5. After prerequisites are met, click **Next** (Figure 6-43).

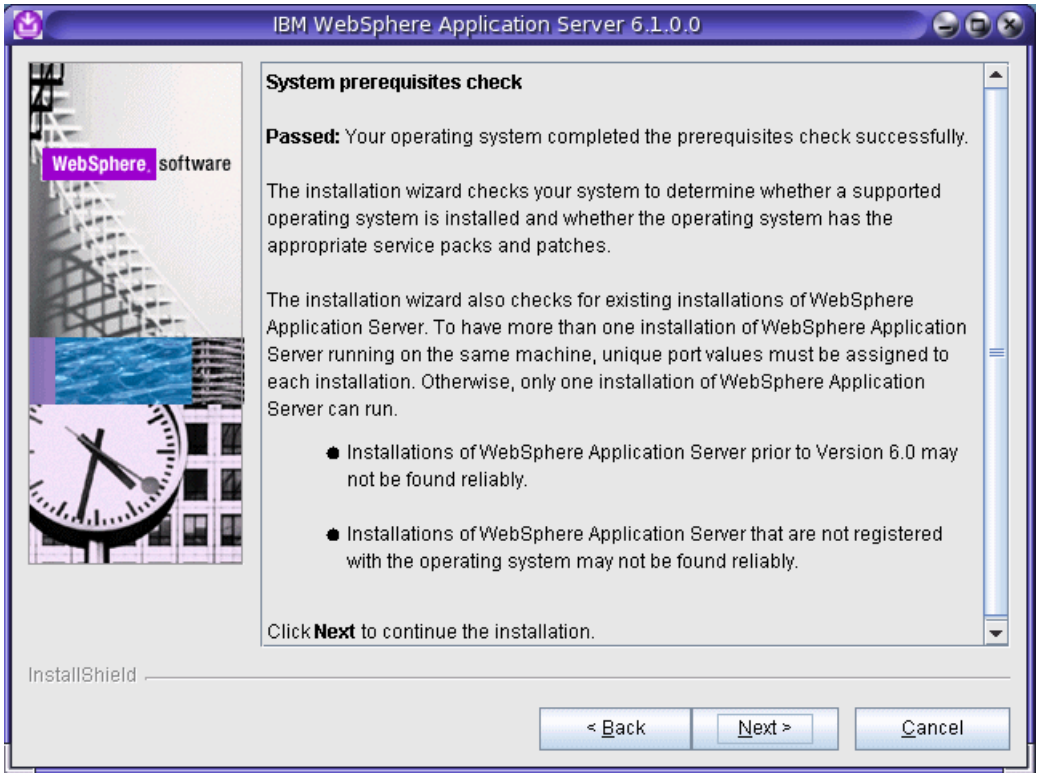


Figure 6-43 CIP installation wizard: System Prerequisites check

6. Select the **Install customization files contained in this installation** check box and click **Next**, as shown in Figure 6-44.

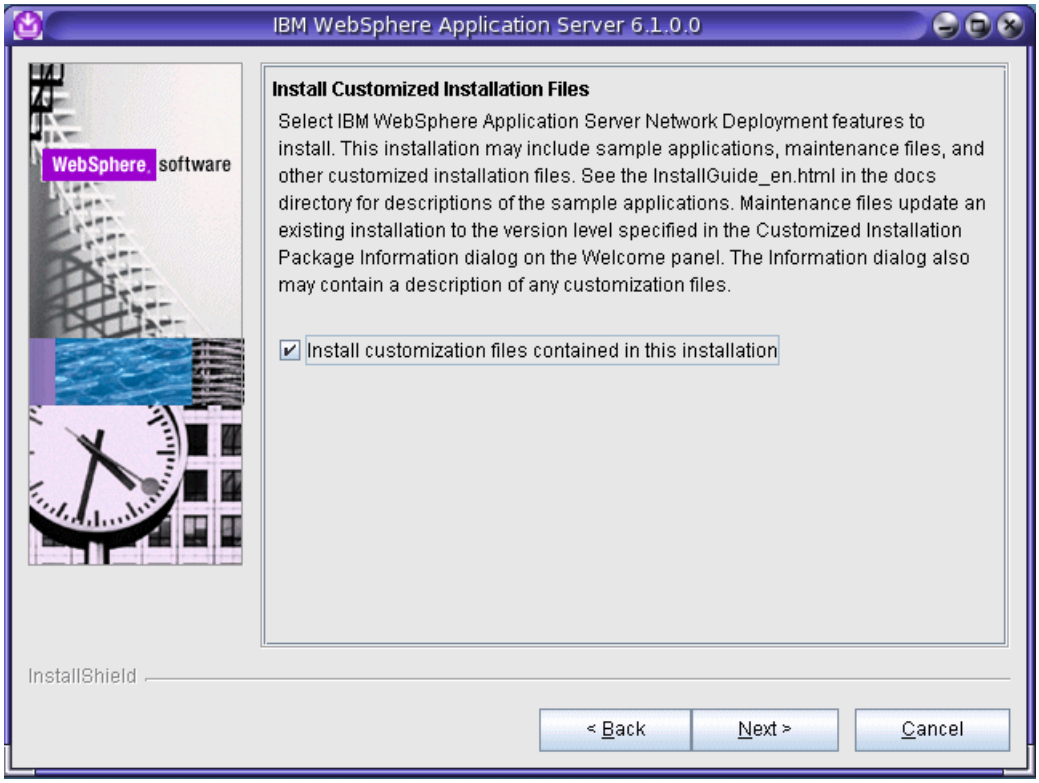


Figure 6-44 CIP installation wizard: Install customized files

7. Choose the installation directory for the installation and click **Next**. In this example, we choose the default installation directory, as shown in Figure 6-45 on page 217.

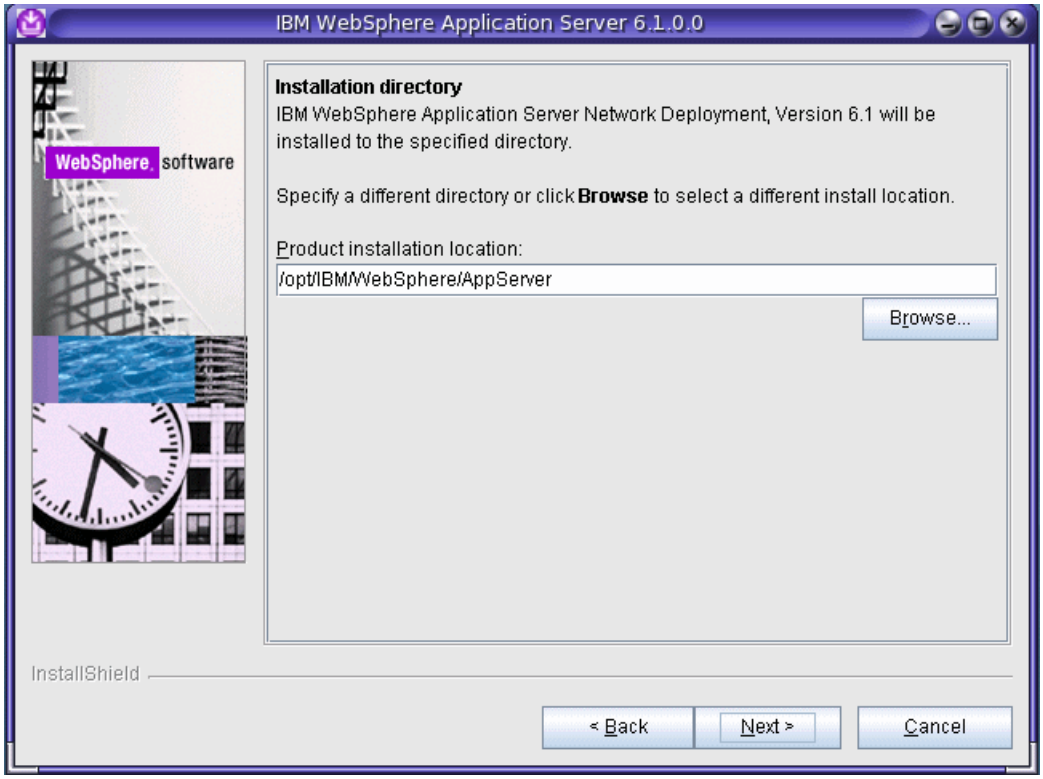


Figure 6-45 CIP installation wizard: Installation root definition

Attention:

- Leaving the destination of the installation root field empty prevents you from continuing.
- Do not use symbolic links as the destination directory because they are not supported.
- Spaces are not supported in the name of the destination installation directory in Solaris.

8. Choose **Application Server** from the Environments menu and click **Next**, as shown in Figure 6-46.

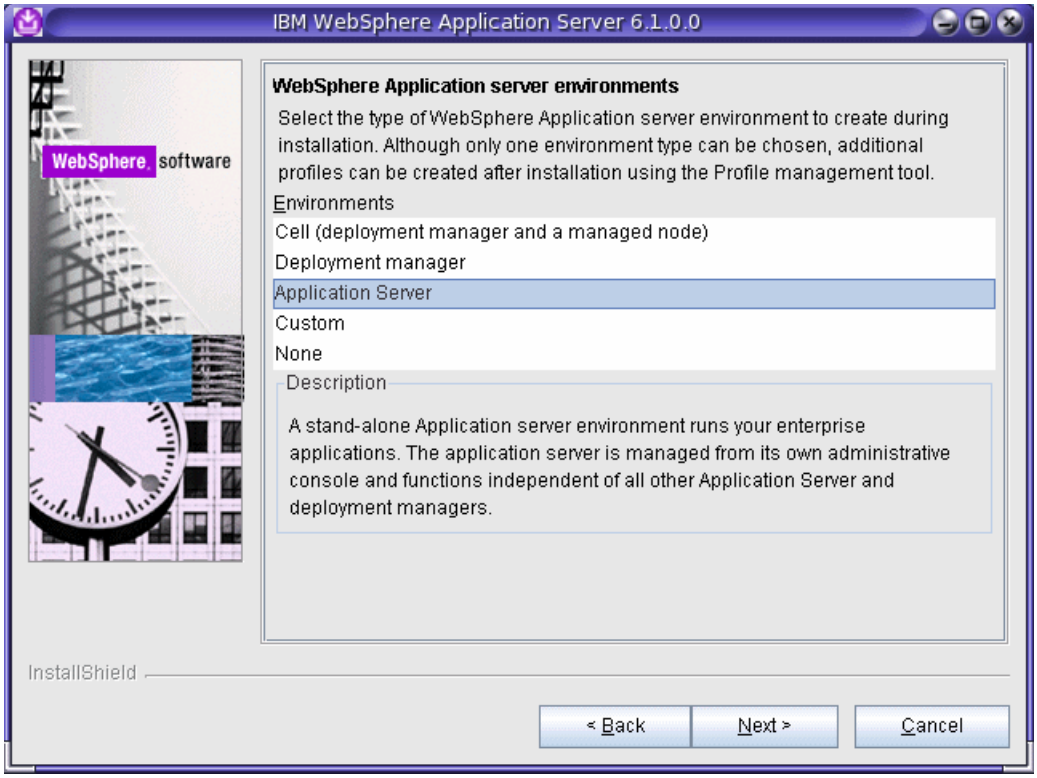


Figure 6-46 CIP installation wizard: Application Server environment selection

9. Optional: Enable administrative security, as shown in Figure 6-47 on page 219, and click **Next**.

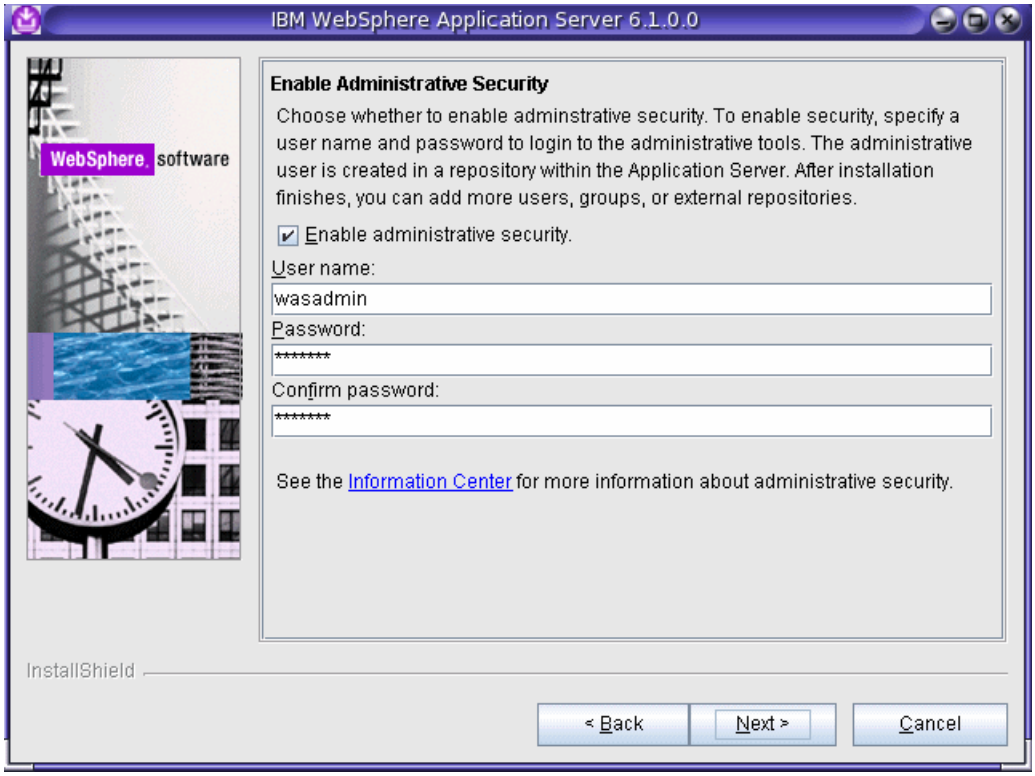


Figure 6-47 CIP installation wizard: Enable administrative security

Attention: If the source system WebSphere Application Server profile has security enabled, the profile's original security configuration will override the values given here.

10. Review the summary shown in Figure 6-48. If you have to change something, click **Back**; otherwise, click **Next**.

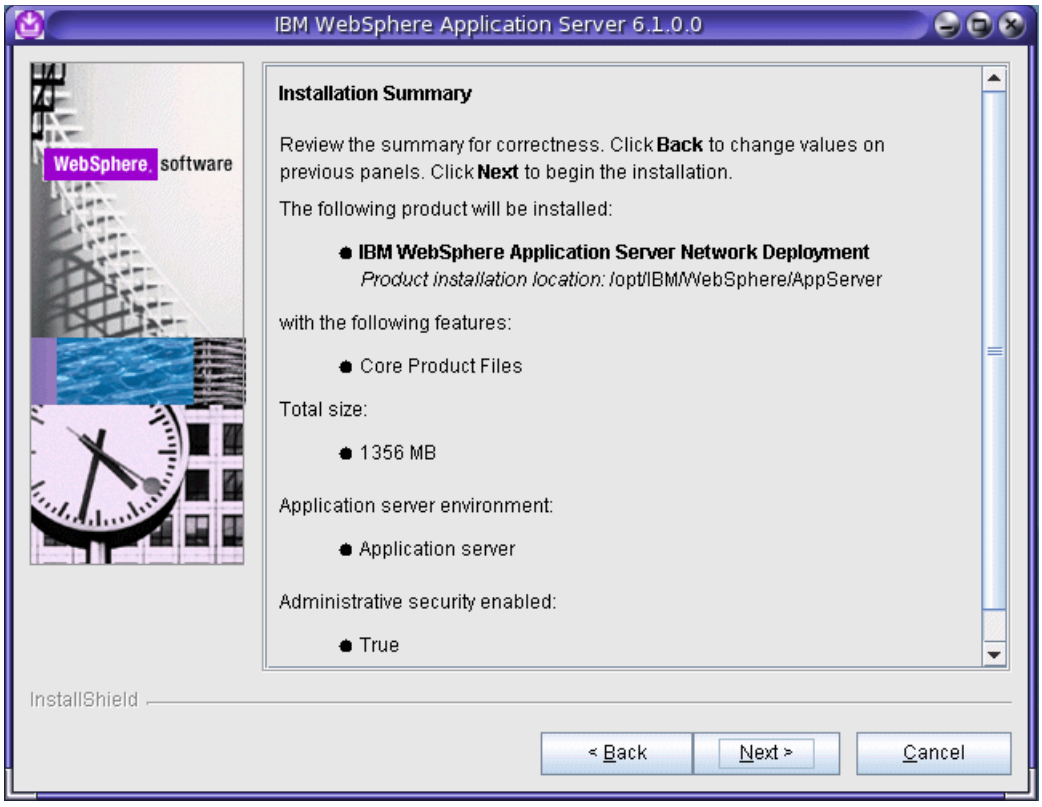


Figure 6-48 CIP installation wizard: Installation summary

The Installation wizard creates the uninstaller program and then displays the progress window that shows which components are being installed. This may take a little time. At the end of the installation, the wizard displays the Installation completion window.

Optional: You can verify the installation by launching the First steps console by clicking the **Launch the First steps console** check box before clicking the **Finish** button of completion window, as shown in Figure 6-49 on page 221.

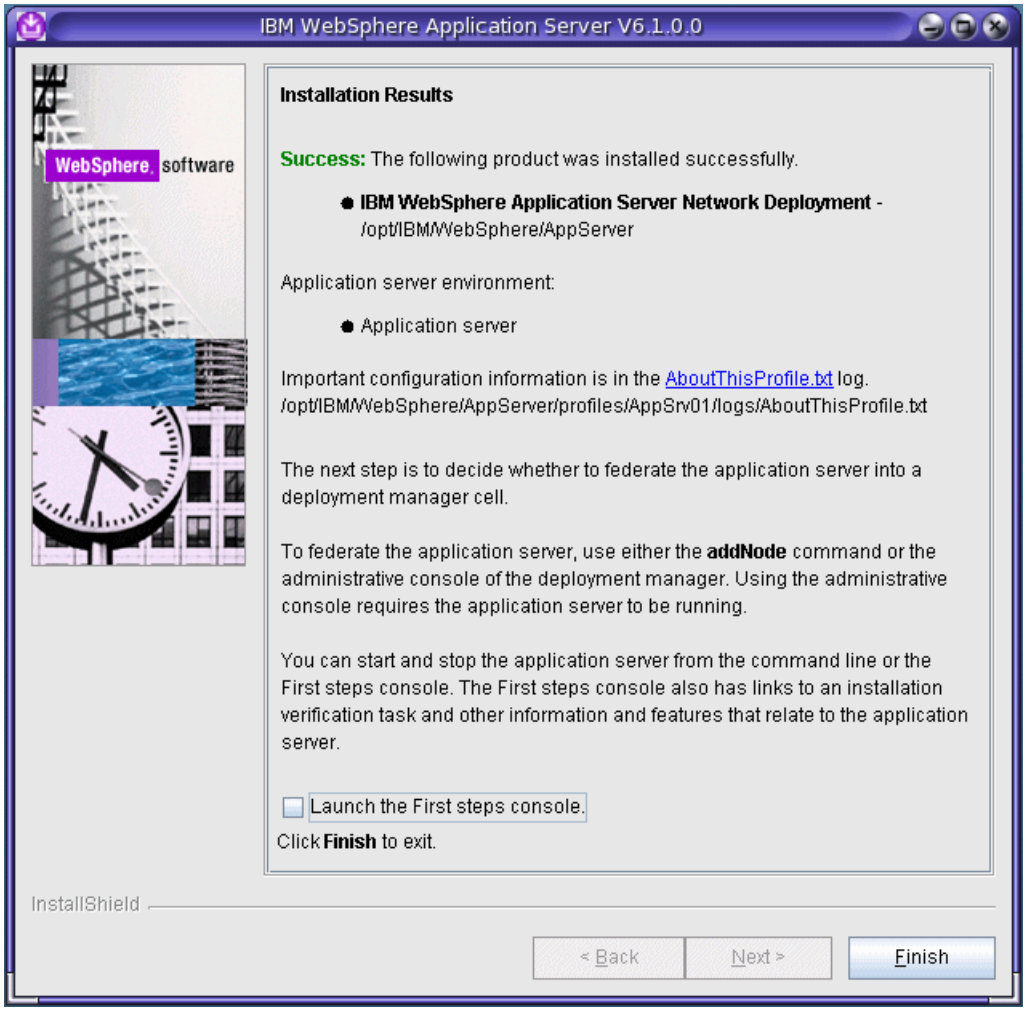


Figure 6-49 CIP installation wizard: Installation completed

Verifying profile installation

This example has an appserver profile with maintenance pack level 11 and one additional customized business application and one customized virtualhost configuration. In this example, we verify that the Fix Pack is installed with CIP, as shown in Example 6-13.

Example 6-13 Fix Pack installed with CIP

```
dest_host# cd /opt/IBM/WebSphere/AppServer/bin/
dest_host# ./versionInfo.sh
WVER0010I: Copyright (c) IBM Corporation 2002, 2005; All rights reserved.
WVER0012I: VersionInfo reporter version 1.15.1.14, dated 11/17/06

--------------------------------------------------------------------------------
IBM WebSphere Application Server Product Installation Status Report
--------------------------------------------------------------------------------

Report at date and time December 4, 2007 2:38:25 PM PST

Installation
--------------------------------------------------------------------------------
Product Directory        /opt/IBM/WebSphere/AppServer
```

```
Version Directory        /opt/IBM/WebSphere/AppServer/properties/version
DTD Directory            /opt/IBM/WebSphere/AppServer/properties/version/dtd
Log Directory            /opt/IBM/WebSphere/AppServer/logs
Backup Directory
/opt/IBM/WebSphere/AppServer/properties/version/nif/backup
TMP Directory            /var/tmp

Product List
--------------------------------------------------------------------------------
ND                       installed

Installed Product
--------------------------------------------------------------------------------
Name                     IBM WebSphere Application Server - ND
Version                  6.1.0.11
ID                       ND
Build Level              cf110734.37
Build Date               8/31/07

--------------------------------------------------------------------------------
End Installation Status Report
--------------------------------------------------------------------------------
```

As a result, Fix Pack 11 was installed with the product files. Next, we verify that the custom Virtualhost configuration is in place, as shown in Figure 6-50.

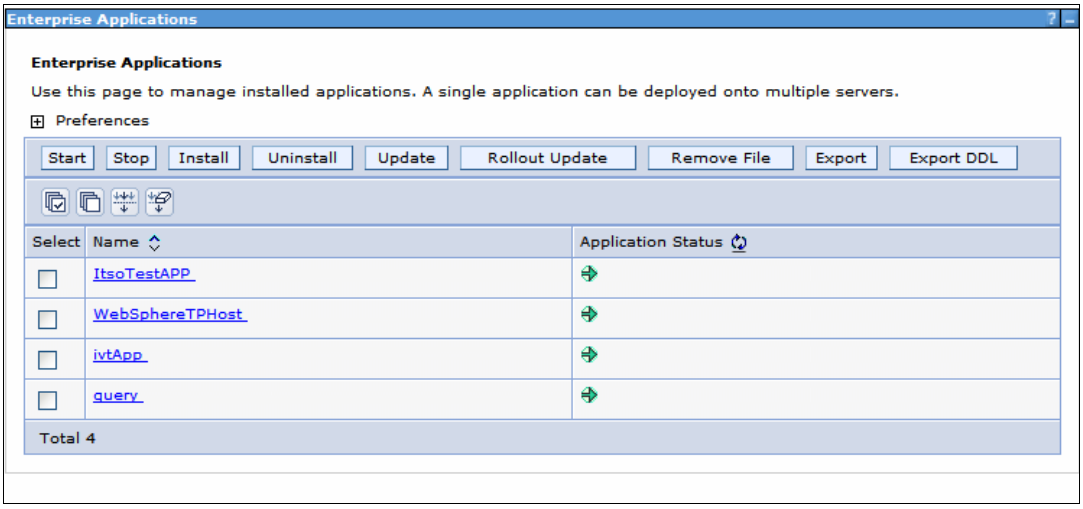


Figure 6-50 Profiles customized applications included in installation

Then we verify that the custom VirtualHost configuration in the new profile is set, as shown in Figure 6-51 on page 223.

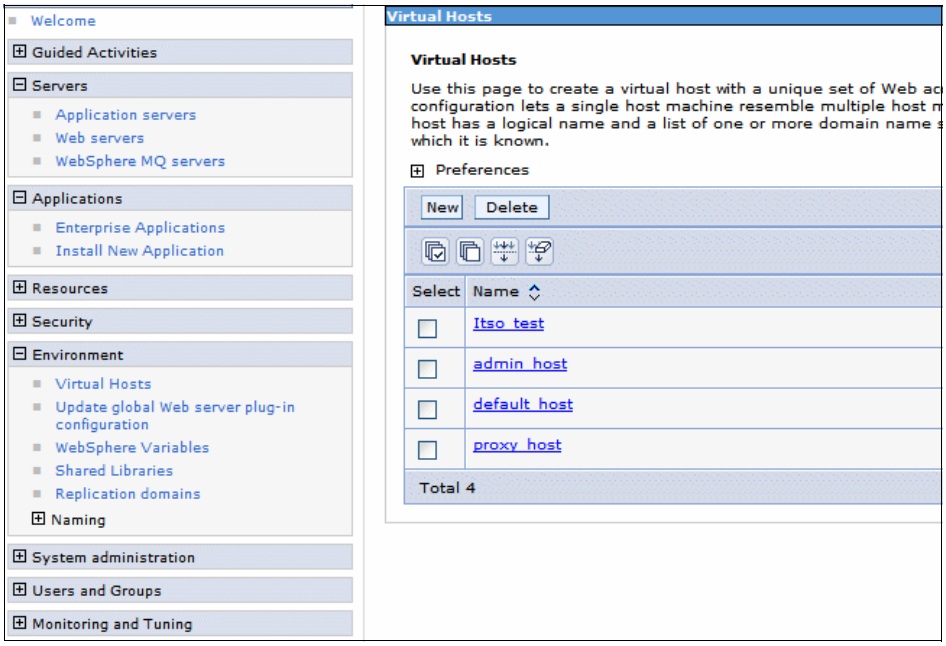


Figure 6-51 Custom VirtualHost configuration propagated in installation

6.4.2 Relocating the WebSphere Application Server environment with containers

To relocate your WebSphere Application Server environment installed in a Solaris Container, you must consider the following items to ensure that the WebSphere Application Server installation gets transferred correctly along with its configurations and profile information:

- If the source and destination systems have different hardware types and OS versions as well as patch levels, you may experience differences in devices, such as the network device changing from bge0 to e1000g0. In this case, you must follow the proper steps to adjust the differences in zone configuration.

  We do not recommend moving a container between two systems that have different OS version and patch level differences, although you can force it to happen. This will create many inconsistencies in your operating environment leading to undesirable results.

- If you have containers that rely on various system resource pools, these must be predefined on the destination system. A good practice is to define the resource control information within the zone configuration.

- If you keep the same host identity between the source and destination systems, the relocation is straight forward. If not, we recommend that you use CIP, as discussed in 6.4.1, “Relocating WebSphere Application Server environment using CIP” on page 196, because you will need to update the WebSphere Application Server server configuration with a new host identity. This will also require changing the sub-directory names within the WebSphere Application Server server config directory that have the host name prepended to the names. Manually modifying such a configuration can be prone to errors and time consuming.

In general, you can consider two different cases for relocating a WebSphere Application Server environment. One case is where WebSphere Application Server is installed independently in each local zone, as shown in Figure 6-52. The other case is where WebSphere Application Server installation is shared by multiple containers from a Global Zone location, as shown in Figure 6-53.

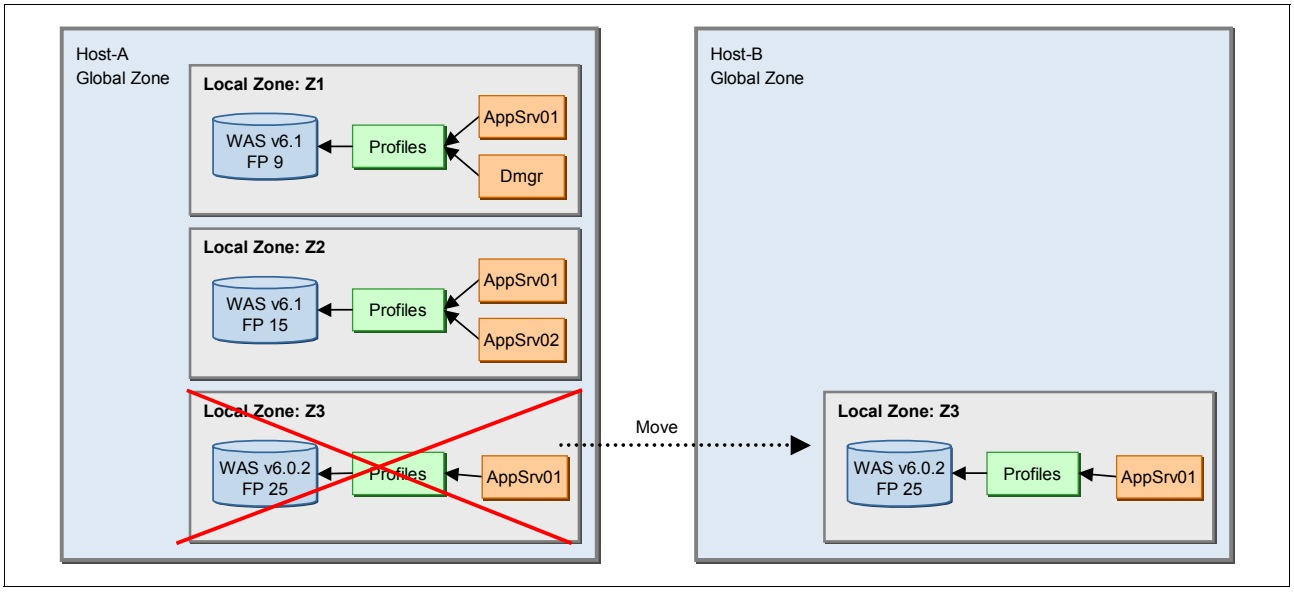


Figure 6-52 Migrating a Solaris Container from Host-A to Host-B where WebSphere Application Server is installed within the zone

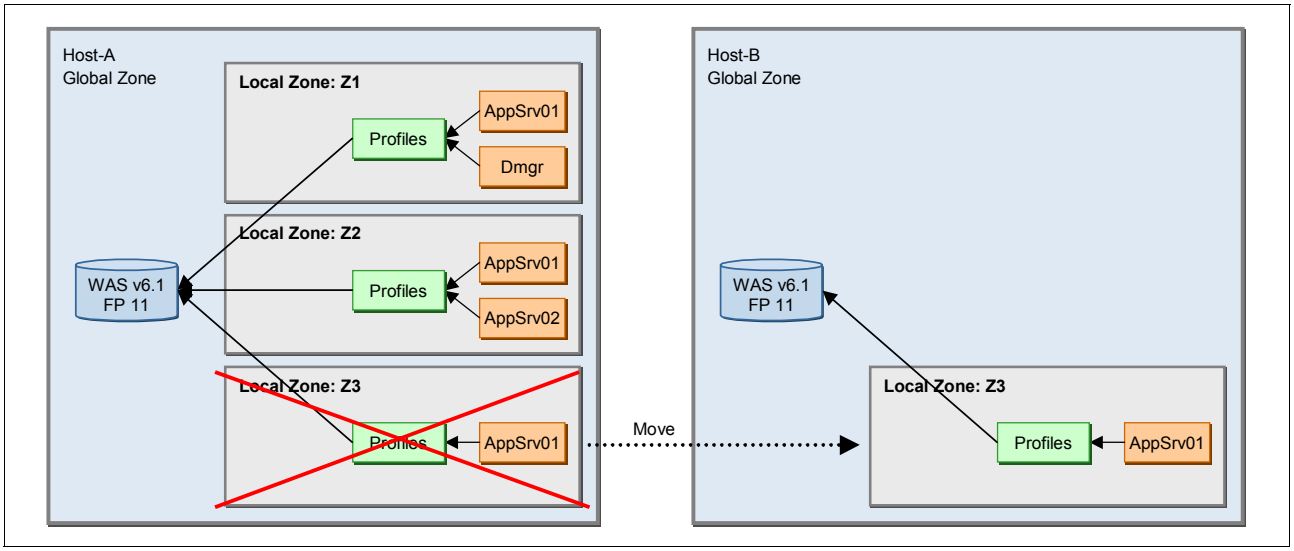


Figure 6-53 Migrating a Solaris Container from Host-A to Host-B where the WebSphere Application Server installation is shared

In the shared WebSphere Application Server installation case, you need to consider additional items, because the profile directory in each container is located in a custom defined location, as discussed in 5.3.4, “Scenario 4: Share the WebSphere Application Server installation with zones from the Global Zone” on page 137.

You must first prepare the destination system's Global Zone to have the WebSphere Application Server binary installation along with the necessary Fix Pack level as the source system before you can migrate a zone. You also must make sure that the wasprofile.properties is configured to point to the proper custom path to the profile directory just as you have it configured in the source system. Detailed steps are defined in 5.3.4, "Scenario 4: Share the WebSphere Application Server installation with zones from the Global Zone" on page 137.

Consider another scenario where you copy the WebSphere Application Server installation directories from the source system to the destination system:

- If WebSphere Application Server binaries are installed as the root user on the source system, you will not have the package information entry in the Solaris package registry on the destination system, as shown in Example 2-7 on page 26.
- If WebSphere Application Server binaries are installed as non-root privileges on the source system, you can copy the binaries as is to the destination system.
- If you wish to copy the WebSphere Application Server binaries installed from the source to the destination system, we recommend using CIP, as discussed in 6.4.1, "Relocating WebSphere Application Server environment using CIP" on page 196.

Attention: If WebSphere Application Server installation has already created and configured profiles:

- You cannot move a container to another host without re-creating the profiles.
- If the container is moved within the same host, you can move the container without additional configuration in WebSphere Application Server.

In every case of moving the WebSphere Application Server installation, we recommend that you always create WebSphere Application Server profiles *after* the container is moved, not before it. If profiles are already created, we recommend using CIP to move the WebSphere Application Server installation from one system to another.

Moving between two similar types of hosts

In order to move a zone between two similar types of systems, which have the same hardware types, OS versions, and patch level, you can follow the procedures described here for two different types of WebSphere Application Server deployments in zones: independent installation within the zone and shared installation. Assume the scenario where WebSphere Application Server is already deployed in the zone and the file system is UNIX File System (UFS).

Independent WebSphere Application Server binary installation

The following example describes how you can move an independent WebSphere Application Server installation environment contained in a zone, as shown in Figure 6-52 on page 224:

1. Log in to the zone (for example, yourzone) to be relocated and make sure all WebSphere Application Server profiles in the zone have no users logged in. Stop all of the WebSphere Application Server server processes.
2. Log in to the source system where the zone is hosted. This zone has a WebSphere Application Server deployment that contains the binaries, profiles, and other WebSphere Application Server server directories and files.
3. Halt the zone:

```
source_host# zoneadm -z yourzone halt
```

4. Detach the zone from the source system and create an archive file of the zone with the commands shown in Example 6-14.

Example 6-14 Detaching and packing the zone on the source system for the transfer

```
source_host# zoneadm -z yourzone detach
source_host# cd /export/zones/yourzone
source_host# pax -w@f /tmp/yourzone.pax -p e *
```

Note: You can use any of the utilities to archive your zone directory structure. In this example, we use **pax**. Run **man pax** on Solaris to get more information.

5. Transfer the archive file yourzone.pax from the source system to the destination system.
6. Log in to the Global Zone of the destination system as root.
7. Prepare the directory path for the new zone to be created on the destination system and unpack the zone archive file, as shown in Example 6-15.

Example 6-15 Unpacking zone package on the destination system

```
dest_host# mkdir -m 700 -p /export/zones/yourzone
dest_host# cd /export/zones/yourzone
dest_host# pax -r@f /path_to_pax/yourzone.pax -p e
```

8. On the destination system, you must first create a new zone. This is just a place holder for you to attach the zone files transferred from the source file to this newly created zone, as shown in Example 6-16.

Attention: Zone names are unique within a host while host names are unique within a network. If you wish to keep the zone name the same on the source and destination systems, you may do that. As we pointed out in the beginning of this section, if you keep the host identity the same when you move a zone, it is straight forward that you do not have to make any additional modifications to the WebSphere Application Server environment.

Example 6-16 Creating a new zone and attaching moved files to that zone

```
dest_host# zonecfg -z yourzone
zonecfg:yourzone> create -a /export/zones/yourzone
zonecfg:yourzone> exit
dest_host# zoneadm -z yourzone attach
```

9. Now you can boot and log in to this new zone, as shown in Example 6-17, and start the WebSphere Application Server process(es).

Example 6-17 Booting and login to migrated zone

```
dest_host# zoneadm -z yourzone boot
dest_host# zlogin -C yourzone
dest_host# cd /opt/IBM/WebSphere/AppServer/profiles/AppSrv01/bin
dest_host# startServer.sh server1
```

Shared WebSphere Application Server binary installation

For a shared WebSphere Application Server installation environment, you need to do a few extra steps to ensure the shared structure from the source system is preserved in the destination system's environment:

1. If the destination system does not have WebSphere Application Server installed, you must first lay out the WebSphere Application Server binaries in the Global Zone of the destination system. You must put the WebSphere Application Server installation in the exact location as the source system; otherwise, you must reconfigure the zone to inherit the correct location of the WebSphere Application Server installation. You do not need to create any profiles at this point. It is imperative that you put the same WebSphere Application Server version with same level of fix packs as well as the same directory structure as in the source system.

Tip: Instead of having to reinstall the WebSphere Application Server binary installation on the destination system, you would be better off using CIP, as described in 6.4.1, "Relocating WebSphere Application Server environment using CIP" on page 196, to preserve the WebSphere Application Server binary installation with no profiles from the source system. Then, use it to install on the destination system so that you can easily ensure that you receive the matching WebSphere Application Server version along with its Fix Pack level.

2. Edit the wasprofiles.properties, as described in "Global Zone preparations and actions" on page 137. It is imperative for you to put the same directory path for the profiles as in the source system; otherwise, you will have to put the correct location of the WebSphere Application Server profile directory in the wasprofiles.properties file.
3. Now you can follow the steps described in "Independent WebSphere Application Server binary installation" on page 225 to complete moving the zone from the source to destination systems.

Moving between two different types of hosts

In the previous section, we discussed moving methodologies between two similar types of a system. This may not be the case most of the time. Thus, we discuss the extra steps you need to take for you to move a zone from one physical host to another physical host where they may have two different architectures. We strongly recommend that the difference in architecture does not include the microprocessor (CPU) architecture. To reiterate, the assumption here is that your move is taking place between two different types of system, but they both use processors that belong to the same family, for example, UltraSPARC III to UltraSPARC T2. These processors are binary compatible; therefore, you do not have to obtain a different set of WebSphere Application Server binaries.

Important: You should keep the Solaris version and its patch level the same between the source and the destination systems for an easier and safer move. Otherwise, you will have to do lots of manual work to ensure your application environment will function properly. If you cannot resolve the discrepancies of the mismatch in the OS level or patches, you may experience server failure in your WebSphere Application Server.

The example provided here is merely a guideline and you must be aware that there can be many other differences between the systems involved. It is your responsibility to do your due diligence to ensure that you study all the differences and know how to accommodate those differences.

Tip: On a Solaris system, there are commands like **prtconf**, **prtdiag**, **ifconfig**, **uname**, and so on. You can use them to examine your hardware configuration. The Solaris OS release information is stored in /etc/release. You can determine the patch level with the **showrev -p** command.

The moving procedure is very similar to one discussed in "Independent WebSphere Application Server binary installation" on page 225, but you will need to do additional tasks:

1. Follow the procedure described in "Independent WebSphere Application Server binary installation" on page 225 until step 8.

   Because the zone is moved to a system with a different hardware configuration, we need to make sure that the zone is configured with the new device information to ensure that it boots on the destination system. One example we pick here is the network device in the next step.

2. Discover the name of the network device by using the **ifconfig** command:

```
dest_host# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232
index 1
        inet 127.0.0.1 netmask ff000000
e1000g0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.1.182 netmask ff000000 broadcast 9.255.255.255
        ether 0:14:4f:8e:5a:76
```

   Make a note about what network devices are available, such as e1000g0. You will need this information to configure your zone's network. If the network device you need to use is not up, you will need to "plumb" it properly, as discussed in "Configuring network interfaces" on page 25.

3. Create a new zone on the destination system, as shown in Example 6-18, and attach the zone files from the source system.

Example 6-18 Creating a zone and attaching files into it with -F flag

```
dest_host# zonecfg -z yourzone
zonecfg:yourzone> create -a /export/zones/yourzone
zonecfg:yourzone> exit
dest_host# zoneadm -z yourzone attach
```

4. If the network device is different on the destination system, you will have to reconfigure it, as shown in Example 6-19, so that the zone can boot successfully.

Important: Keeping the same IP address of the zone when you move it from the source to destination system will allow you to keep your existing profiles in the zone the way they are configured. If you change the IP address of the zone while moving, you must either reconfigure your application server profiles or create a new profile, as shown in 4.2.2, "Creating profiles with Profile Management Tool" on page 60, reconfigure the server, and redeploy your business applications.

Example 6-19 Overwriting source_hosts network configuration

```
dest_host# zonecfg -z yourzone
zonecfg:yourzone> remove net
zonecfg:yourzone> add net
zonecfg:yourzone:net> set physical=<dest_systems_network_device>
zonecfg:yourzone:net> set address=<ip_address_here>
```

```
zonecfg:yourzone:net> end
zonecfg:yourzone> exit
```

5. As shown in Example 6-17 on page 226, you can boot up and log in to the zone and start up your WebSphere Application Server server.

Tip: There are other possibilities to move containers more efficiently on Solaris, such as using ZFS. More information about this topic is available at:

http://sun.com/solaris/howtoguides/moving_containers.jsp

6.5 Backup and recovery

You can back up the WebSphere Application Server configuration to a zipped file by using the **backupConfig** command.

For a stand-alone node, run the **backupConfig** utility at the node level. For a network deployment cell, run the **backupConfig** utility at the deployment manager level, because it contains the master repository. This is the reason that you should not perform the **backupConfig** command at the node level of a cell.

The **restoreConfig** command restores the configuration of your stand-alone node or cell from the zipped file that was created using the **backupConfig** command.

The WebSphere Application Server commands **backupConfig** and **restoreConfig** only affect configurations of the WebSphere Application Server environment. We recommend using standard system backup tools to back up the remaining portion of the WebSphere Application Server installation or the whole environment.

6.5.1 Backing up a profile

Use the **backupConfig** command to back up a profile. The command will pack the configuration file in a zip file and store it in the current directory of a specified file name. To restore a configuration zip file, you can use the **restoreConfig** command. If you have an application server running when you issue the **backupConfig** command, it will stop all server processes in the configuration before performing the actual backup.

Here are a couple of rules for using **backupConfig**:

- Executing **backupConfig** from the WAS_ROOT/bin directory without the -profileName parameter will back up the default profile directory.
- Executing **backupConfig** from the PROFILE_ROOT/bin directory without the -profileName parameter will back up that profile.
- To back up a node configuration, specify the node profile in the -profileName parameter.
- To back up a cell configuration, specify the deployment manager profile in the -profileName parameter.
- To back up a stand-alone application server, specify the application server profile in -profileName parameter.

Example 6-20 shows an example of **backupConfig** command usage.

Example 6-20 Usage of the backupConfig command

```
#./backupConfig.sh Itso20071205.zip -profileName AppSrv01 -username wasadmin
-password <omitted>
ADMU0116I: Tool information is being logged in file
           /opt/IBM/WebSphere/AppServer/profiles/AppSrv01/logs/backupConfig.log
ADMU0128I: Starting tool with the AppSrv01 profile
ADMU5001I: Backing up config directory
           /opt/IBM/WebSphere/AppServer/profiles/AppSrv01/config to file
           /opt/IBM/WebSphere/AppServer/bin/Itso20071205.zip
ADMU0505I: Servers found in configuration:
ADMU0506I: Server name: server1
ADMU2010I: Stopping all server processes for node ilariNode01
ADMU0510I: Server server1 is now STOPPED
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................
ADMU5002I: 369 files successfully backed up
```

Parameters that can be used with the **backupConfig** command are shown in Table 6-2.

Table 6-2 Parameters for the backupConfig command

Parameter	Description
-nonstop	Servers are not to be stopped before backing up the configuration.
-quiet	Suppresses the printing of progress information.
-logfile <filename>	Name of the log file to which information get written. The default is <profile_home>/logs/backupConfig.log.
-profileName	Profile to run the command against. If the command is run from <was_home>/bin and -profileName is not specified, the default profile is used. If run from <profile_home>/bin, that profile is used.
-replacelog	Replaces the log file instead of appending it to the current log.
-trace	Generates trace information into the log file for debugging purposes.
-username <name>	User name for authentication if security is enabled in the server.
-password <password>	Specifies the password for authentication if security is enabled.
-help of -?	Prints command syntax information.

6.5.2 Restoring a profile

To restore a back up profile created previously with the **backupConfig** command, use the **restoreConfig** command. If the configuration you are restoring already exists, the existing directory is renamed config.old (then config.old_1, and so on) before the restore begins. The command restores the entire contents of the <profile_home>/config directory. By default, all servers on the node stop before the configuration is restored, so that a node synchronization does not occur during the restoration.

- Executing **restoreConfig** from the <was_home>/bin directory without the -profileName parameter will restore the default directory.
- Executing **restoreConfig** from the <profile_home>/bin directory without the -profileName parameter will restore that profile.

Example 6-21 shows an example of the usage of the **restoreConfig** command.

Example 6-21 Usage of the restoreConfig command

```
#./restoreConfig.sh Itso20071205.zip -profileName AppSrv01
ADMU0116I: Tool information is being logged in file
           /opt/IBM/WebSphere/AppServer/profiles/AppSrv01/logs/restoreConfig.log
ADMU0128I: Starting tool with the AppSrv01 profile
ADMU0505I: Servers found in configuration:
ADMU0506I: Server name: server1
ADMU2010I: Stopping all server processes for node ilariNode01
ADMU0512I: Server server1 cannot be reached. It appears to be stopped.
ADMU5502I: The directory /opt/IBM/WebSphere/AppServer/profiles/AppSrv01/config
           already exists; renaming to
           /opt/IBM/WebSphere/AppServer/profiles/AppSrv01/config.old
ADMU5504I: Restore location successfully renamed
ADMU5505I: Restoring file Itso20071205.zip to location
           /opt/IBM/WebSphere/AppServer/profiles/AppSrv01/config
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................
ADMU5506I: 369 files successfully restored
ADMU6001I: Begin App Preparation -
ADMU6009I: Processing complete.
```

Table 6-3 shows the parameters to use with the **restoreConfig** command.

Table 6-3 Parameters for the restoreConfig command

Parameter	Description
-nowait	Do not wait for the servers to be stopped before backing up the existing configuration.
-quiet	Suppresses the printing of progress information.
-location <directory_name>	Location of the backup file.
-logfile <fileName>	Location of the log file to which information gets written. The default is <profile_name>/logs/backupConfig.log.

Parameter	Description
-profileName <profile>	The profile to run the command against. If the command is run from <was_home>/bin and -profileName is not specified, the default profile is used. If run from <profile_home>/bin, that profile is used.
-replacelog	Replaces the log file instead of appending to the current log.
-trace	Generates trace information into the log file for debugging purposes.
-username <name>	User name for authentication if security is enabled in the server.
-password <password>	Specifies the password for authentication if security is enabled.
-help or -?	Prints the command syntax information.

6.5.3 Exporting and importing profiles

WebSphere Application Server V6.x provides a mechanism that allows you to export certain profiles or server objects to an archive. The archive can be distributed and imported to other installations.

An exported archive is a zip file of the config directory with host-specific information removed. The recommended extension to use in such files is .car. The exported archive can be complete configuration or a subset of configuration. Importing the archive creates the configurations defined in the archive.

The target configuration of an archive export or import can be a specific server or an entire profile. To use an archive, you would:

1. Export a WebSphere Application Server configuration. This creates a zip file with the configuration.
2. Unzip the files for browsing or update for use on other systems. For example, you might need to update resource references.
3. Send the configuration to the new system. An import can work with the zip file or with the expanded format.
4. Import the archive. The import process requires that you identify the object in the configuration you want to import and the target object in the existing configuration. The target can be the same object type as the archive or its parent:
   - If you import a server archive to a server configuration, the configurations are merged.
   - If you import a server archive to a node, the server is added to the node.

Server archives

The following command in **wsadmin** can be used to create an archive of a server:

```
$AdminTask exportServer { -archive <archive_location> -nodeName <node> -serverName
<server> }
```

This process removes applications from the server that you want to specify, and breaks the relationship between the server that you specify and the core group of the server, cluster, or bus membership. If you export a single server of a cluster, the relation to the cluster is eliminated.

To import a server archive, use the following command:

```
$AdminTask importServer {-archive <archive_location> [-nodeInArchive <node>]
[-serverInArchive <server>] [-nodeName <node>] [-serverName <server>]}
```

When you use the **importServer** command, you select a configuration object in the archive as the source and select a configuration object on the system as the target. The target object can match the source object of its parent. If the source and target are the same, the configurations are merged.

Profile archives

You can create a configuration archive (CAR) file containing the configuration of a stand-alone application server profile for later restoration. A CAR file can be used to clone the original profile to another machine or system. CAR files can be bundled in a customized installation package for use with the Installation Factory feature. For more information about using the Installation Factory, refer to the Information Center.

You can only create an archive of an unfederated profile (stand-alone application server profile). Refer to 6.4.1, “Relocating WebSphere Application Server environment using CIP” on page 196 for more information about using the CAR file for creating a Customized Installation Package with the Installation Factory.

The following command in **wsadmin** can be used to create an archive of a profile:

```
$AdminTask exportWasprofile { -archive <archive_location> }
```

7

Advanced topologies

This chapter discusses WebSphere Application Server advanced deployment topologies on Sun Solaris 10 OS. It augments the information in the Redbooks mentioned below and the IBM WebSphere Application Server ND V6.1 InfoCenter. To better understand these technologies, we suggest reading this chapter in conjunction with the following IBM Redbooks publications:

- *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392
- *WebSphere Application Server V6 System Management & Configuration Handbook*, SG24-6451
- *WebSphere Application Server Network Deployment V6: High Availability Solutions*, SG24-6688
- *Optimizing Operations with WebSphere Extended Deployment V6.1*, SG24-7422

7.1 Background

Large scale enterprise applications, such as store fronts, online banking, or trading, must provide a pleasant user experience with high reliability. They must also be designed to be resilient to unanticipated workload demands and be agile to adopt changes in the business climate. Deployment of these applications require many considerations to exploit the power of many distributed systems within the different tiers of the computing infrastructure. The advanced topologies concerned with the deployment architecture of these systems have the following properties:

- Scalability
- Workload management
- Availability
- Maintainability
- Session management
- Performance impacts of WebSphere Application Server security

Note that scaling the application server environment does not help if your application has an unscalable design. For Web application and EJB development best practices, refer to the white paper *WebSphere Application Server Development Best Practices for Performance and Scalability*, found at:

http://www.ibm.com/software/webservers/appserv/ws_bestpractices.pdf

7.1.1 Scalability

Scalability defines how easily a site will expand. Web sites must expand, sometimes with little warning, and grow to support an increased load. The increased load may come from many sources:

- New markets
- Normal growth
- Extreme peaks

An application and infrastructure that is architected for good scalability makes site growth possible and easy.

Most often, one achieves scalability by adding hardware resources to improve throughput. A more complex configuration, employing additional hardware, should allow one to service a higher client load than that provided by the simple basic configuration. Ideally, it should be possible to service any given load simply by adding additional servers or machines (or upgrading existing resources).

However, adding new systems or processing power does not always provide a linear increase in throughput. For example, doubling the number of processors in your system will not necessarily result in twice the processing capacity. Adding an additional horizontal server in the Application Server tier will not necessarily result in twice the request serving capacity. Adding additional resources has an additional impact on resource management and request distribution. While the impact and corresponding degradation may be small, you need to remember that adding *n* additional machines does not always result in *n* times the throughput.

Also, you should not simply add hardware without doing some investigation and possible software tuning first to identify potential bottlenecks in your application or any other performance-related software configurations. Adding more hardware may not necessarily improve the performance if the software is badly designed or not tuned correctly. Once the software optimization has been done, then the hardware resources should be considered the

the next step for improving performance. See 14.2, “Scalability”, of *IBM WebSphere V6 Planning and Design Handbook,* SG24-6446 for more information about this topic.

There are two different ways to improve performance when adding hardware: vertical scaling and horizontal scaling. See “Scalability” on page 246 for more information about these concepts.

Performance

Performance involves minimizing the response time for a given request or transaction.

The response time is the time it takes from the moment the user initiates a request at the browser to the moment the result of the HTML page returns to the browser. At one time, there was an unwritten rule of the Internet known as the “8 second rule.” This rule stated that any page that did not respond within eight seconds would be abandoned. Many enterprises still use this as the response time benchmark threshold for Web applications.

Throughput

Throughput, while related to performance, more precisely defines the number of concurrent transactions that can be accommodated.

For example, if an application can handle 10 customer requests simultaneously and each request takes one second to process, this site has a potential throughput of 10 requests per second.

7.1.2 Workload management

Workload management (WLM) is a WebSphere facility to provide load balancing and affinity between application servers in a WebSphere clustered environment. Workload management can be an important facet of performance. 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 the session state will be in the memory of this application server.

The proposed configuration should ensure that each machine 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 machine overloaded while another machine is mostly idle. If all machines have roughly the same capacity (for example, CPU power), each should process a roughly equal share of the load. Otherwise, there likely needs to be a provision for a workload to be distributed in proportion to the processing power available on each machine.

Furthermore, if the total load changes over time, the system should automatically adapt itself; for example, all machines may use 50% of their capacity, or all machines may use 100% of their capacity. But not one machine uses 100% of its capacity while the rest uses 15% of their capacity.

In this chapter, we discuss both Web server load balancing using WebSphere Edge Components and WebSphere workload management techniques.

7.1.3 Availability

Also known as resiliency, availability is the description of the system's ability to respond to requests no matter the circumstances. Availability requires that the topology provide some degree of process redundancy in order to eliminate single points of failure. While vertical scalability can provide this by creating multiple processes, the physical machine then becomes a single point of failure. For this reason, a high availability topology typically involves horizontal scaling across multiple machines.

Hardware-based high availability

Using a WebSphere Application Server multiple machine configuration eliminates a given application server process as a single point of failure. In WebSphere Application Server V5.0 and higher, the removal of the application dependencies on the administrative server process for security, naming, and transactions further reduces the potential that a single process failure can disrupt processing on a given node. In fact, the only single point of failure in a WebSphere cell is the Deployment Manager, where all central administration is performed. However, a failure at the Deployment Manager only impacts the ability to change the cell configuration and to run the Tivoli Performance Viewer, which is now included in the Administrative Console.

Failover

The proposition to have multiple servers (potentially on multiple independent machines) naturally leads to the potential for the system to provide failover. That is, if any one machine or server in the system were to fail for any reason, the system should continue to operate with the remaining servers. The load balancing property should ensure that the client load gets redistributed to the remaining servers, each of which will take on a proportionately slightly higher percentage of the total load. Of course, such an arrangement assumes that the system is designed with some degree of overcapacity, so that the remaining servers are indeed sufficient to process the total expected client load.

Ideally, the failover aspect should be totally transparent to clients of the system. When a server fails, any client that is currently interacting with that server should be automatically redirected to one of the remaining servers, without any interruption of service and without requiring any special action on the part of that client. In practice, however, most failover solutions may not be completely transparent. For example, a client that is currently in the middle of an operation when a server fails may receive an error from that operation, and may be required to retry (at which point the client would be connected to another, still available server). Or the client may observe a pause or delay in processing, before the processing of its requests resumes automatically with a different server. The important point in failover is that each client, and the set of clients as a whole, is able to eventually continue to take advantage of the system and receive service, even if some of the servers fail and become unavailable. Conversely, when a previously failed server is repaired and again becomes available, the system may transparently start using that server again to process a portion of the total client load.

The failover aspect is also sometimes called fault tolerance, in that it allows the system to survive a variety of failures or faults. It should be noted, however, that failover is only one technique in the much broader field of fault tolerance, and that no such technique can make a system 100 percent safe against every possible failure. The goal is to greatly minimize the *probability* of system failure, but keep in mind that the *possibility* of system failure cannot be completely eliminated.

Note that in the context of discussions on failover, the term *server* most often refers to a physical machine (which is typically the type of component that fails). However, we will see that WebSphere Application Server also allows for the possibility of one server process on a given machine to fail independently, while other processes on that same machine continue to operate normally.

HAManager

WebSphere Application Server V6 introduces a new concept for advanced failover and thus higher availability, called the High Availability Manager (HAManager). The HAManager enhances the availability of WebSphere Application Server singleton services like transaction services or JMS message services. It runs as a service within each application server process that monitors the health of WebSphere Application Server clusters. In the event of a server failure, the HAManager will fail over the singleton service and recover any in-flight transactions. Refer to Chapter 9, “WebSphere HAManager”, of *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392 for details.

7.1.4 Maintainability

Maintainability is the ability to keep the system running before, during, and after scheduled maintenance. When considering maintainability in performance and scalability, remember that maintenance periodically needs to be performed on hardware components, operating systems, and software products in addition to the application components.

While maintainability is somewhat related to availability, there are specific issues that need to be considered when deploying a topology that is maintainable. In fact, some maintainability factors are at cross purposes to availability. For example, ease of maintainability would dictate that one should minimize the number of application server instances in order to facilitate online software upgrades. Taken to the extreme, this would result in a single application server instance, which of course would not provide a high availability solution. In many cases, it is also possible that a single application server instance would not provide the required throughput or performance.

Some of the maintainability aspects that we consider are:

- Dynamic changes to configuration
- Mixed configuration
- Fault isolation

Dynamic changes to configuration

In certain configurations, it may be possible to modify the configuration on the fly without interrupting the operation of the system and its service to clients. For example, it may be possible to add or remove servers to adjust to variations in the total client load. Or it may be possible to temporarily stop one server to change some operational or tuning parameters, then restart it and continue to serve client requests. Such characteristics, when possible, are highly desirable, since they enhance the overall manageability and flexibility of the system.

Mixed configuration

In some configurations, it may be possible to mix multiple versions of a server or application, so as to provide for staged deployment and a smooth upgrade of the overall system from one software or hardware version to another. Coupled with the ability to make dynamic changes to the configuration, this property may be used to effect upgrades without any interruption of service.

Fault isolation

In the simplest application of failover, we are only concerned with clean failures of an individual server, in which a server simply ceases to function completely, but this failure has no effect on the health of other servers. However, there are sometimes situations where one malfunctioning server may in turn create problems for the operation of other, otherwise healthy servers. For example, one malfunctioning server may hoard system resources, or database resources, or hold critical shared objects for extended periods of time, and prevent other servers from getting their fair share of access to these resources or objects. In this context, some configurations may provide a degree of fault isolation, in that they reduce the potential for the failure of one server to affect other servers.

7.1.5 Session state

Unless you have only a single application server or your application is completely stateless, maintaining state between HTTP client requests also plays a factor in determining your configuration. Use of the session information, however, is a fine line between convenience for the developer and performance and scalability of the system. It is not practical to eliminate session data altogether, but care should be taken 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 Application Server environment, you need to take session needs into account as early as possible.

In WebSphere Application Server V6, there are two methods for sharing of sessions between multiple application server processes (cluster members). One method is to persist the session to a database. An alternate approach is to use memory-to-memory session replication functionality, which was added to WebSphere Application Server V5 and is implemented using WebSphere internal messaging. The memory-to-memory replication (sometimes also referred to as “in-memory replication”) eliminates a single point of failure found in the session database (if the database itself has not been made highly available using clustering software).

Tip: Refer to *A Practical Guide to DB2 UDB Data Replication V8*, SG24-6828 for details about DB2 replication.

7.1.6 Performance impact of WebSphere Application Server security

The potential to distribute the processing responsibilities between multiple servers and, in particular, multiple machines, also introduces a number of opportunities for meeting special security constraints in the system. Different servers or types of servers may be assigned to manipulate different classes of data or perform different types of operations. The interactions between the various servers may be controlled, for example, through the use of firewalls, to prevent undesired accesses to data.

SSL

Building up SSL communication causes extra HTTP requests and responses between the machines and every SSL message is encrypted on one side and decrypted on the other side.

SSL at the front end or Web tier is a common implementation. This allows for secured purchases, secure viewing of bank records, and so on. SSL handshaking, however, is expensive from a performance perspective. The Web server is responsible for encrypting data sent to the user, and decrypting the request from the user. If a site will be operated using SSL more often than not and the server is operating at close to maximum CPU, then using some

form of SSL accelerator hardware in the server, or even an external device that offloads all SSL traffic prior to communication with the Web server, may help improve performance.

Performance related security settings

The following settings can help to fine-tune the security-related configurations to enhance performance:

- Security cache timeout

  This setting determines how long WebSphere Application Server should cache information related to permission and security credentials. When the cache timeout expires, all cached information becomes invalid. Subsequent requests for the information result in a database lookup. Sometimes, acquiring the information requires invoking an LDAP-bind or native authentication, both of which are relatively costly operations in terms of performance.

- HTTP session timeout

  This parameter specifies how long a session will be considered active when it is unused. After the timeout, the session expires and another session object will be created. With high-volume Web sites, this may influence the performance of the server.

- Registry and database performance

  Databases and registries used by an application influence the WebSphere Application Server performance. In a distributed environment, we highly recommend that you put the authorization server onto a separate machine in order to offload application processing. When planning for a secured environment, special attention should be given to the sizing and implementation of the directory server. It is a best practice to first tune the directory server and make sure that it performs well before looking at the WebSphere environment. Also, make sure that this directory does not introduce a new single point of failure.

Note: WebSphere Application Server security is out of the scope of this book. However, an entire book is dedicated to this topic: *WebSphere Application Server V6: Security Handbook,* SG24-6316.

7.2 Workload management

Workload management (WLM) is the process of spreading multiple requests for work over the resources that can do the work. It optimizes the distribution of processing tasks in the WebSphere Application Server environment. Incoming work requests are distributed to the application servers and other objects 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.

Workload management is most effective when the deployment topology is comprised of application servers on multiple machines, since 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.

Workload management provides the following benefits to WebSphere applications:

- It balances client processing requests, allowing incoming work requests to be distributed according to a configured WLM selection policy.
- It provides failover capability by redirecting client requests to a running server when one or more servers are unavailable. This improves the availability of applications and administrative services.
- It enables systems to be scaled up to serve a higher client load than provided by the basic configuration. With clusters and cluster members, additional instances of servers can easily be added to the configuration. See 7.2.3, “Workload management using WebSphere clustering” on page 243 for details.
- It enables servers to be transparently maintained and upgraded while applications remain available for users.
- It centralizes the administration of application servers and other objects.

Two types of requests can be workload-managed in IBM WebSphere Application Server Network Deployment V6:

- *HTTP requests* can be distributed across multiple Web containers, as described in 7.2.2, “Workload management with Web server plug-in” on page 243. We refer to this as *Plug-in WLM*.
- *EJB requests* can be distributed across multiple EJB containers, as described in 7.2.4, “Enterprise Java Services workload management” on page 248. We refer to this as *EJS WLM*.

7.2.1 Workload management with Web servers and load balancers

If your environment uses multiple Web servers, a mechanism is needed to allow servers to share the load. The HTTP traffic must be spread among a group of servers. These servers must appear as one server to the Web client (browser), making up a *cluster*, which is a group of independent nodes interconnected and working together as a single system. This cluster concept should not be confused with a WebSphere cluster, as described in 7.2.3, “Workload management using WebSphere clustering” on page 243.

As shown in Figure 7-1, a load balancing mechanism called *IP spraying* can be used to intercept the HTTP requests and redirect them to the appropriate machine on the cluster, providing scalability, load balancing, and failover.

See Chapter 4, “Introduction to WebSphere Edge Components”, of *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392 for more details.

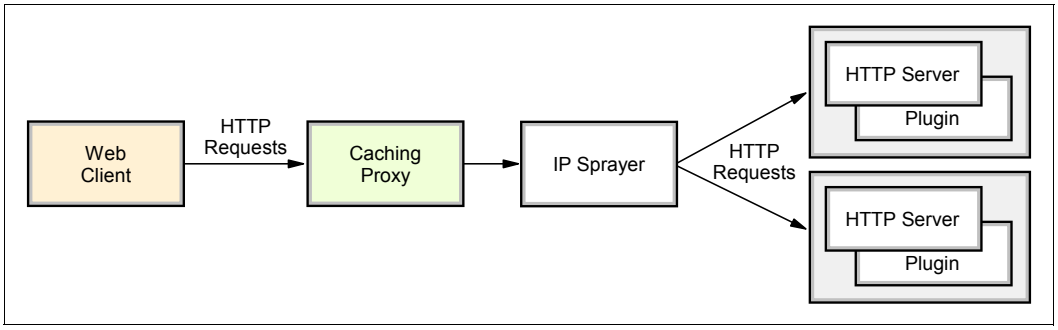


Figure 7-1 Web server workload management

7.2.2 Workload management with Web server plug-in

This is a short introduction to plug-in workload management, where servlet requests are distributed to the Web container in clustered application servers, as shown in Figure 7-2. This configuration is also referred to as the *servlet clustering architecture*.

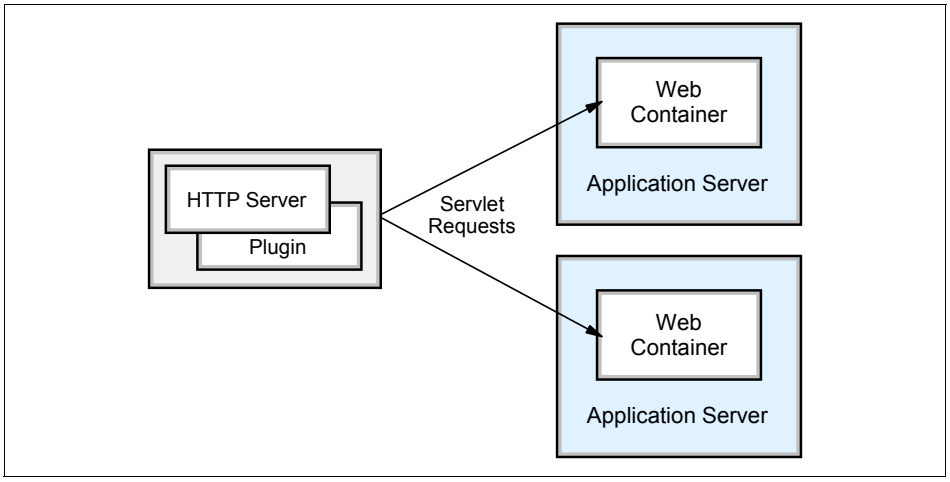


Figure 7-2 Plug-in (Web container) workload management

Clustering application servers that host Web containers automatically enable plug-in workload management for the application servers and the servlets they host. In the simplest case, the cluster is configured on a single machine, where the Web server process also runs.

Routing of servlet requests occurs between the Web server plug-in and the clustered application servers using HTTP or HTTPS. This routing is based purely on the weights associated with the cluster members. If all cluster members have identical weights, the plug-in sends an equal number of requests to all members of the cluster, when assuming no session affinity. If the weights are scaled in the range from zero to 20, the plug-in routes requests to those cluster members with the higher weight value more often. A rule of thumb formula for determining routing preference would be:

% routed to Server1 = weight1 / (weight1+weight2+...+weight*n*)

where *n* is the number of cluster members in the cluster. The Web server plug-in distributes requests around cluster members that are not available.

See Chapter 6, “Plug-in workload management and failover”, of *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392 for more details.

7.2.3 Workload management using WebSphere clustering

This section describes how workload management is implemented in IBM WebSphere Application Server Network Deployment V6 by using application server clusters and cluster members.

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 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, 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, that is, 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. Depending on the capacity of your systems, you can define different weights for the various cluster members.

It may be a good idea to create the cluster with a single member first, adjust the member's configuration, and then add the other members. This process guarantees that all cluster members are created with the same settings. 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 though 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 automatically starts or stops all cluster members, and changes to the application are propagated to all application servers in the cluster.

Figure 7-3 on page 245 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.

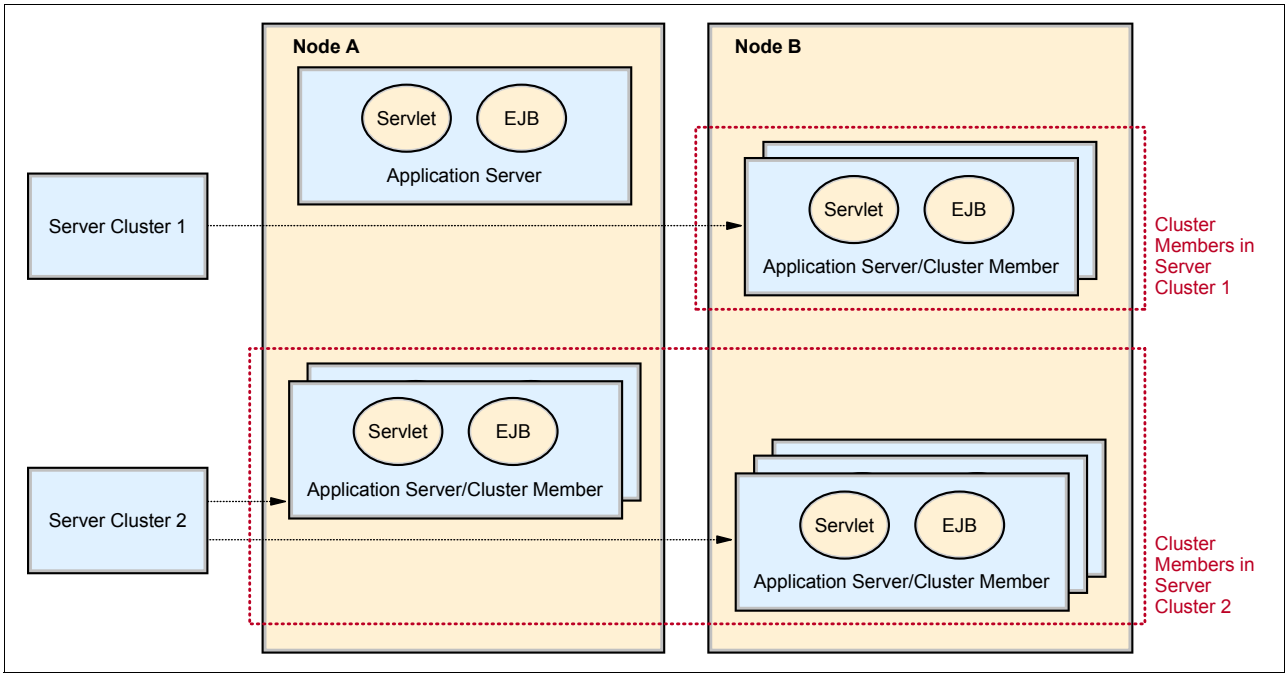


Figure 7-3 Server clusters and cluster members

Clusters and cluster members provide the necessary support for workload management, failover, and scalability. For details, refer to Chapter 6, “Plug-in workload management and failover”, of *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392.

Distributing workloads

The ability to route a request to any server in a group of clustered application servers allows the servers to share work and improving 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. Using weighted definitions of cluster members allows nodes 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.

Failover

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.

Scalability

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

- In *vertical scaling*, shown in Figure 7-4, multiple cluster members for an application server are defined on the same physical machine, or node, which may allow the machine's processing power to be more efficiently allocated.

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

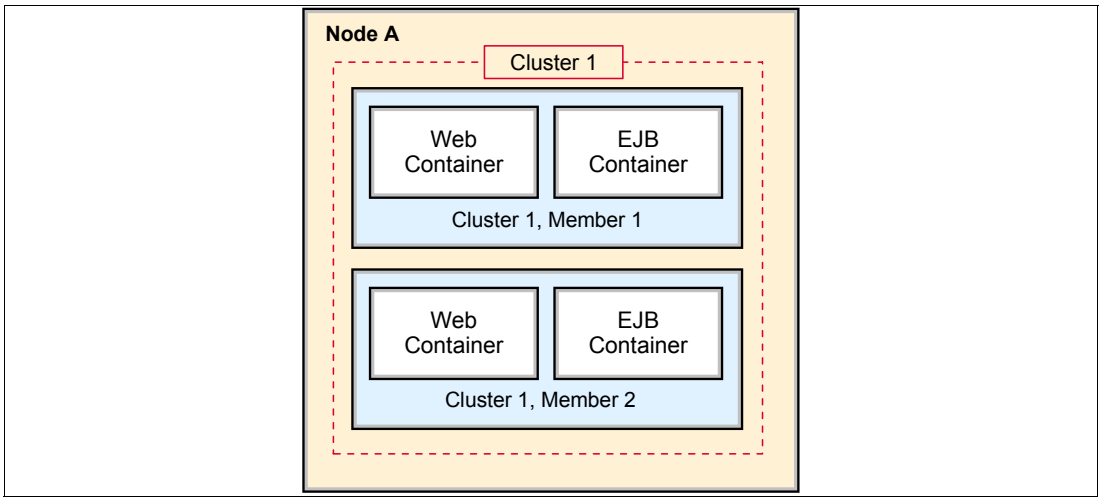


Figure 7-4 Vertical scaling

  We recommend that you avoid using "rules of thumb" when determining the number of cluster members needed for a given machine. The only way to determine what is correct for your environment and application(s) is to tune a single instance of an application server for throughput and performance, then add it to a cluster, and incrementally add additional cluster members. 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.

- In *horizontal scaling*, shown in Figure 7-5 on page 247, cluster members are created on multiple physical machines. 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 smaller, less powerful machines. Client requests that overwhelm a single machine can be distributed over several machines in the system. Failover is another benefit of horizontal scaling. If a machine becomes unavailable, its workload can be routed to other machines containing cluster members.

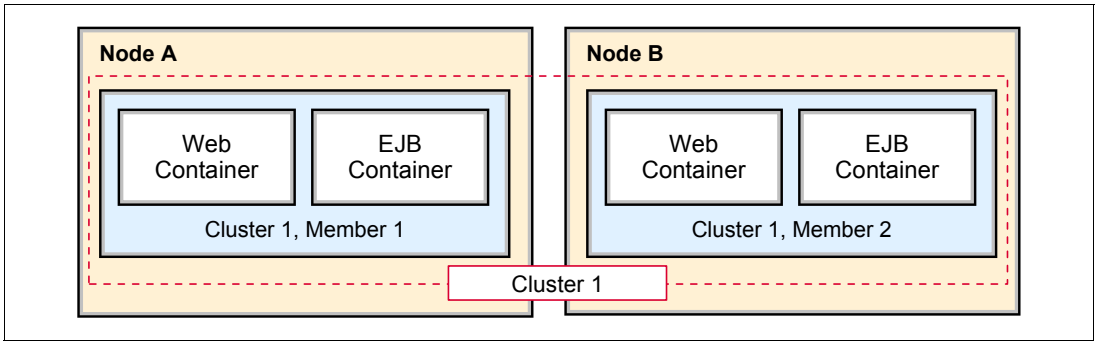


Figure 7-5 Horizontal scaling

Horizontal scaling can handle application server process failures and hardware failures (or maintenance) without significant interruption to client service.

Note: WebSphere Application Server V5.0 and higher supports horizontal clustering across different platforms and operating systems. Horizontal cloning on different platforms was not supported in WebSphere Application Server V4.

- WebSphere Application Server applications can combine horizontal and vertical scaling to reap the benefits of both scaling techniques, as shown in Figure 7-6.

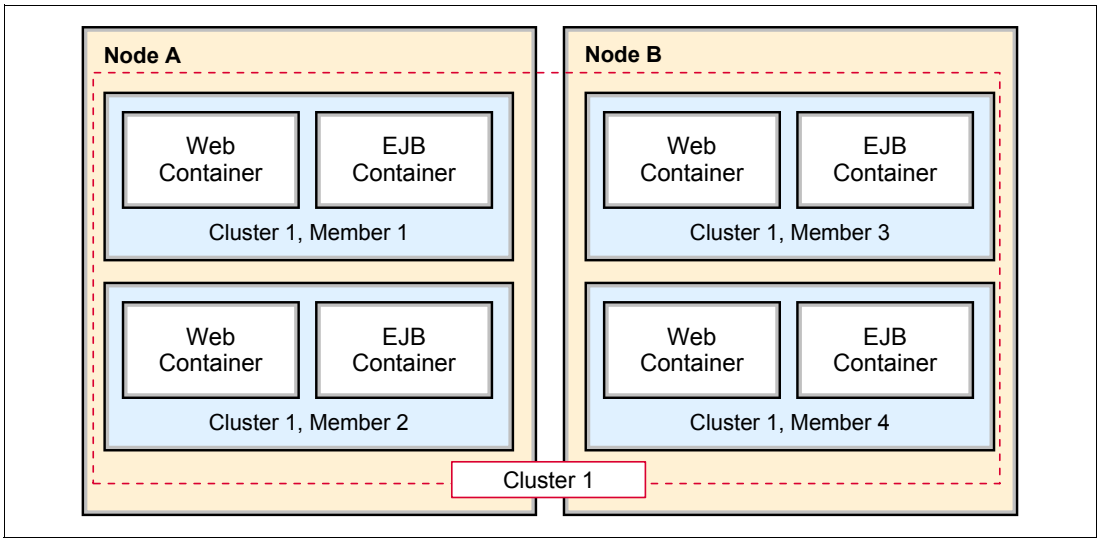


Figure 7-6 Vertical and horizontal scaling

Secure application cluster members

The workload management service has its own built-in security, which works with the WebSphere Application Server security service to protect cluster member resources. If security is needed for your production environment, enable security before you create a cluster for the application server. This enables security for all of the members in that cluster.

The EJB method permissions, Web resource security constraints, and security roles defined in an enterprise application are used to protect EJBs and servlets in the application server cluster. Refer to *WebSphere Application Server V6: Security Handbook,* SG24-6316 for more information.

7.2.4 Enterprise Java Services workload management

In this section, we discuss Enterprise Java Services workload management (EJS WLM), in which the Web container and EJB container are on different application servers. The application server hosting the EJB container is clustered.

Configuring the Web container in a separate application server from the Enterprise JavaBean container (an EJB container handles requests for both session and entity beans) enables distribution of EJB requests between the EJB container clusters, as seen in Figure 7-7.

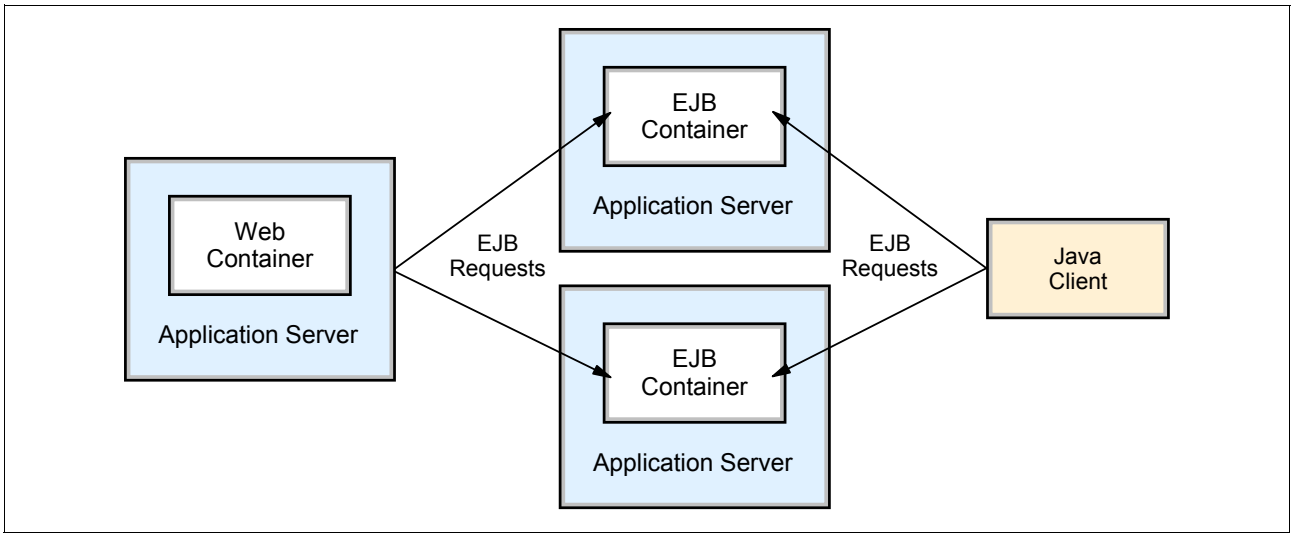


Figure 7-7 EJB workload management

In this configuration, EJB client requests are routed to available EJB containers based on the workload management EJB selection policy (Server-weighted round robin routing or Prefer local).

Important: Although it is possible to split the Web container and EJB container, it is not recommended because of the negative performance impact. In addition, application maintenance also becomes more complex when the application runs in different application servers.

The EJB clients can be servlets operating within a Web container, stand-alone Java programs using RMI/IIOP, or other EJBs.

EJS workload management is covered in detail in Chapter 7, “EJB workload management”, of *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392.

7.3 Topology selection criteria

While a variety of factors come into play when considering the appropriate topology for a WebSphere deployment (see 7.1, “Background” on page 236), the primary factors to plan for typically include:

- Security
- Performance
- Throughput
- Scalability

- Availability
- Maintainability
- Session state

For detailed information about topologies, their advantages and disadvantages, required software, as well as topology selection criteria, refer to the *IBM WebSphere V6 Planning and Design Handbook,* SG24-6446.

Note: For each of the Network Deployment topologies, a decision needs to be made regarding the placement of the Deployment Manager and master cell repository. The Deployment Manager can be located either on a dedicated machine, or on the same machine as one of its nodes. It is, however, considered a best practice to place the Deployment Manager on a separate machine. For more information about possible configurations, refer to 9.3, “Cell topologies”, in *IBM WebSphere V6 Planning and Design Handbook,* SG24-6446.

7.4 Strategies for scalability and availability

On demand computing requires the ability to scale up or scale down an application, depending on the current requirements. Thus, scalability is important to improve efficiency and reduce cost.

We start by discussing scalability strategies using WebSphere Application Server that can help us in ensuring high availability, load balancing, and removing bottlenecks.

We can exploit a strategy, distributing the load among the most appropriate resources, and using workload management techniques such as vertical and horizontal scaling, as described in 7.2.3, “Workload management using WebSphere clustering” on page 243. WebSphere Application Servers can benefit from vertical and horizontal scaling and the HTTP servers can be horizontally scaled on a clustered configuration. The use of these techniques is represented in Figure 7-9 on page 251.

We describe various topologies where different techniques are applied on a set of distributed machines, in order to provide a reliable and efficient processing environment.

Session persistence considerations

If the application maintains the state between HTTP requests and we are using vertical or horizontal scaling, then we 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 logging out users, we need to share the session data between multiple processes. There are two ways of doing 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 (and properly configured) application server starts.

- Database persistence

  Session data is stored in a database shared by all application servers.

Memory-to-memory replication has the following advantages and disadvantages compared to database persistence:

- No separate database product is required.
- Enabling and configuring replication is very easy using the Administrative Console.
- The impact of replicating session information might be significant depending on the number of application servers in your replication domain, the size of the sessions, and the configuration of the replication domain (Single replica, Entire Domain, and Number of replicas). Therefore, care must be taken to configure the replication domain correctly and database persistence might be the better choice in a memory constraint environment.

Refer to Chapter 12, “Session management”, especially 12.9, “Persistent session management”, of *WebSphere Application Server V6 System Management and Configuration Handbook,* SG24-6451 for more detailed information.

7.5 Topology for vertical scaling

Vertical scaling refers to configuring multiple application servers on a single machine and creating a cluster of associated application servers all hosting the same J2EE application(s). This configuration is depicted in Figure 7-8.

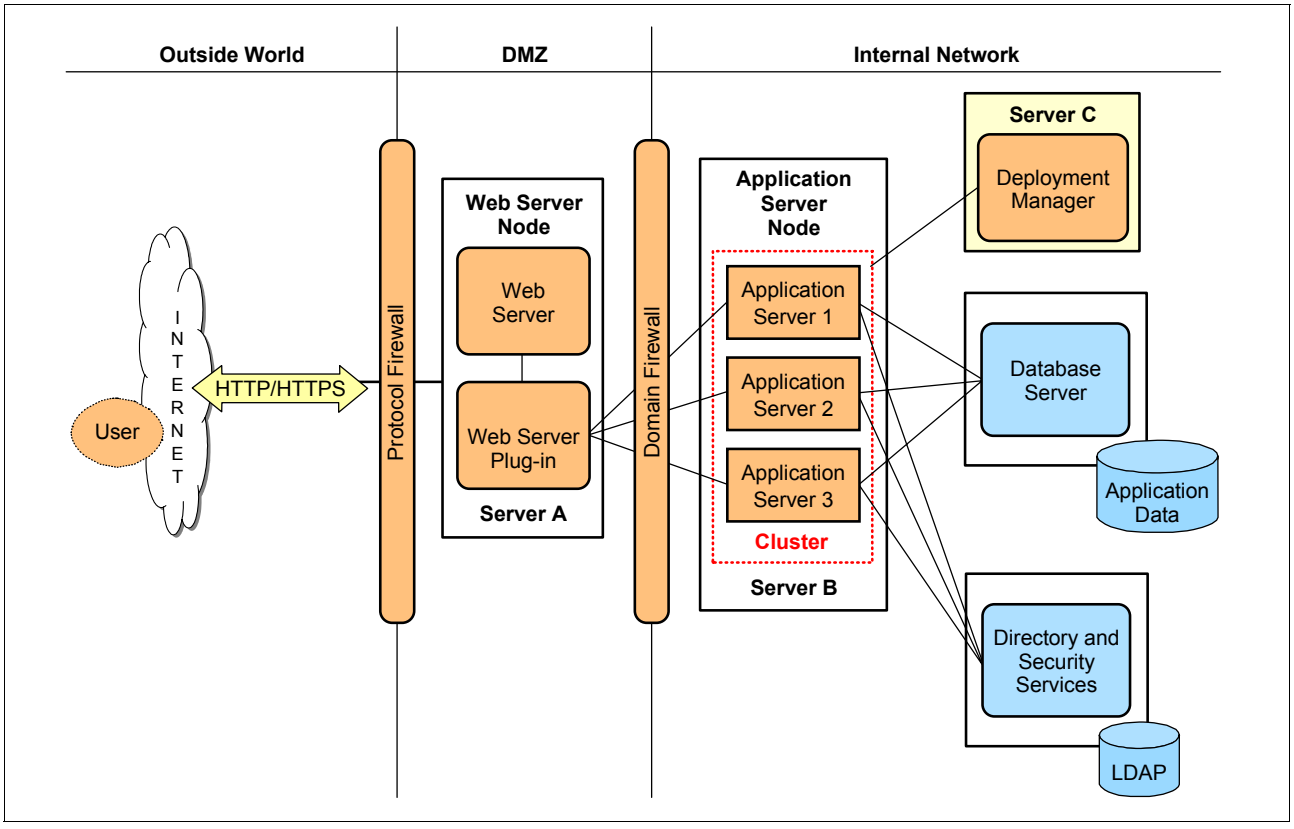


Figure 7-8 Vertical scaling

Figure 7-8 shows a vertical scaling example that includes a cluster with three cluster members. In this case, the Web server plug-in routes the requests according to the application servers availability. Load balancing is performed at the Web server plug-in level based on a round-robin algorithm and with consideration of the session state. Failover is also possible as long as there are active application servers (JVMs) on the system.

Vertical scaling can be combined with other topologies to boost performance, throughput, and availability.

7.6 Topology for horizontal scaling

Horizontal scaling exists when the cluster members are located across multiple machines. This lets a single application span over several machines, yet still presents the application as a single logical image. Figure 7-9 illustrates a horizontal scaling topology with two Application Server Nodes, each one on a separate machine (Servers B and C). Notice that there is a fourth server, Server D, where the Deployment Manager is installed to manage the cluster.

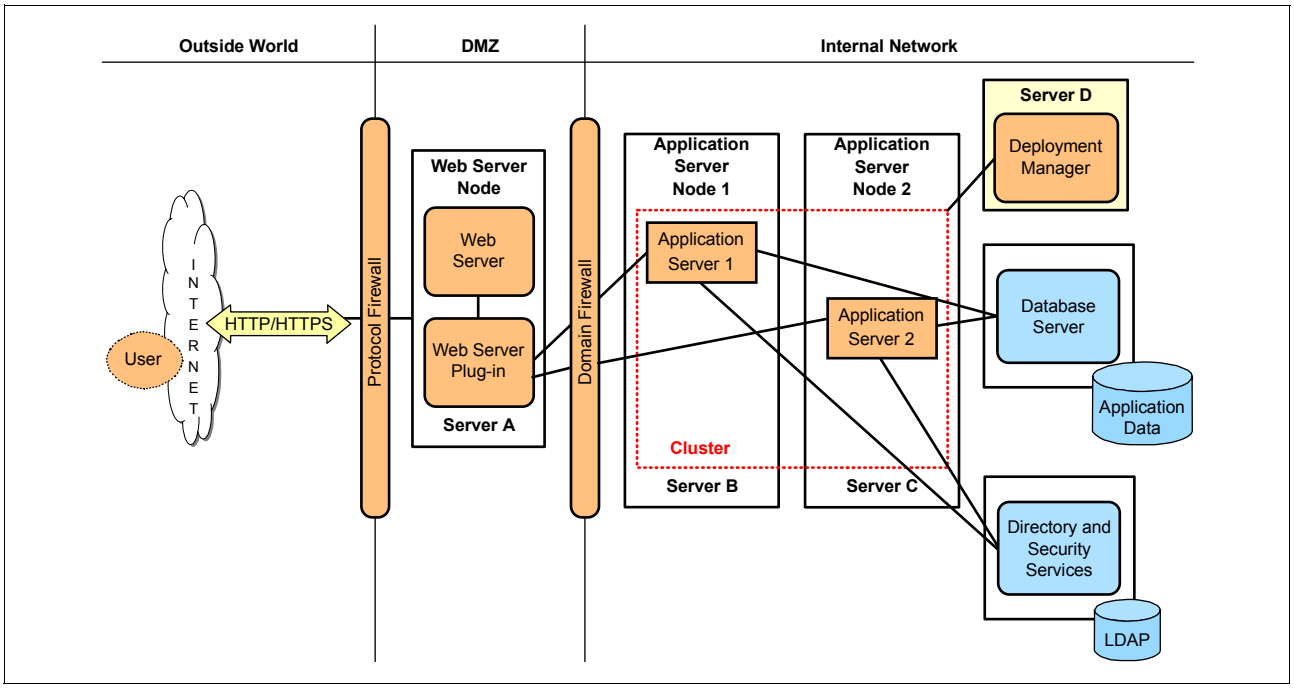


Figure 7-9 Horizontal scaling with cluster

The Web server plug-in distributes requests to the cluster members on each node and performs load balancing and failover. If the Web server (Server A) goes down, then the WebContainer Inbound Chain of Server B or C could be utilized (limited throughput) while Server A or the Web server on Server A is repaired.

Be aware that this configuration introduces a single point of failure; when the HTTP server is out of service, your entire application is inaccessible from the outside network (internal users could still access the application server(s) using the WebContainer Inbound Chain). You can omit this SPOF by adding a backup Web server.

If any component in the Application Server Node 1 (hardware or software) fails, the Application Server Node 2 can still serve requests from the Web Server Node and vice versa.

The Load Balancer, part of the WebSphere Edge Components, can be configured to create a cluster of Web servers and add it to a cluster of application servers. This is shown in 7.7, “Topology with IP sprayer front end” on page 252.

7.7 Topology with IP sprayer front end

Load balancing products can be used to distribute HTTP requests among Web servers that are running on multiple physical machines.

The Dispatcher component of Load Balancer, which is part of the WebSphere Edge Components, is an IP sprayer that performs intelligent load balancing among Web servers based on server availability and workload capacity as the main selection criteria to distribute the requests. Refer to Chapter 4, “Introduction to WebSphere Edge Components”, of *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392, for more details.

Figure 7-10 illustrates a horizontal scaling configuration that uses an IP sprayer on the Load Balancer Node to distribute requests between Web servers on multiple machines.

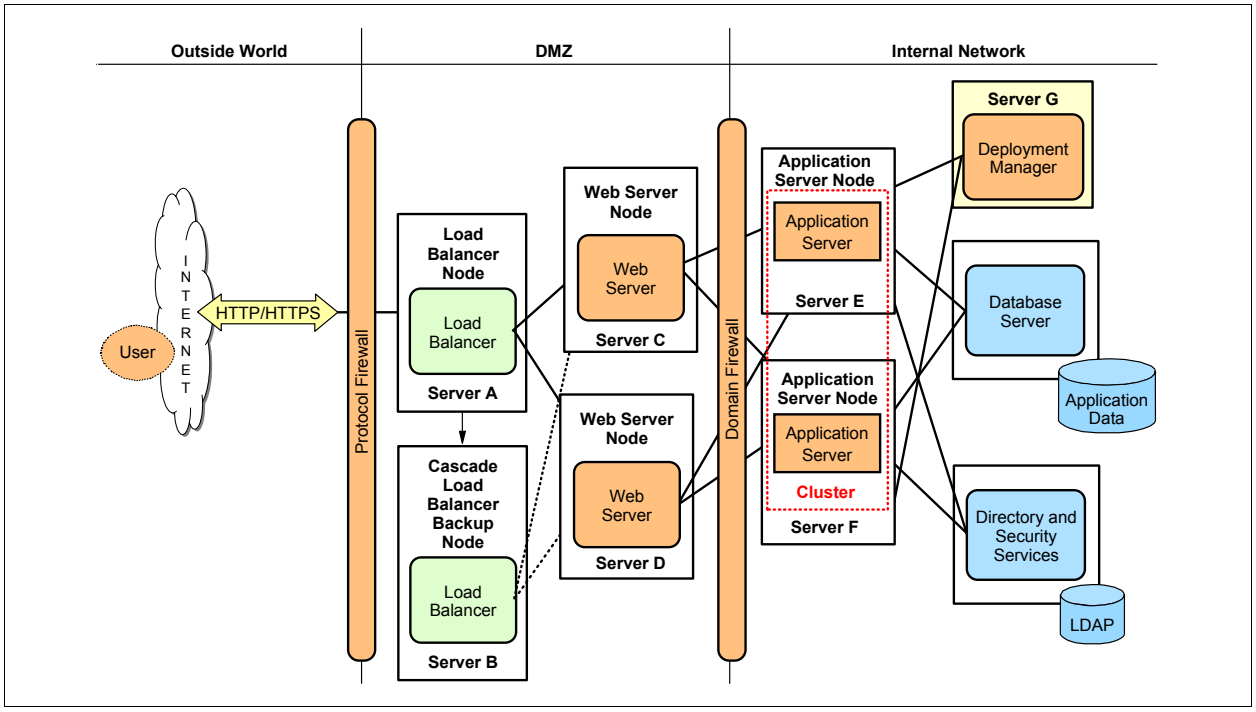


Figure 7-10 IP sprayer horizontally scaled topology

The Load Balancer Node sprays Web client requests to the Web servers. The Load Balancer is configured in cascade. The primary Load Balancer communicates to this backup through a heartbeat to perform failover, if needed, and thus eliminates the Load Balancer Node as a single point of failure.

Both Web servers perform load balancing and failover between the application servers (cluster members) through the Web server plug-in.

Tip: The Web server and Load Balancer can be collocated.

If any component on Server C, D, E, or F fails, the other ones can still continue receiving requests.

7.7.1 An example of IP Sprayer implementation

In this section, we demonstrate how the WebSphere Edge's Dispatcher component, also known as IP sprayer, can be configured on Solaris. The example we use here is with Solaris Logical Domains (LDoms), as detailed in 5.1.2, "Logical Domains" on page 117. This procedure can be applied to any Solaris systems, including systems with Dynamic Domains.

One of the test cases we performed is to demonstrate that a Load Balancer's Dispatcher component can distribute workloads between two Solaris 10 systems that are deployed in LDom, as shown in Figure 7-11. The dispatcher can be configured to work with servers that use standard protocols like HTTP. Thus, in our example, we show how the dispatcher can be configured to communicate with two WebSphere Application Server nodes.

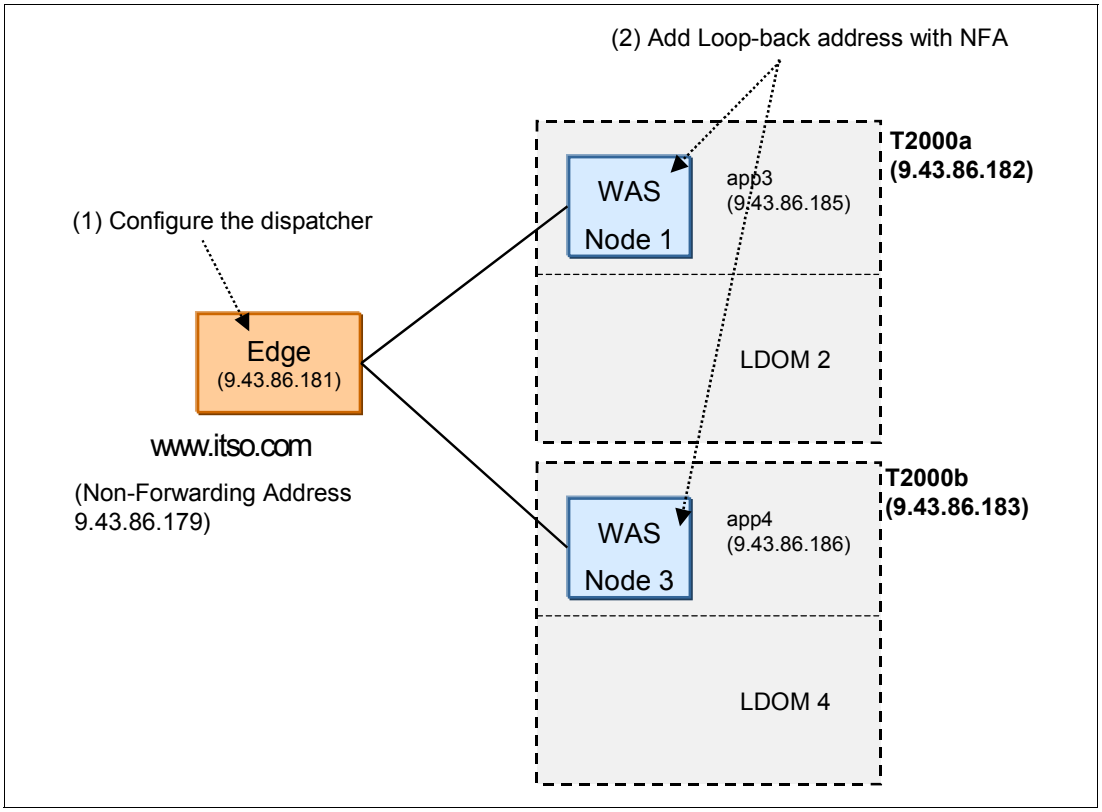


Figure 7-11 An example Edge's Dispatcher distributing workloads to back-end WebSphere Application Server nodes

Procedures to configure the Dispatcher

To successfully configure the Load Balancer's Dispatcher component, you must perform two steps of configuration: one on the Edge system and the rest on the back-end systems that could be serving the HTTP protocol, such as IHS or WebSphere Application Server. In our case, it is WebSphere Application Server nodes.

Part 1: Configure the Edge system

You must configure the Edge component:

1. Edit the ibmlb.conf file, as shown in Example 7-1, to make sure that the correct network device information is defined in this file. You can use the **`ifconfig -a`** command to query the available network devices on the system.

Example 7-1 Sample ibmlb.conf bge interface on Sun Solaris Server

```
# cat /opt/ibm/edge/lb/servers/ibmlb.conf
bge  -1  0 ibmlb
```

2. Set up the Dispatcher and its components, namely the server, executor, manager, and advisor, as shown in Example 7-2.

Example 7-2 Sample setup configuration

```
dsserver
dscontrol set loglevel 1
dscontrol executor start
dscontrol executor xm 46 1

dscontrol cluster add www.itso.com address 9.43.86.179 primaryhost 9.43.86.181
dscontrol cluster set www.itso.com proportions 49 50 1 0
dscontrol executor configure 9.43.86.179 bge0 255.255.252.0

dscontrol port add www.itso.com:9080 reset no

dscontrol server add www.itso.com:9080:app3 address 9.43.86.185
dscontrol server set www.itso.com:9080:app3 weight 10

dscontrol server add www.itso.com:9080:app4 address 9.43.86.186
dscontrol server set www.itso.com:9080:app4 weight 10

dscontrol manager start manager.log 10004

dscontrol advisor start Http 9080 Http_9080.log
```

Part 2: Configure the back-end servers

You must add the non-forwarding address for loop-back to the back-end HTTP servers. In our case, the back-end servers are WebSphere Application Server nodes. You must repeat this action for all the back-end systems defined as the dispatcher's cluster nodes.

Example 7-3 Add the non-forwarding address for the Load Balancer

```
global# ifconfig lo0:1 plumb 9.43.86.179 netmask 255.255.252.0 up
```

Testing the environment

Go to any client that can reach the www.itso.com address. Bring up a Web browser and go to http://www.itso.com, assuming this URL's address is resolvable by your client.

7.8 Topology with redundancy of several components

The idea behind having as much redundancy of components as possible is to eliminate (keep minimized) the single points of failure (SPOF). Most of the components allow some kind of redundancy like a Load Balancer backup node for the primary Load Balancer node, or clustered Web servers or application servers, and so on. Some other components, such as the Deployment Manager, do not support any automatic backup or failover. Figure 7-12 on page 256 illustrates a topology with redundancy for several components, including:

- Two Load Balancers

  The one on Server A is the primary (active) Load Balancer. It is synchronized, through a heartbeat, with a backup Load Balancer (in standby status) on another machine, Server B.

- Two Web servers

  Both of them receive requests from the Load Balancer and share the requests that come from the Internet. Each one is installed on a different machine.

- An application server cluster

  The cluster implements vertical and horizontal scaling.

- Eight cluster members

  Two on each Application Server Node.

- Four Application Server Nodes

  Each one hosting two application servers. The nodes on Server E are independent installations. The nodes on Server F are profiles of a single installation.

- Two database servers

  Using a HA (high availability) software product. This means that one copy of the database is the one that is being used and the other one is a replica that will replace the first one if it fails.

- Two LDAP servers

  Using a HA (high availability) software product. This means that one copy of the database is the one that is being used and the other one is a replica that will replace the first one if it fails.

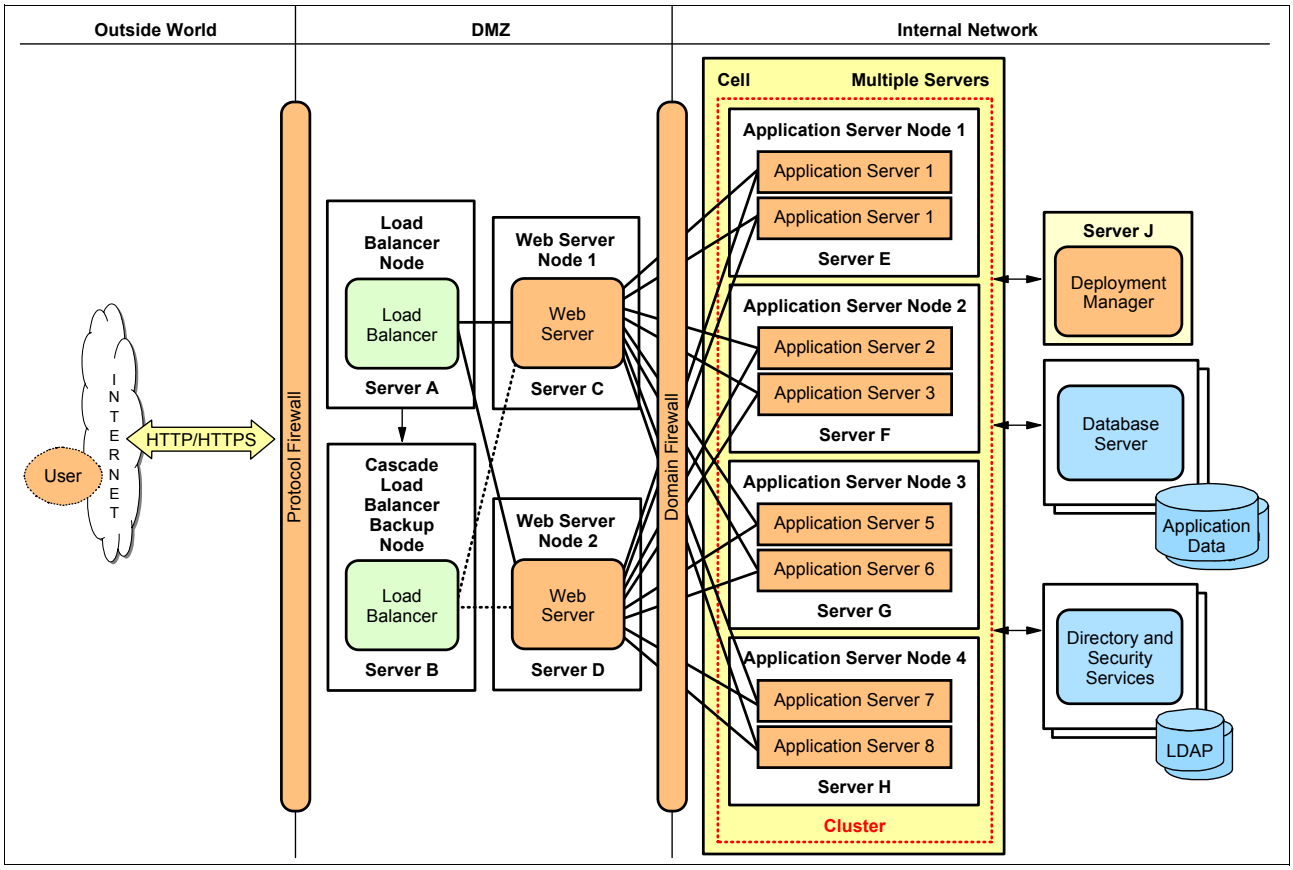


Figure 7-12 Topology with redundancy of several components

7.8.1 Implementing system level redundancy with Solaris Cluster

As discussed in 7.1.3, “Availability” on page 238, WebSphere Application Server has built-in fail-over capabilities except for the Deployment Manager. There are also other software components that WebSphere Application Server relies on that need to rely on external system’s HA solution. The Solaris Cluster system is an integrated hardware and software solution that is used to create highly available and scalable services. The Solaris Cluster system extends Solaris into a cluster operating system. Solaris Cluster provides many HA agents, including WebSphere Message Queue (WMQ), WebSphere Message Brokers (WMB), and DB2. You can also build an HA solution for your WebSphere Deployment Manager.

Important: For details about using Sun Cluster and WebSphere, refer to *WebSphere Application Server Network Deployment V6: High Availability Solutions*, SG24-6688.

7.9 Topology with other Web and reverse-proxy servers

WebSphere Application Server V6.1 supports a number of different Web servers and provides the relevant plug-in installation and configuration utility. These Web servers are then used to front end the WebSphere Application Server nodes and manages client connections. They can also perform static content hosting as well as Web container processing of JSPs and servlets. For example, you can handle J2EE components and Web services processing

on the WebSphere Application Server nodes and the static contents, such as HTML, CSS, and images, on these Web servers. The details of various Web servers supported by WebSphere Application Server can be found at:

http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg27006921

Sun Java System Web Server (SJSWS) is a highly scalable and secure Web server that can front end the WebSphere Application Server environment in different ways. In addition to front ending WebSphere Application Server, there are a number of other benefits that SJSWS can provide. We discuss some example scenarios here to help you take advantage of numerous features of SJSWS. The details of various features and documentation of the Web server can be found at:

http://docs.sun.com/app/docs/coll/1653.1

7.9.1 Sample scenarios using SJSWS

As Sun Java System Web Server can be used in various ways in application deployment topologies; here we list some of the most common scenarios to help you decide the best configuration for your deployment:

- Use as a front-end Web server.
- Use as a Web server to distribute the load over a number of WebSphere Application Server servers using SJSWS as a reverse proxy.
- To off-load the SSL processing to the Web server so as to conserve the resources on the WebSphere Application Server server.
- To leverage underlying hardware features (for example, SSL acceleration).
- Deploy some of the dynamic contents on SJSWB as well as the Web server to provide some of the J2EE processing capability.

7.9.2 Configuring the WebSphere plug-in for Sun Java System Web Server

We described the concept of workload management using the Web server plug-in in 7.2.2, “Workload management with Web server plug-in” on page 243. The procedure is defined in the IBM InfoCenter at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/twsv_plugin.html

7.9.3 Reverse proxy configuration

Sun Java System Web Server has a built-in HTTP reverse proxy capability that is highly scalable to front end the WebSphere Application Server application environment. A reverse proxy acts like a proxy that may appear to the client as serving their request, but actually it gets forwarded to another server, in this case, WebSphere Application Server. The details of the server processing of the client request is not actually known to the clients. As the proxy server appears to be a Web server serving the client requests, clients are transparent to the reverse proxy and do not need any additional configuration. All configurations have to be done at the Web server side only. Neither the WebSphere Application Server or clients need special configuration to use this feature. When configuring a reverse proxy to front end a group of WebSphere Application Server nodes, it is necessary to for all nodes to have the same set of applications deployed for proper functionality. This is the same for WebSphere Application Server nodes front ended with Web server plug-ins. By configuring a given a Web server as a reverse proxy to forward requests to similarly configured multiple WebSphere

Application Server nodes, the reverse proxy can operate as an application level software load balancer, as shown in Figure 7-13.

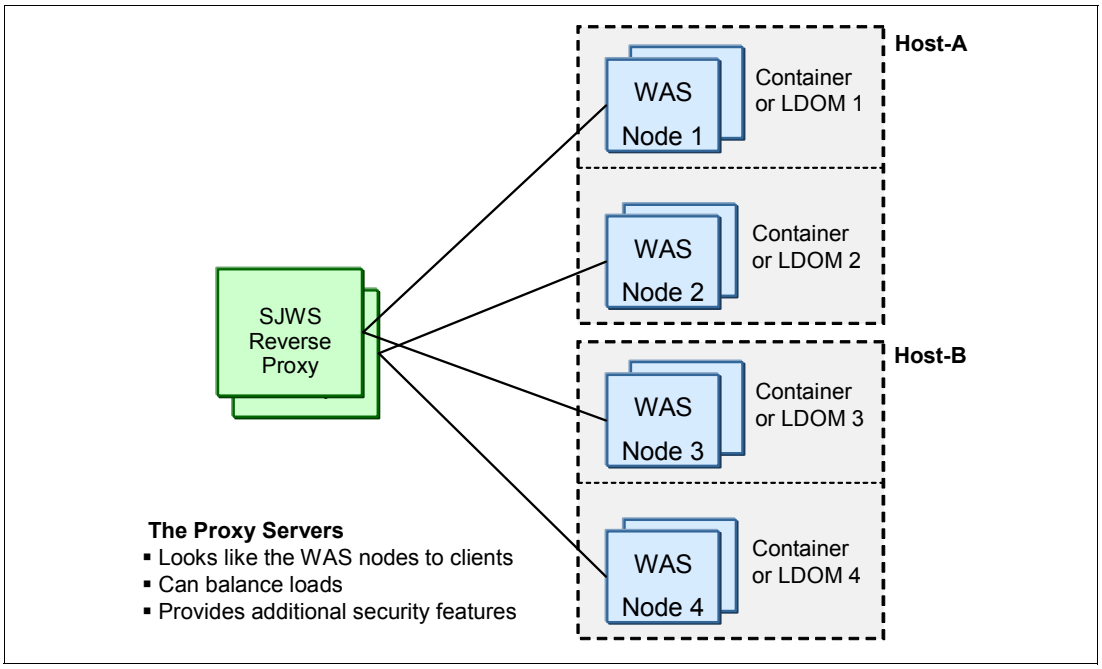


Figure 7-13 Topology with Sun Java System Web Server as reverse proxy server

Details of the configuration steps for reverse proxy can be found here:

http://docs.sun.com/app/docs/doc/820-1061/6ncopp92h

When you follow the steps mentioned in the above documentation to configure the proxy server with the Sun Java System Web Server admin console, your steps will resemble the windows shown in Figure 7-14 to Figure 7-17 on page 260.

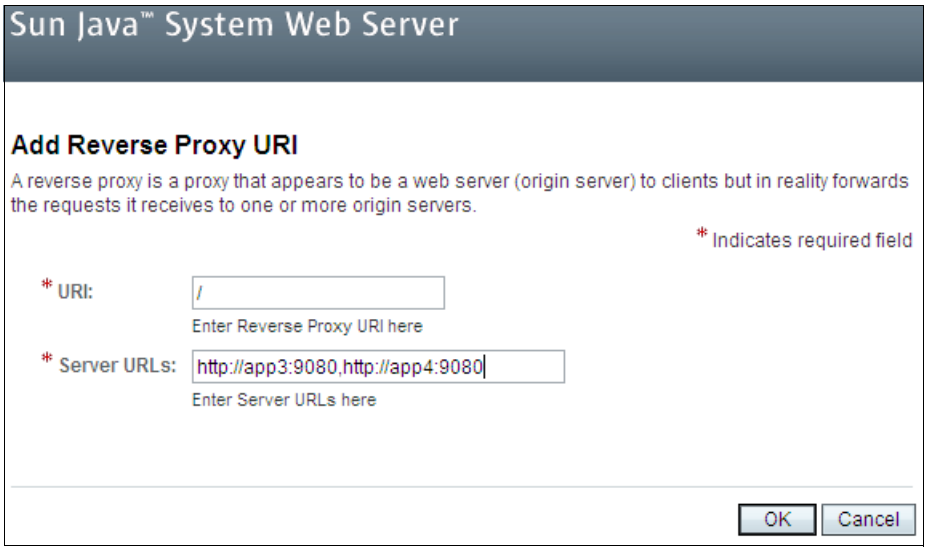


Figure 7-14 Sun Java System Web Server Proxy configuration Step 1

After clicking **OK** in Figure 7-14 on page 258, your configuration is created and will be validated during deployment, as shown in Figure 7-15, Figure 7-16, and Figure 7-17 on page 260.

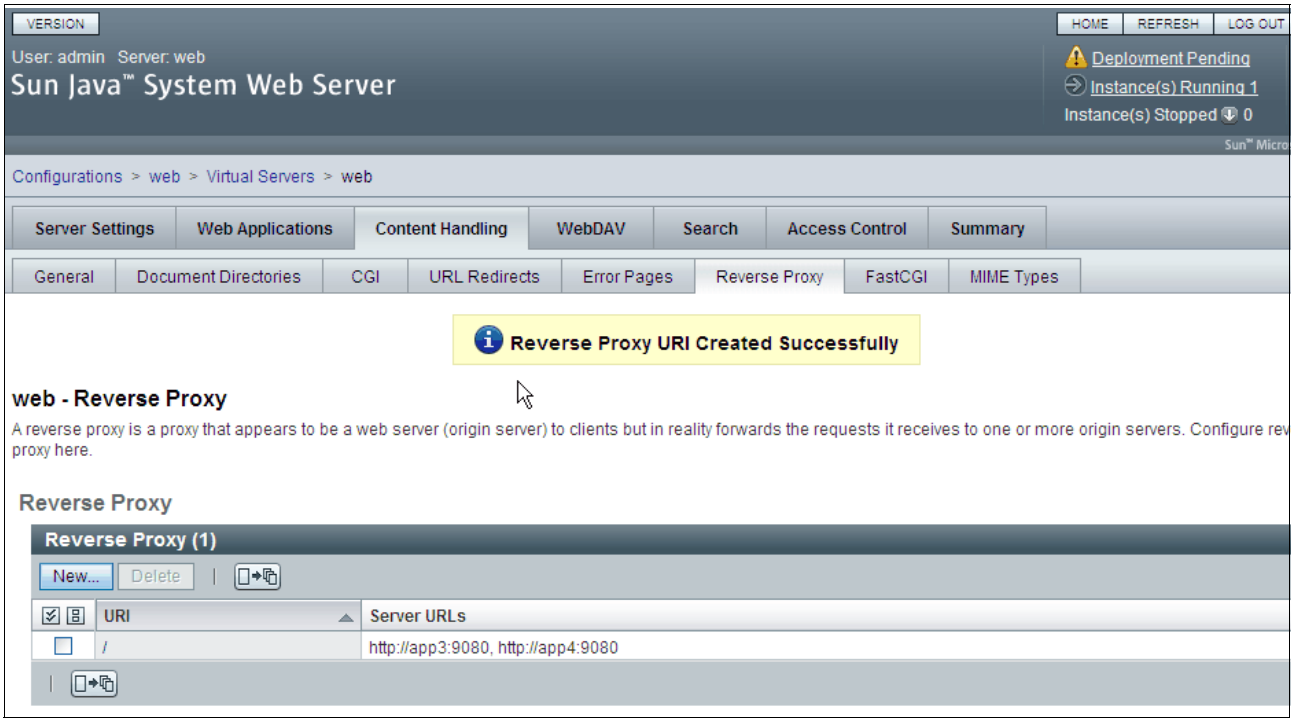


Figure 7-15 Sun Java System Web Server Proxy configuration Step 2

Figure 7-16 Sun Java System Web Server Proxy configuration Step 3

Figure 7-17 Sun Java System Web Server Proxy configuration step 4

In addition to very basic configuration, there are utilities that can be set to instruct SJSWS to do special processing:

- Sticky Cookie: Cookie name that, when present in a response, will cause subsequent requests to stick to that origin server.
- Sticky URI Parameter: The URI parameter name to inspect for route information. When the URI parameter is present in a request URI and its value contains a colon (:) followed by a route ID, the request will "stick" to the origin server identified by that route ID.
- Route Header: The HTTP request header name used to communicate route IDs to origin servers.
- Route Cookie: Cookie name generated by the server when it encounters a sticky cookie in a response. The route cookie stores a route ID that enables the server to direct subsequent requests back to the same origin server.

7.9.4 Offload SSL processing to SJSWS

As discussed in 7.1.6, “Performance impact of WebSphere Application Server security” on page 240, WebSphere Application Server performance can be impacted by enabling security features such as SSL processing. WebSphere Application Server can offload its SSL overhead to SJSWS, which can substantially reduce the processing impact on WebSphere Application Server. This can be augmented by Sun hardware features, such as Niagara Crypto Processor (NCP) on UltraSPARC-T1 and T2 based systems, to further offload the SSL processing from WebSphere Application Server and SJSWS. This way, all the resources on your WebSphere Application Server hosts can be spent on processing the actual user requests and business logic while the Web server takes over the burden of processing the SSL.

The best part of this capability is that you do not need any additional configuration on your client or server (WebSphere Application Server) sides. All the configuration has to done on the SJSWS system to enable it to listen on the SSL port to receive the HTTPS requests, which in turn sends plain HTTP requests to WebSphere Application Server. As mentioned before, servers like Sun SPARC Enterprise T2000 and T5220 have built-in cryptographic modules, NCP, which SJSWS can further configure to take advantage of onboard SSL processing. This can also reduce your need for specialized hardware and simplify your deployment environment. Details about configuring NCP with Sun Java System Web Server can be found here:

http://www.sun.com/blueprints/0306/819-5782.pdf

7.9.5 Deploy dynamic Web applications on SJSWS

As the SJSWS implements the latest JSP™ and Servlet standard available in J2EE, it can be used to write these kind of applications and, for additional EJB and other complex application server processing, it can depend on WebSphere Application Server. This will minimize the processing on WebSphere Application Server and instead of just doing the connection management, the Web server can do some processing on behalf of the WebSphere Application Server. The WebSphere Application Server can now do the complex tasks that cannot be handled by the Web server; in this way, resources can be optimized and processing can be minimized at different levels based on deployment need.

7.10 Topology for dynamic Scalability with WebSphere Extended Deployment

WebSphere Extended Deployment (XD) provides an IT infrastructure that dynamically and reliably adapts to changing business demands. WebSphere Extended Deployment extends the capabilities of WebSphere Application Server Network Deployment and other middleware to help you optimize the utilization and management of your deployments and enhance the quality of service of your business-critical applications. WebSphere XD is a separate product, but it is worth mentioning for our discussion about advanced topologies for WebSphere application environments.

Description of WebSphere Application Server Extended Deployment (XD)

In WebSphere Application Server XD V6.1, you will find that previous capabilities still exist, but are enhanced and available selectively. You can get all the capabilities, or select a subset from the following packaging options:

- WebSphere Extended Deployment Operations Optimization
- WebSphere Extended Deployment Data Grid
- WebSphere Extended Deployment Compute Grid

You can get more detailed information about the XD product in *Optimizing Operations with WebSphere Extended Deployment V6.1*, SG24-7422.

Static versus dynamic

In this chapter, we discuss various scalable topologies based on a traditional WebSphere Application Server environment that is static in nature. It is comprised of a set number of servers and clusters that serve specific applications. Resources are dedicated to applications to ensure that they operate at capacity during peak loads. Because different applications often have varying needs for resources (high at times, low at others), this resource dedication

often leads to an excess of physical capacity in terms of CPU and memory during off-peak periods.

The characteristics of these static environments is that they are not making the best use of the overall capacity and the configuration in terms of numbers of servers. Additionally, these environments cannot quickly respond to unexpected changes in workload. For example, if an application has a dramatic increase in load, there may be insufficient capacity in the servers set aside for that application to meet the demand. However, there may be sufficient capacity in other servers running other applications that cannot be used.

By using the dynamic operations features of Operations Optimization, you can change the way a typical WebSphere environment is configured today to one that has the following features:

- Improves the utilization of available resources, such as CPU and memory.
- Classifies and monitors the workload.
- Provides a business-centric view of the workload and how it is performing.
- Can respond in real time to changes in the workload mix (without human intervention if so desired), using business guidelines that the organization specified.

WebSphere Extended Deployment 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 (or unexpected) spikes in workload demand, resources can be allocated where they are needed most.

User-defined policies based on business requirements specify performance goals for each application. WebSphere Extended Deployment dynamically allocates resources to each application aiming to meet these performance goals.

Optimization of the computing resources that you already own might allow you to run more applications on the machines that you already have in place.

One of the key elements of WebSphere XD's Operations Optimization V6.1 is the *On Demand Router (ODR).* The ODR is an intelligent proxy that acts as the entry point for traffic coming into an Extended Development cell, performing request prioritization, flow control, and dynamic workload management for HTTP requests and SOAP over HTTP requests.

We discuss briefly how to apply this ODR along with Sun virtualization technologies in Chapter 5, "Configuration of WebSphere Application Server in an advanced Solaris 10 Environment" on page 115 to build out an advanced WebSphere deployment topology.

7.10.1 A sample topology with XD ODR

The ODR is responsible for orchestrating the queuing and dispatching of requests in accordance with your service policy. In optimizing queue lengths and dispatch rates, several factors are considered, including the processing capacity of the nodes, performance goals, relative importance, and load balancing.

The ODR is a component that logically replaces and extends the functionality of the WebSphere Web server plug-in. However, the ODR can and often does work in concert with existing Web servers that use the WebSphere Web server plug-in.

The ODR provides the standard functionality of an HTTP/1.0 and HTTP/1.1 compliant proxy server with the following added on demand features:

- Request classification and prioritization
- Request queuing
- Request routing
- Dynamic load balancing
- HTTP session affinity
- SSL ID affinity
- WebSphere Partitioning Facility partition affinity

Important: Because the ODR is the entry point into your application server environment, it is extremely important that you make the ODR highly available by using multiple ODRs.

There can be multiple ODRs in a topology. Each request goes through only one ODR, but for environments with more than one ODR, the initial request could be routed to any one of them.

The ODR is *aware* of its environment because of a component called the on demand configuration service (ODC). The ODC collects information about all of the WebSphere Extended Deployment and WebSphere Network Deployment application servers and applications that are deployed in the cell. It also gathers information about the defined non-WebSphere middleware servers and middleware applications. This information is used by the ODR to dynamically route HTTP requests to both WebSphere and non-WebSphere servers. The ODC dynamically configures the routing rules at runtime.

Figure 7-18 shows an example (logical diagram) of how the XD ODR can be implemented in the topologies we have previously discussed in this chapter.

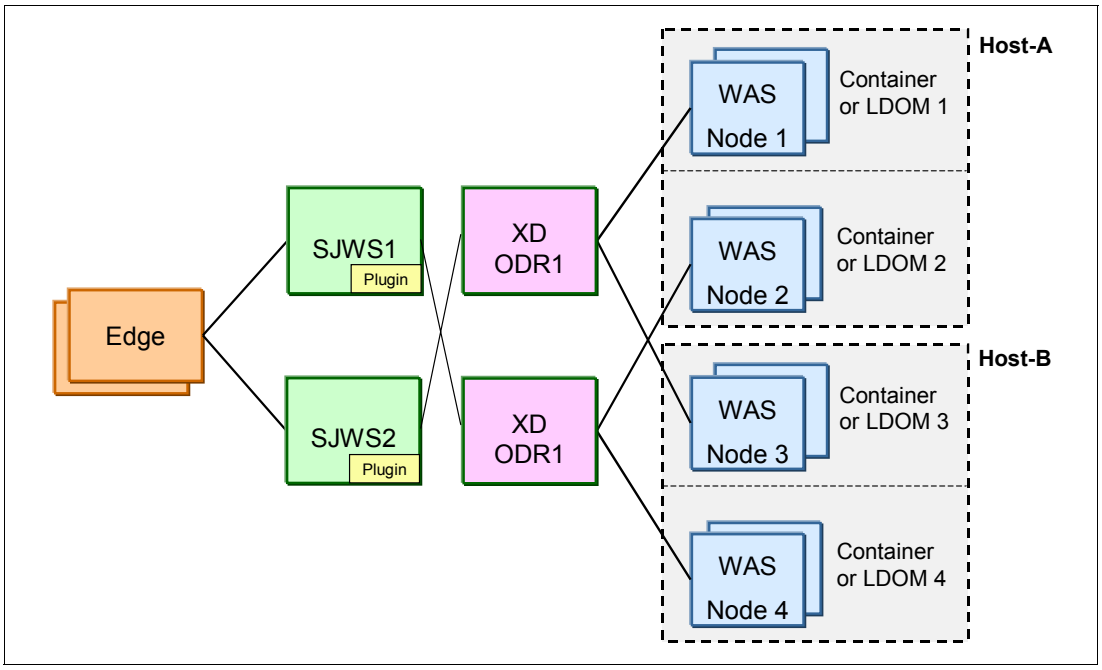


Figure 7-18 IBM WebSphere Edge Component, Sun Java System Web Server, and IBM WebSphere XD ODR

Note: You can implement and deploy the WebSphere XD ODR component in the Sun virtualization environment, such as Containers and LDoms, as described for WebSphere Application Server V6.1 in Chapter 5, “Configuration of WebSphere Application Server in an advanced Solaris 10 Environment” on page 115.

7.11 Implementation considerations

As discussed in this chapter, there are a number of ways you can achieve your enterprise deployment. It will have different characteristics based on the deployment technology that you choose for WebSphere Application Server as well as the virtualization capabilities that Solaris 10 provides. The capabilities from both work together to provide various deployment options for high availability and scalability. During the implementation phase, it is essential to understand the capabilities and limitations of the particular technology chosen.

With Solaris zones, you can deploy WebSphere Application Server nodes in separate virtual operating environment while they are isolated at the process level. With the WebSphere Application Server HA clustering capability, you can configure these nodes to handle workload in the cluster. However, as you realize if the cluster members deployed in zones end up being hosted on a single physical system, the hardware becomes a single point of failure. Similarly, LDoms can provide process level isolation as well as different OS instances and patch levels, but the underlying hardware is still a single point of failure. You should also take similar consideration when choosing the Web server, plug-ins or WebSphere XD ODR. As shown in Figure 7-18 on page 263, when two WebSphere Application Server instances are connected to different ODR, they can provide some level of redundancy to prevent the system downtime in case of hardware failure.

For higher availability at the hardware level, you can achieve it with Sun servers that have Dynamic System Domains or implement with Sun Cluster technology. Additionally, you can use Solaris Resource Management capabilities to ensure that collocated services on a system can coexist well together while maximizing the system utilization. You can also use Solaris least privilege security mechanism to limit applications’ process privileges in case of malicious activities.

To implement WebSphere advanced topologies, you need high investment in software, hardware, and skilled personnel. Your WebSphere production system’s reliability is determined by the weakest link in the whole system chain. Therefore, it is very important to evaluate each part of the end-to-end in your deployment topology and eliminate any potential single points of failure.

Finally, understanding the scalability of all these components in your WebSphere infrastructure and applying appropriate scaling techniques can greatly improve availability and performance.

8

Security and identity management

This chapter provides information about IBM WebSphere Application Server security components, which collaborate with other services to provide authentication, authorization, delegation, and data protection.

This chapter discusses the following topics:

- WebSphere Application Server security components
- User registry
- Secure communications using SSL
- Kernel SSL Proxy
- Integrating third-party HTTP reverse proxy servers

8.1 WebSphere Application Server Security components

IBM WebSphere Application Server provides security components that collaborate with other services to provide authentication, authorization, delegation, and data protection. WebSphere Application Server also supports the security features described in the Java 2 Enterprise Edition (J2EE) specification.

Security is established at two levels. The first level is administrative security. Administrative security applies to all applications running in the environment and determines whether security is used at all, the type of registry against which authentication takes place, and other values, many of which act as defaults.

The second level is application security. Application security, which can vary with each application, determines the requirements specific to the application. In some cases, these values can override global defaults. Application security includes settings such as mechanisms for authenticating users and authorization requirements.

Administrative security

Administrative security determines whether security is used at all, the type of registry against which authentication takes place, and other values, many of which act as defaults. Proper planning is required because incorrectly enabling administrative security can lock you out of the administrative console or cause the server to end abnormally.

Administrative security can be thought of as a "big switch" that activates a wide variety of security settings for WebSphere Application Server. Values for these settings can be specified, but they will not take effect until administrative security is activated. The settings include the authentication of users, the use of Secure Sockets Layer (SSL), and the choice of user account repository. In particular, application security, including authentication and role-based authorization, is not enforced unless administrative security is active. As of WebSphere Application Server V6.1, administrative security can be enabled during product installation.

Administrative security represents the security configuration that is effective for the entire security domain. A security domain consists of all of the servers that are configured with the same user registry realm name. In some cases, the realm can be the machine name of a local operating system registry. In this case, all of the application servers must reside on the same physical machine. In other cases, the realm can be the machine name of a stand-alone Lightweight Directory Access Protocol (LDAP) registry.

The basic requirement for a security domain is that the access ID that is returned by the registry or repository from one server within the security domain is the same access ID as that returned from the registry or repository on any other server within the same security domain. The access ID is the unique identification of a user and is used during authorization to determine if access is permitted to the resource.

The administrative security configuration applies to every server within the security domain.

Reasons for administrative security

Turning on administrative security activates the settings that protect your server from unauthorized users. Administrative security is enabled by default during the profile creation time. There might be some environments where no security is needed, such as a development system. On these systems, you can elect to disable administrative security. However, in most environments, you should keep unauthorized users from accessing the administrative console and your business applications. Administrative security must be enabled to restrict access.

What administrative security protects

The configuration of administrative security for a security domain involves configuring the following technologies:

- Authentication of HTTP clients
- Authentication of IIOP clients
- Administrative console security
- Naming security
- Use of SSL transports
- Role-based authorization checks of servlets, enterprise beans, and Mbeans
- Propagation of identities (RunAs)
- The common user registry
- The authentication mechanism
- Other security information that defines the behavior of a security domain includes:
  - The authentication protocol (Remote Method Invocation over the Internet Inter-ORB Protocol (RMI/IIOP) security)
  - Other miscellaneous attributes

Securing your environment before installation

The following instructions explain how to perform a product installation with proper authority. Complete the following steps when you plan to use the local operating system as your user registry:

1. Log on as root and verify that the umask value is 022.
2. To verify that the umask value is 022, run the **umask** command.
3. To set up the umask value as 022, run the **umask 022** command.
4. Make sure that the /etc directory contains a shadow password file. The shadow password file is named shadow and is in the /etc directory. If the shadow password file does not exist, an error occurs after enabling administrative security and configuring the user registry as local operating system.
5. To create the shadow file, run the **pwconv** command (without any parameters). This command creates an /etc/shadow file from the /etc/passwd file. After creating the shadow file, you can configure local operating system security.

Securing your environment after installation

WebSphere Application Server depends on several configuration files that are created during installation. These files contain password information and need protection. Although the files are protected to a limited degree during installation, this basic level of protection is probably not sufficient for your site. You should verify that these files are protected in compliance with the policies of your site.

Note: A Kerberos keytab configuration file contains a list of keys that are analogous to user passwords. It is important for hosts to protect their Kerberos keytab files by storing them on the local disk, which makes them readable only by authorized users.

The files in the app_server_root/profiles/*profile_name*/config and app_server_root/profiles/*profile_name*/properties need protection. For example, give permission to the user who logs onto the system for WebSphere Application Server primary administrative tasks. Other users or groups, such as WebSphere Application Server console users and console groups, need permissions as well.

This procedure applies only to the ordinary UNIX file system. If your site uses access control lists, secure the files by using that mechanism. Any site-specific requirements can affect the owner, group, and corresponding privileges should be used.

Do the following:

1. Go to the install_root directory and change the ownership of the directory configuration and properties to the user who logs onto the system for WebSphere Application Server primary administrative tasks. Run the following command:

   ```
   chown -R logon_name directory_name
   ```

   Where *login_name* is a specified user or group and *directory_name* is the name of the directory that contains the files

   We recommend that you assign ownership of the files that contain password information to the user who runs the application server. If more than one user runs the application server, provide permission to the group in which the users are assigned in the user registry.

2. Set up the permissions by running the following command:

   ```
   chmod -R 770 directory_name
   ```

3. Go to the app_server_root/profiles/*profile_name*/properties directory and set the file permissions. Set the access permissions for the following files as it pertains to your security guidelines:
   - TraceSettings.properties
   - client.policy
   - client_types.xml
   - sas.client.props
   - sas.stdclient.properties
   - sas.tools.properties
   - soap.client.props
   - wsadmin.properties
   - wsjaas_client.conf

For example, you might issue the command **chmod 770** *file_name*, where *file_name* is the name of the file listed previously in the install_root/profiles/*profile_name*/properties directory. These files contain sensitive information, such as passwords.

4. Create a group for WebSphere Application Server and put the users who perform full or partial WebSphere Application Server administrative tasks in that group.

Tip: If you want to use WebSphere MQ as a Java Messaging Service (JMS) provider, restrict access to the /var/mqm directories and log files used. Give write access to the user ID mqm or members of the mqm user group only.

Important: After securing your environment, only the users with permission can access the files. Failure to adequately secure these files can lead to a breach of security in your WebSphere Application Server applications.

8.1.1 Enabling security

By enabling security, you protect your server from unauthorized users and are then able to provide application isolation and requirements for authenticating application users.

It is helpful to understand security from an infrastructure perspective so that you know the advantages of different authentication mechanisms, user registries, authentication protocols, and so on. Picking the right security components to meet your needs is a part of configuring security.

Note: For WebSphere Application Server V6.1, administrative security is enabled by default whenever a new profile is created, either during the initial install when you create a new profile or during post-install when you use the profile creation tooling. You can decide not to enable administrative security during profile creation time by instead enabling security post-profile creation using the administrative console.

You can configure security in WebSphere Application Server using the following steps:

1. Start the WebSphere Application Server administrative console.

   If security is currently disabled, you are only prompted for a user ID. Log in with any user ID. However, if security is currently enabled, you are prompted for both a user ID and a password. Log in with a predefined administrative user ID and password.

2. Select **Security** → **Secure administration, applications, and infrastructure**. Use the Security Configuration Wizard to configure security, or do it manually. The configuration order is not important.

3. Configure the user account repository. On the Secure administration, applications, and infrastructure window, you can configure user account repositories, such as federated repositories, local operating system, stand-alone Lightweight Directory Access Protocol (LDAP) registry, and stand-alone custom registry.

Tip: You can choose to specify either a server ID and password for interoperability or enable a WebSphere Application Server V6.1 installation to automatically generate an internal server ID.

For more information about automatically generating server IDs, see *IBM WebSphere Application Server V6.1 Security Handbook* SG24-6316.

One of the details common to all user registries or repositories is the Primary administrative user name. This ID is a member of the chosen repository, but also has special privileges in WebSphere Application Server. The privileges for this ID and the privileges that are associated with the administrative role ID are the same. The Primary administrative user name can access all of the protected administrative methods.

In stand-alone LDAP registries, verify that the Primary administrative user name is a member of the repository and not just the LDAP administrative role ID. The entry must be searchable.

The Primary administrative user name does not run WebSphere Application Server processes. Rather, the process ID runs the WebSphere Application Server processes.

The process ID is determined by the way the process starts. For example, if you use a command line to start processes, the user ID that is logged into the system is the process ID. If running as a service, the user ID that is logged into the system is the user ID running the service. If you choose the local operating system registry, the process ID requires special privileges to call the operating system APIs. The process ID must have *root* privileges.

4. Select the **Set as current** option after you configure the user account repository. When you click **Apply** and the Enable administrative security option is set, a verification occurs to see if an administrative user ID has been configured and is present in the active user registry. The administrative user ID can be specified at the active user registry window or from the console users link. If you do not configure an administrative ID for the active user registry, the validation fails.

   When you switch user registries, the admin-authz.xml file should be cleared of existing administrative IDs and application names. Exceptions will occur in the logs for IDs that exist in the admin-authz.xml file but do not exist in the current user registry.

5. Configure the authentication mechanism.

   Configure Lightweight Third-Party Authentication (LTPA), which is the default authentication mechanism, in the Authentication mechanisms and expiration window. LTPA credentials can be forwarded to other machines. For security reasons, credentials expire; however, you can configure the expiration dates on the console. Valid LTPA credentials enable browsers to visit different product servers without having to authenticate multiple times.

6. Configure the authentication protocol for special security requirements from Java clients, if needed.

   You can configure Common Secure Interoperability Version 2 (CSIv2) through links on the Secure administration, applications, and infrastructure window. The Security Authentication Service (SAS) protocol is provided for backwards compatibility with previous product releases, but is deprecated. Links to the SAS protocol windows display on the Secure administration, applications, and infrastructure window if your environment contains servers that use previous versions of WebSphere Application Server and support the SAS protocol.

Important: SAS is supported only between Version 6.0.x and previous version servers that have been federated in a Version 6.1 cell.

Attention: IBM no longer ships or supports the Secure Authentication Service (SAS) IIOP security protocol. We recommend that you use the Common Secure Interoperability Version 2 (CSIv2) protocol.

7. Modify or a create a default Secure Sockets Layer (SSL) configuration. This action protects the integrity of the messages sent across the Internet. The product provides a single location where you can specify SSL configurations that the various WebSphere Application Server features that use SSL can utilize, including the LDAP registry, Web container, and the authentication protocol (CSIv2 and SAS). For more information, see 8.4.1, “Configuration of the KSSL module on Solaris” on page 303. After you modify a configuration or create a new configuration, specify it in the SSL configurations window. To get to the SSL configurations window, complete the following steps:

   a. Select **Security** → **SSL certificate and key management**.

b. Under Configuration settings, select **Manage endpoint security configurations** → ***configuration_name***.

c. Under Related items, click **SSL configurations**.

You can also edit the DefaultSSLConfig file or create a new SSL configuration with a new alias name. If you create a new alias name for your new keystore and truststore files, change every location that references the DefaultSSLConfig SSL configuration alias. The following list specifies the locations of where the SSL configuration repertoire aliases are used in the WebSphere Application Server configuration.

For any transports that use the new network input/output channel chains, including HTTP and Java Message Service (JMS), you can modify the SSL configuration repertoire aliases in the following locations for each server:

– Select **Security** → **Secure administration, applications, and infrastructure**. Under RMI/IIOP security, click **CSIv2 inbound transport**.

– Select **Security** → **Secure administration, applications, and infrastructure**. Under RMI/IIOP security, click **CSIv2 outbound transport**.

For the SOAP Java Management Extensions (JMX™) administrative transports, you can modify the SSL configurations repertoire aliases by selecting **Servers** → **Application servers** → ***server_name***. Under Server infrastructure, select **Administration** → **Administration services**. Under Additional properties, select **JMX connectors** → **SOAPConnector**. Under Additional properties, click **Custom properties**. If you want to point the sslConfig property to a new alias, click **New** and type sslConfig in the name field, and its value in the Value field.

For the Lightweight Directory Access Protocol (LDAP) SSL transport, you can modify the SSL configuration repertoire aliases by selecting **Security** → **Secure administration**, applications, and infrastructure. Under the User account repository, click the **Available realm definitions** drop-down menus and select **Standalone LDAP registry**.

8. Select **Security** → **Secure administration, applications, and infrastructure** to configure the rest of the security settings and enable security.

9. Validate the completed security configuration by clicking **OK** or **Apply**. If problems occur, they display at the top of the console page in red type.

10. If there are no validation problems, click **Save** to save the settings to a file that the server uses when it restarts. Saving writes the settings to the configuration repository.

Important: If you do not click **Apply** or **OK** in the Secure administration, applications, and infrastructure window before you click **Save**, your changes are not written to the repository. The server must be restarted for any changes to take effect when you start the administrative console.

11. Start the WebSphere Application Server administrative console.

If security is currently disabled, log in with any user ID. If security is currently enabled, log in with the Primary administrative user ID and password. The Primary administrative user ID is specified when you configure the user registry.

8.1.2 Using Java 2 security

Java 2 security provides a policy-based, fine-grain access control mechanism that increases overall system integrity by checking for permissions before allowing access to certain protected system resources. Java 2 security guards access to system resources such as file I/O, sockets, and properties. Java 2 Platform, Enterprise Edition (J2EE) security guards

access to Web resources such as servlets, Java Server Pages (JSP) files, and Enterprise Java Beans (EJB) methods.

WebSphere Application Server security includes the following technologies:

- Java 2 Security Manager
- Java Authentication and Authorization Service (JAAS)
- Java 2 Connector authentication data entries
- Common Secure Interoperability Version 2 (CSIv2) or Security Authentication Service (SAS)
- J2EE role-based authorization
- Secure Sockets Layer (SSL) configuration

Important: SAS is supported only between Version 6.0.x and previous version servers that have been federated in a Version 6.1 cell.

Because Java 2 security is relatively new, many existing or even new applications might not be prepared for the very fine-grain access control programming model that Java 2 security is capable of enforcing. Administrators need to understand the possible consequences of enabling Java 2 security if applications are not prepared for Java 2 security. Java 2 security places some new requirements on application developers and administrators.

Java 2 security for deployers and administrators

Although Java 2 security is supported, it is disabled by default. You can configure Java 2 security and administrative security independently of one another. Disabling administrative security does not disable Java 2 security automatically. You need to explicitly disable it.

If your applications or third-party libraries are not ready, having Java 2 security enabled causes problems. You can identify these problems as Java 2 security *AccessControlExceptions* in the system log or trace files. If you are not sure about the Java 2 security readiness of your applications, disable Java 2 security initially to get your application installed and verify that it is working properly.

Implications exist if Java 2 security is enabled; deployers or administrators are required to make sure that all the applications are granted the required permissions; otherwise, applications might fail to run.

For the details of default permissions granted to applications in the product, refer to the following policy files:

- app_server_root/java/jre/lib/security/java.policy

  The file represents the default permissions that are granted to all classes. The policy of this file applies to all the processes launched by the Java Virtual Machine in the WebSphere Application Server.

- app_server_root/properties/server.policy

  This policy defines the policy for the *WebSphere Application Server classes*. At present, all the server processes on the same installation share the same server.policy file. However, you can configure this file so that each server process can has a separate server.policy file. Define the policy file as the value of the java.security.policy Java system properties. For details of how to define Java system properties, refer to the Process definition section of the Manage application servers file.

The server.policy file is not a configuration file managed by the repository and the file replication service. Changes to this file are local and do not get replicated to other machines. Use the server.policy file to define Java 2 security policy for server resources.

- profile_root/config/cells/cell_name/nodes/node_name/app.policy

  Use the app.policy file (per node) or the was.policy file (per enterprise application) to define Java 2 security policy for enterprise application resources.

The policy embodied by the policy files cannot be made more restrictive because the product might not have the necessary Java 2 security doPrivileged APIs in place. The restrictive policy is the default policy. You can grant additional permissions, but you cannot make the default more restrictive because AccessControlExceptions exceptions are generated from within WebSphere Application Server. The product does not support a more restrictive policy than the default that is defined in the policy files.

Several policy files are used to define the security policy for the Java process. These policy files are static (code base is defined in the policy file) and in the default policy format provided by the IBM Developer Kit, Java Technology Edition. For enterprise application resources and utility libraries, WebSphere Application Server provides dynamic policy support. The code base is dynamically calculated based on deployment information and permissions are granted based on template policy files during runtime.

Attention: Syntax errors in the policy files cause the application server process to fail, so edit these policy files carefully. It is recommended that you use the **`policytool`** in <was_home>/java/bin to edit policy files.

8.2 User registry

WebSphere Application Server provides implementations that support multiple types of registries and repositories including the local operating system registry, a stand-alone Lightweight Directory Access Protocol (LDAP) registry, a stand-alone custom registry, and federated repositories.

With WebSphere Application Server, a user registry or a repository, such as virtual member manager, authenticates a user and retrieves information about users and groups to perform security-related functions, including authentication and authorization.

With WebSphere Application Server, a user registry or repository is used for:

- Authenticating a user using basic authentication, identity assertion, or client certificates
- Retrieving information about users and groups to perform security-related administrative functions, such as mapping users and groups to security roles

Although WebSphere Application Server supports different types of user registries, only one user registry can be active. This active registry is shared by all of the product server processes.

After configuring the registry or repository, you must specify it as the active repository. Through the administration console, you can select an available realm definition for the registry or repository from the User account repository section of the Secure administration, applications, and administration window. After selecting the registry or repository, first click **Set as current**, and then click **Apply**.

8.2.1 Custom registry interface

Before you begin this task, implement and build the UserRegistry interface. For more information about developing stand-alone custom registries, refer to 8.2.2, “Developing the UserRegistry interface for using custom registries” on page 275. The following steps are required to configure stand-alone custom registries through the administrative console:

1. Select **Security** → **Secure administration, applications, and infrastructure**.
2. Under User account repositories, click **Standalone custom registry** and click **Configure**.
3. Enter a valid user name in the Primary administrative user name field. This ID is the security server ID, which is only used for WebSphere Application Server security and is not associated with the system process that runs the server. The server calls the local operating system registry to authenticate and obtain privileged information about users by calling the native APIs in that particular registry.
4. Enter the complete location of the dot-separated class name that implements the com.ibm.websphere.security.UserRegistry interface in the Custom registry class name field. For the sample, this file name is com.ibm.websphere.security.FileRegistrySample. The file exists in the WebSphere Application Server class path, preferably in the app_server_root/lib/ext directory.

Important: The sample provided is intended to familiarize you with this feature. Do not use this sample in an actual production environment. You can find the sample file in Appendix C, “Additional material” on page 457.

5. Add your custom registry class name to the class path. We recommend that you add the Java Archive (JAR) file that contains your custom user registry implementation to the app_server_root/classes directory.
6. Optional: Select the **Ignore case for authorization** option for the authorization to perform a case insensitive check. Enabling this option is necessary only when your user registry is case insensitive and does not provide a consistent case when queried for users and groups.
7. Click **Apply** if you have any other additional properties to enter for the registry initialization.
8. Optional: Enter additional properties to initialize your implementation:
   a. Select **Custom properties** → **New**.
   b. Enter the property name and value.

      For the sample, enter the two properties shown in Table 8-1. It is assumed that the users.props file and the groups.props file are in the customer_sample directory under the product installation directory. You can place these properties in any directory that you choose and reference their locations through custom properties. However, make sure that the directory has the appropriate access permissions.

Table 8-1 Properties for customer sample

Property name	Property value
usersFile	${USER_INSTALL_ROOT}/customer_sample /users.props
groupsFile	${USER_INSTALL_ROOT}/customer_sample /groups.props

Samples of these two properties are available in Appendix C, “Additional material” on page 457.

The Description, Required, and Validation Expression® fields are not used and can remain blank.

The WebSphere Application Server Version 4 based custom user registry is migrated to the custom user registry based on the com.ibm.websphere.security.UserRegistry interface:

c. Click **Apply**.

d. Repeat this step to add other additional properties.

9. Select **Security** → **Secure administration, applications, and infrastructure**.
10. Under User account repository, click the **Available realm definitions** drop-down menu, select **Standalone custom registry**, and click **Configure**.
11. Select either the **Automatically generated server identity** or **Server identity that is stored in the repository** option. If you select the **Server identity that is stored in the repository** option, enter the following information:
    - **Server user ID or administrative user**: Specify the short name of the account that is chosen in the second step.
    - **Server user password**: Specify the password of the account that is chosen in the second step.
12. Click **OK** and complete the required steps to turn on security.

8.2.2 Developing the UserRegistry interface for using custom registries

Implementing this interface enables WebSphere Application Server security to use custom registries. This capability extends the java.rmi file. With a remote registry, you can complete this process remotely.

To implement this interface, you should complete the following methods:

Attention: The following code samples are split for illustrative purposes only.

1. Initialize the UserRegistry method, with initialize(java.util.Properties):

```
public void initialize(java.util.Properties props)
       throws CustomRegistryException,
              RemoteException;
```

   This method is called to initialize the UserRegistry method. For all the properties that are defined in the Custom User file, the initialize method retrieves the names of the registry files that contain the user and group information.

   This method is called during server bringup to initialize the registry. This method is also called when validation is performed by the administrative console, when security is on.

2. Authenticate users with checkPassword(String,String):

```
public String checkPassword(String userSecurityName, String password)
       throws PasswordCheckFailedException
              CustomRegistryException,
              RemoteException;
```

The checkPassword method is called to authenticate users when they log in using a name or user ID and a password. This method returns a string which, in most cases, is the user security name. A credential is created for the user for authorization purposes. This user name is also returned for the getCallerPrincipal enterprise bean call and the servlet calls the getUserPrincipal and getRemoteUser methods. See the getUserDisplayName method for more information if you have display names in your registry. In some situations, if you return a user other than the one who is logged in, you must verify that the user is valid in the registry.

For the FileRegistrySample.java sample file (you can get it in Appendix C, "Additional material" on page 457), the mapCertificate method gets the distinguished name (DN) from the certificate chain and makes sure it is a valid user in the registry before returning the user. For the sample, the checkPassword method checks the name and password combination in the user registry and, if they match, the method returns the user being authenticated.

This method is called for various scenarios, for example, by the administrative console to validate the user information after the user registry is initialized. This method is also called when you access protected resources in the product for authenticating the user and before proceeding with the authorization.

3. Obtain user names from X.509 certificates with mapCertificate(X509Certificate[]):

```
public String mapCertificate(X509Certificate[] cert)
       throws CertificateMapNotSupportedException,
              CertificateMapFailedException,
              CustomRegistryException,
              RemoteException;
```

The **`mapCertificate`** method is called to obtain a user name from an X.509 certificate chain that is supplied by the browser. The complete certificate chain is passed to this method and the implementation can validate the chain if needed and get the user information. A credential is created for this user for authorization purposes. If browser certificates are not supported in your configuration, you can create the `CertificateMapNotSupportedException` exception. The consequence of not supporting certificates is an authentication failure if the challenge type is certificates, even if valid certificates are in the browser.

This method is called when certificates are provided for authentication. For Web applications, when the authentication constraints are set to `client-cert` in the web.xml file of the application, this method is called to map a certificate to a valid user in the registry. For Java clients, this method is called to map the client certificates in the transport layer, when using transport layer authentication. When the identity assertion token, using the CSIv2 authentication protocol, is set to contain certificates, this method is called to map the certificates to a valid user.

4. Obtain the security realm name with getRealm:

```
public String getRealm()
       throws CustomRegistryException,
       RemoteException;
```

The getRealm method is called to get the name of the security realm. The name of the realm identifies the security domain for which the registry authenticates users. If this method returns a null value, a customRealm default name is used.

For the FileRegistrySample.java sample file, the getRealm method returns the customRealm string. One of the calls to this method occurs when the user registry information is validated.

5. Obtain the list of users from the registry with getUsers(String,int):

```
public Result getUsers(String pattern, int limit)
       throws CustomRegistryException,
       RemoteException;
```

   The getUsers method returns the list of users from the registry. The names of users depend on the pattern parameter. The number of users are limited by the limit parameter. In a registry that has many users, getting all the users is not practical. So the limit parameter is introduced to limit the number of users retrieved from the registry. A limit of zero (0) returned all the users that match the pattern and might cause problems for large registries. Use this limit with care.

   The custom registry implementations are expected to support at least the wildcard search (*). For example, a pattern of asterisk (*) returns all the users and a pattern of (a*) returns the users starting with a.

   The return parameter is an object with a com.ibm.websphere.security.Result type. This object contains two attributes: a java.util.List and a java.lang.boolean attribute. The list contains the users that are returned and the Boolean flag indicates if more users are available in the user registry for the search pattern. This Boolean flag is used to indicate to the client whether more users are available in the registry.

   In the FileRegistrySample.java sample file, the getUsers method retrieves the required number of users from the user registry and sets them as a list in the Result object. To discover if more users are presented than requested, the sample gets one more user than requested and if it finds the additional user, it sets the Boolean flag to true. For pattern matching, the match method in the RegExpSample class is used, which supports wildcard characters, such as the asterisk (*) and the question mark (?).

   This method is called by the administrative console to add users to roles in the various map-users-to-roles windows. The administrative console uses the Boolean set in the Result object to indicate that more entries matching the pattern are available in the user registry.

   This method specifies to take the pattern and limit parameters. The return is a Result object, which consists of the list and a flag that indicates if more entries exist. When the list returns, use the Result.setList(List) method to set the list in the Result object. If more entries exist than requested in the limit parameter, set the Boolean attribute to true in the result object, using the Result.setHasMore method. The default for the Boolean attribute in the result object is false.

6. Obtain the display name of a user with getUserDisplayName(String):

```
public String getUserDisplayName(String userSecurityName)
       throws EntryNotFoundException,
              CustomRegistryException,
              RemoteException;
```

   The getUserDisplayName method returns a display name for a user, if one exists. The display name is an optional string that describes the user that you can set in some registries. This descriptive name is for the user and does not have to be unique in the registry.

   If you do not need display names in your registry, return null or an empty string for this method.

If the display names are not the same as the security name for any user, the display names are returned for the previously mentioned enterprise beans and servlet methods. Returning display names for these methods might become problematic in some situations because the display names might not be unique in the user registry. Avoid this problem by changing the default behavior to return the user security name instead of the user display name in this version of the product.

In the FileRegistrySample.java sample file, this method returns the display name of the user whose name matches the user name that is provided. If the display name does not exist, this method returns an empty string.

This method can be called by the product to present the display names in the administrative console, or by using the command line and the **wsadmin** tool. Use this method for display purposes only.

7. Obtain the unique ID of a user with getUniqueId(String):

```
public String getUniqueUserId(String userSecurityName)
       throws EntryNotFoundException,
              CustomRegistryException,
              RemoteException;
```

   This method returns the unique ID of the user, given the security name.

   In the FileRegistrySample.java sample file, this method returns the uniqueUserId value of the user whose name matches the supplied name. This method is called when forming a credential for a user and also when creating the authorization table for the application.

8. Obtain the security name of a user with getUserSecurityName(String):

```
public String getUserSecurityName(String uniqueUserId)
       throws EntryNotFoundException,
              CustomRegistryException,
              RemoteException;
```

   This method returns the security name of a user given the unique ID. In the FileRegistrySample.java sample file, this method returns the security name of the user whose unique ID matches the supplied ID.

   This method is called to make sure a valid user exists for a given uniqueUserId. This method is called to get the security name of the user when the uniqueUserId is obtained from a token.

9. Check whether a given user is a valid user in the registry with isValidUser(String):

```
public boolean isValidUser(String userSecurityName)
       throws CustomRegistryException,
       RemoteException;
```

   This method indicates whether the given user is a valid user in the registry.

   In the FileRegistrySample.java sample file, this method returns true if the user is found in the registry; otherwise, this method returns false. This method is primarily called in situations where knowing if the user exists in the directory prevents problems later. For example, in the mapCertificate call, when the name is obtained from the certificate if the user is not found as a valid user in the user registry, you can avoid trying to create the credential for the user.

10. Return the list of groups from the user registry with getGroups(String,int):

```
public Result getGroups(String pattern, int limit)
       throws CustomRegistryException,
       RemoteException;
```

The getGroups method returns the list of groups from the user registry. The names of groups depend on the pattern parameter. The number of groups is limited by the limit parameter. In a registry that has many groups, getting all the groups is not practical. So, the limit parameter is introduced to limit the number of groups retrieved from the user registry. A limit of zero (0) implies returning all the groups that match the pattern and can cause problems for large user registries. Use this limit with care. The custom registry implementations are expected to support at least the wildcard search (*). For example, a pattern of asterisk (*) returns all the users and a pattern of (a*) returns the users starting with a.

The return parameter is an object of the com.ibm.websphere.security.Result type. This object contains the java.util.List and java.lang.boolean attributes. The list contains the groups that are returned and the Boolean flag indicates whether more groups are available in the user registry for the pattern searched. This Boolean flag is used to indicate to the client if more groups are available in the registry.

In the FileRegistrySample.java sample file, the getUsers method retrieves the required number of groups from the user registry and sets them as a list in the Result object. To discover if more groups are presented than requested, the sample gets one more user than requested and if it finds the additional user, and it sets the Boolean flag to true. For pattern matching, the match method in the RegExpSample class is used, which supports the asterisk (*) and question mark (?) characters.

This method is called by the administrative console to add groups to roles in the various map-groups-to-roles windows. The administrative console uses the Boolean set in the Result object to indicate that more entries matching the pattern are available in the user registry.

The return is a Result object, which consists of the list and a flag that indicates whether more entries exist. Use the Result.setList(List) method to set the list in the Result object. If more entries exist than requested in the limit parameter, set the Boolean attribute to true in the Result object using the Result.setHasMore method. The default for the Boolean attribute in the Result object is false.

11. Obtain the display name of a group with getGroupDisplayName(String):

```
public String getGroupDisplayName(String groupSecurityName)
       throws EntryNotFoundException,
              CustomRegistryException,
       RemoteException;
```

The getGroupDisplayName method returns a display name for a group if one exists. The display name is an optional string that describes the group that you can set in some user registries. This name is a descriptive name for the group and does not have to be unique in the registry. If you do not need to have display names for groups in your registry, return null or an empty string for this method.

In the FileRegistrySample.java sample file, this method returns the display name of the group whose name matches the group name that is provided. If the display name does not exist, this method returns an empty string.

The product can call this method to present the display names in the administrative console or through the command line using the **wsadmin** tool. This method is used for display purposes only.

12. Obtain the unique ID of a group with getUniqueGroupId(String):

```
public String getUniqueGroupId(String groupSecurityName)
       throws EntryNotFoundException,
              CustomRegistryException,
        RemoteException;
```

This method returns the unique ID of the group that is given the security name.

In the FileRegistrySample.java sample file, this method returns the security name of the group whose unique ID matches the supplied ID. This method verifies that a valid group exists for a given uniqueGroupId ID.

13. Obtain the unique IDs of all groups to which a user belongs with getUniqueGroupIds(String):

```
public List getUniqueGroupIds(String uniqueUserId)
       throws EntryNotFoundException,
       CustomRegistryException,
       RemoteException;
```

This method returns the unique IDs of all the groups to which a user belongs.

In the FileRegistrySample.java sample file, this method returns the unique ID of all the groups that contain this uniqueUserID ID. This method is called when creating the credential for the user. As part of creating the credential, all the groupUniqueIds IDs in which the user belongs are collected and put in the credential for authorization purposes when groups are given access to a resource.

14. Obtain the security name of a group with getGroupSecurityName(String):

```
public String getGroupSecurityName(String uniqueGroupId)
           tthrows EntryNotFoundException,
           CustomRegistryException,
           RemoteException;
```

This method returns the security name of a group given its unique ID.

In the FileRegistrySample.java sample file, this method returns the security name of the group whose unique ID matches the supplied ID. This method verifies that a valid group exists for a given uniqueGroupId ID.

15. Determine whether a group is a valid group in the registry with isValidGroup(String):

```
public Boolean isValidGroup(String groupSecurityName)
       throws CustomRegistryException,
              RemoteException;
```

This method indicates if the given group is a valid group in the registry.

In the FileRegistrySample.java sample file, this method returns true if the group is found in the registry; otherwise, the method returns false. This method can be used in situations where knowing whether the group exists in the directory might prevent problems later.

16. Obtain all groups to which a user belongs with getGroupsForUser(String):

```
public List getGroupsForUser(String userSecurityName)
      throws EntryNotFoundException,
             CustomRegistryException,
             RemoteException;
```

This method returns all the groups to which a user belongs whose name matches the supplied name. This method is similar to the getUniqueGroupIds method with the exception that the security names are used instead of the unique IDs.

In the FileRegistrySample.java sample file, this method returns all the group security names that contain the userSecurityName name.

This method is called by the administrative console or the scripting tool to verify that the users entered for the RunAs roles are already part of that role in the users and groups-to-role mapping. This check is required to ensure that a user cannot be added to a RunAs role unless that user is assigned to the role in the users and groups-to-role mapping either directly or indirectly through a group that contains this user. Because a

group in which the user belongs can be part of the role in the users and groups-to-role mapping, this method is called to check if any of the groups that this user belongs to mapped to that role.

17. Retrieve users from a specified group with getUsersForGroup(String,int):

```
public Result getUsersForGroup(String groupSecurityName, int limit)
     throws NotImplementedException,
            EntryNotFoundException,
            CustomRegistryException,
            RemoteException;
```

This method retrieves users from the specified group. The number of users returned is limited by the limit parameter. A limit of zero (0) indicates returning all of the users in that group. This method is not directly called by the WebSphere Application Server security component. However, this method can be called by other components. In rare situations, if you are working with a user registry where getting all the users from any of your groups is not practical, you can create the NotImplementedException exception for the particular groups. In this case, verify that if the process choreographer is installed the staff assignments are not modeled using these particular groups. If no concern exists about returning the users from groups in the user registry, we recommend that you do not create the NotImplemented exception when implementing this method.

The return parameter is an object with a com.ibm.websphere.security.Result type. This object contains the java.util.List and java.lang.boolean attributes. The list contains the users that are returned and the Boolean flag, which indicates whether more users are available in the user registry for the search pattern. This Boolean flag indicates to the client whether users are available in the user registry.

In the example, this method gets one user more than the requested number of users for a group, if the limit parameter is not set to zero (0). If the method succeeds in getting one more user, the Boolean flag is set to true.

This method can create the NotImplementedException exception in situations where it is not practical to get the requested set of users. However, create this exception in rare situations, as other components can be affected. It returns a Result object, which consists of the list and a flag that indicates whether more entries exist. When the list is returned, use the Result.setList(List) method to set the list in the Result object. If more entries than requested are in the limit parameter, set the Boolean attribute to true in the Result object using Result.setHasMore method. The default for the Boolean attribute in the Result object is false.

18. Implement the createCredential(String) method:

```
public com.ibm.websphere.security.cred.WSCredential createCredential(String
userSecurityName)
 throws NotImplementedException,
            EntryNotFoundException,
            CustomRegistryException,
            RemoteException;
```

In this release of the product, the createCredential method is not called. You can return null. In the example, a null value is returned.

Tip: You can find the FileRegistrySample.java file and the properties files users.props and groups.props in Appendix C, “Additional material” on page 457.

8.3 Secure communications using Secure Sockets Layer

The Secure Sockets Layer (SSL) protocol provides transport layer security, including authenticity, data signing, and data encryption to ensure a secure connection between a client and server that uses WebSphere Application Server. The foundation technology for SSL is public key cryptography, which guarantees that when an entity encrypts data using its private key, only entities with the corresponding public key can decrypt that data.

WebSphere Application Server uses Java Secure Sockets Extension (JSSE) as the SSL implementation for secure connections. JSSE is part of the Java 2 Standard Edition (J2SE) specification and is included in the IBM implementation of the Java Runtime Extension (JRE). JSSE handles the handshake negotiation and protection capabilities that are provided by SSL to ensure secure connectivity exists across most protocols. JSSE relies on X.509 certificate-based asymmetric key pairs for secure connection protection and some data encryption. Key pairs effectively encrypt session-based secret keys that encrypt larger blocks of data. The SSL implementation manages the X.509 certificates.

SSL configurations

An SSL configuration comprises a set of configuration attributes that you can associate with an endpoint or set of endpoints in the WebSphere Application Server topology. The SSL configuration enables you to create an SSLContext object, which is the fundamental JSSE object that the server uses to obtain SSL socket factories. You can manage the following configuration attributes:

- An alias for the SSLContext object
- A handshake protocol version
- A keystore reference
- A truststore reference
- A key manager
- One or more trust managers
- A security level selection of a cipher suite grouping or a specific cipher suite list
- A certificate alias choice for client and server connections

To understand the specifics of each SSL configuration attribute, see 8.3.1, “Secure Sockets Layer configurations” on page 286.

Selecting SSL configurations

In previous releases of WebSphere Application Server, you can reference an SSL configuration only by selecting the SSL configuration alias directly. Each secure endpoint was denoted by an alias attribute that references a valid SSL configuration within a repertoire of SSL configurations. When you made a single configuration change, you had to re-configure many alias references across the various processes. Although the current release still supports direct selection, this approach is no longer recommended.

The current release provides improved capabilities for managing SSL configurations and more flexibility when you select SSL configurations. In this release, you can select from the following approaches.

Programmatic selection

You can set an SSL configuration on the running thread prior to an outbound connection. WebSphere Application Server ensures that most system protocols, including Internet Inter-ORB Protocol (IIOP), Java Message Service (JMS), Hyper Text Transfer Protocol (HTTP), and Lightweight Directory Access Protocol (LDAP), accept the configuration.

Dynamic selection

You can associate an SSL configuration dynamically with a specific target host, port, or outbound protocol by using a predefined selection criteria. When it establishes the connection, WebSphere Application Server checks to see if the target host and port match a predefined criteria that includes the domain portion of the host. Additionally, you can predefine the protocol for a specific outbound SSL configuration and certificate alias selection.

Direct selection

You can select an SSL configuration by using a specific alias, as in past releases. This method of selection is maintained for backwards compatibility because many applications and processes rely on alias references.

Scope selection

You can associate an SSL configuration and its certificate alias, which is located in the keystore associated with that SSL configuration, with a WebSphere Application Server management scope. This approach is recommended to manage SSL configurations centrally. You can manage endpoints more efficiently because they are located in one topology view of the cell. The inheritance relationship between scopes reduces the number of SSL configuration assignments that you must set.

Each time you associate an SSL configuration with a cell scope, the node scope within the cell automatically inherits the configuration properties. However, when you assign an SSL configuration to a node, the node configuration overrides the configuration that the node inherits from the cell. Similarly, all of the application servers for a node automatically inherit the SSL configuration for that node unless you override these assignments. Unless you override a specific configuration, the topology relies on the rules of inheritance from the cell level down to the endpoint level for each application server.

The topology view displays an inbound tree and outbound tree. You can make different SSL configuration selections for each side of the SSL connection based on what that server connects to as an outbound connection and what the server connects to as an inbound connection.

The runtime uses an order of precedence for determining which SSL configuration to choose because you have many ways to select SSL configurations. Consider the following order of precedence when you select a configuration approach:

1. Programmatic selection. If an application sets an SSL configuration on the running thread using the com.ibm.websphere.ssl.JSSEHelper application programming interface (API), the SSL configuration is guaranteed the highest precedence.
2. Dynamic selection criteria for outbound host and port or protocol.
3. Direct selection.
4. Scope selection. Scope inheritance guarantees that the endpoint that you select is associated with an SSL configuration and is inherited by every scope beneath it that does not override this selection.

Default self-signed certificate configuration

By default, WebSphere Application Server creates a unique self-signed certificate for each node. WebSphere Application Server no longer relies on the default or dummy certificate that is shipped with the product. The key.p12 default keystore and the trust.p12 truststore are stored in the configuration repository within the node directory.

When you federate a base application server, the following situations occur: the keystore and truststore are included, and the signer certificate is added to the deployment manager common truststore, which is located in the cell directory of the configuration repository.

All of the nodes put their signer certificates in this common truststore (trust.p12). After you federate a node, the default SSL configuration is automatically modified to point to the common truststore, which is located in the cell directory. The node can now communicate with all other servers in the cell.

All default SSL configurations contain a keystore with the name suffix DefaultKeyStore and a truststore with the name suffix DefaultTrustStore. These default suffixes instruct the WebSphere Application Server runtime to add the signer of the personal certificate to the common truststore. If a keystore name does not end with DefaultKeyStore, the keystore signer certificates are not added to the common truststore when you federate the server. You can change the default SSL configuration, but you must ensure that the correct trust is established for administrative connections, among others.

For more information, refer to *IBM WebSphere Application Server V6.1 Security Handbook*, SG24-6316.

Certificate expiration monitoring

Certificate monitoring ensures that the self-signed certificate for each node is not allowed to expire. The certificate expiration monitoring function issues a warning before certificates and signers are set to expire. Those certificates and signers that are located in keystores managed by the WebSphere Application Server configuration can be monitored. You can configure the expiration monitor to automatically replace a self-signed certificate with a new self-signed certificate that is based upon the same data that is used for the initial creation. The monitor can also automatically replace old signers with the signers from the new self-signed certificates in keystores that are managed by WebSphere Application Server. The existing signer exchanges that occurred by the runtime during federation and by administration are preserved.

WebSphere Application Server clients: signer-exchange requirements

A new self-signed certificate is generated for each node during its initial startup. To ensure trust, clients must be given these generated signers to establish a connection. Several enhancements in the current release make this process simpler. You can gain access to the signer certificates of various nodes to which the client must connect with any one of the following options:

- A signer exchange prompt enables you to import signer certificates that are not yet present in the truststores during a connection to a server. By default, this prompt is enabled for administrative connections and can be enabled for any client SSL configuration. When this prompt is enabled, any connection that is made to a server where the signer is not already present offers the signer of the server, along with the certificate information, a Secure Hash Algorithm (SHA) digest of the certificate for verification. The user is given a choice whether to accept these credentials. If the credentials are accepted, the signer is added to the truststore of the client until the signer is explicitly removed. The signer exchange prompt does not occur again when connecting to the same server unless the personal certificate changes.

Attention: It is unsafe to trust a signer exchange prompt without verifying the SHA digest. An unverified prompt can originate from a browser when a certificate is not trusted.

- You can run a retrieveSigners administrative script from a client prior to making connections to servers. To download signers, no administrative authority is required. To upload signers, you must have Administrator role authority. The script downloads all of the signers from a specified server truststore into the specified client truststore and can be called to download only a specific alias from a truststore. You can also call the script to upload signers to server truststores. When you select the CellDefaultTrustStore truststore as the specified server truststore and common truststore for a cell, all of the signers for that cell are downloaded to the specified client truststore, which is typically ClientDefaultTrustStore.
- You can physically distribute to clients the trust.p12 common truststore that is located in the cell directory of the configuration repository. When doing this distribution, however, you must ensure that the correct password has been specified in the ssl.client.props client SSL configuration file. The default password for this truststore is WebAS. Change the default password prior to distribution. Physical distribution is not as effective as the previous options. When changes are made to the personal certificates on the server, automated exchange can fail.

Dynamic SSL configuration changes

The SSL runtime for WebSphere Application Server maintains listeners for most SSL connections. A change to the SSL configuration causes the inbound connection listeners to create a new SSLContext object. Existing connections continue to use the current SSLContext object. Outbound connections automatically use the new configuration properties when they are attempted.

Make dynamic changes to the SSL configuration during off-peak hours to reduce the possibility of timing-related problems and to prevent the possibility of the server starting again. If you enable the runtime to accept dynamic changes, then change the SSL configuration and save the security.xml file. Your changes take effect when the new security.xml file reaches each node.

Note: If configuration changes cause SSL handshake failures, administrative connectivity failures can also occur, which can lead to outages. In this case, you must re-configure the SSL connections then perform manual node synchronization to correct the problem. You must carefully complete any dynamic changes. We highly recommend that you perform changes to SSL configurations on a test environment prior to making the same changes to a production system

Built-in certificate management

Certificate management that is comparable to iKeyMan functionality is now integrated into the keystore management windows of the administrative console. Use built-in certificate management to manage personal certificates, certificate requests, and signer certificates that are located in keystores. Additionally, you can remotely manage keystores. For example, you can manage a file-based keystore that is located outside the configuration repository on any node from the deployment manager. You also can remotely manage hardware cryptographic keystores from the deployment manager.

With built-in certificate management, you can replace a self-signed certificate along with all of the signer certificates scattered across many truststores and retrieve a signer from a remote port by connecting to the remote SSL host and port and intercepting the signer during the handshake. The certificate is first validated according to the certificate SHA digest, then the administrator must accept the validated certificate before it can be placed into a truststore.

When you make a certificate request, you can send it to a certificate authority (CA). When the certificate is returned, you can accept it within the administrative console.

Tip: While iKeyMan functionality still ships with WebSphere Application Server, configure keystores from the administrative console using the built-in certificate management functionality. iKeyMan is still an option when it is not convenient to use the administrative console. For more information, see 8.3.4, “Certificate management using iKeyman” on page 296.

AdminTask configuration management

The SSL configuration management windows in the administrative console rely primarily on administrative tasks, which are maintained and enhanced to support the administrative console function. You can use **`wsadmin`** commands from a Java console prompt to automate the management of keystores, SSL configurations, certificates, and so on.

8.3.1 Secure Sockets Layer configurations

Secure Sockets Layer (SSL) configurations contain attributes that enable you to control the behavior of both the client and the server SSL endpoints. You can assign SSL configurations to have specific management scopes. The scope that an SSL configuration accede to depends upon whether you create it using a cell, node, server, or endpoint link in the configuration topology.

When you create an SSL configuration, you can set the following SSL connection attributes:

- Keystore
- Default client certificate for outbound connections
- Default server certificate for inbound connections
- Truststore
- Key manager for selecting a certificate
- Trust manager or managers for establishing trust during the handshake
- Handshaking protocol
- Ciphers for negotiating the handshake
- Client authentication support and requirements

You can manage an SSL configuration using any of the following methods:

- Central management selection
- Direct reference selection
- Dynamic outbound connection selection
- Programmatic selection

Using the administrative console, you can manage all of the SSL configurations for WebSphere Application Server. From the administrative console, select **Security** → **SSL certificates and key management** → **Manage endpoint security configurations** → **Inbound** → **Outbound** → ***SSL_configuration***. You can view an SSL configuration at the level it was created and in the inherited scope below that point in the topology. If you want the entire cell to view an SSL configuration, you must create the configuration at the cell level in the topology.

SSL configuration in the security.xml file

The attributes defining an SSL configuration repertoire entry for a specific management scope are stored in the *security.xml* file. The scope determines the point at which other levels in the cell topology can see the configuration, as shown in Example 8-1.

Example 8-1 security.xml file

```
<repertoire xmi:id="SSLConfig_1" alias="NodeDefaultSSLSettings"
managementScope="ManagementScope_1" type="JSSE">
<setting xmi:id="SecureSocketLayer_1" clientAuthentication="false"
clientAuthenticationSupported="false" securityLevel="HIGH" enabledCiphers=""
jsseProvider="IBMJSSE2" sslProtocol="SSL_TLS" keyStore="KeyStore_1"
trustStore="KeyStore_2" trustManager="TrustManager_1" keyManager="KeyManager_1"
clientKeyAlias="default" serverKeyAlias="default"/>
</repertoire>
```

The SSL configuration attributes from Example 8-1 are described in Table 8-2.

Table 8-2 security.xml attributes

security.xml attribute	Description	Default	Associated SSL property
xmi:id	The xml:id attribute represents the unique identifier for this XML entry and determines how the SSL configuration is linked to other XML objects, such as SSLConfigGroup. This system-defined value must be unique.	The administrative configuration service defines the default value.	None. This value is used only for XML associations.
alias	The alias attribute defines the name of the SSL configuration. Direct selection uses the alias attribute and the node is not prefixed to the alias. Rather, the management scope takes care of ensuring that the name is unique within the scope.	The default is NodeDefaultSSLSettings.	com.ibm.ssl.alias
managementScope	The managementScope attribute defines the management scope for the SSL configuration and determines the visibility of the SSL configuration at runtime.	The default scope is the node.	The managementScope attribute is not mapped to an SSL property. However, it confirms whether or not the SSL configuration is associated with a process.

security.xml attribute	Description	Default	Associated SSL property
type	The type attribute defines the Java Secure Socket Extension (JSSE) or System Secure Sockets Layer (SSSL) configuration option. JSSE is the SSL configuration type for most secure communications within WebSphere Application Server.	The default is JSSE.	com.ibm.ssl.sslType.
clientAuthentication	The clientAuthentication attribute determines whether SSL client authentication is required.	The default is false.	com.ibm.ssl.clientAuthentication.
clientAuthenticationSupported	The clientAuthenticationSupported attribute determines whether SSL client authentication is supported. The client does not have to supply a client certificate if it does not have a client certificate. Attention: When you set the clientAuthentication attribute to true, you override the value that is set for the clientAuthenticationSupported attribute	The default is false.	com.ibm.ssl.client.AuthenticationSupported.

security.xml attribute	Description	Default	Associated SSL property
securityLevel	The securityLevel attribute determines the cipher suite group. Valid values include HIGH (128-bit ciphers), MEDIUM (40-bit ciphers), LOW (for all ciphers without encryption), and CUSTOM (if the cipher suite group is customized). When you set the enabledCiphers attribute with a specific list of ciphers, the system ignores this attribute.	The default is HIGH.	com.ibm.ssl.securityLevel.
enabledCiphers	You can set the enabledCiphers attribute to specify a unique list of cipher suites. Separate each cipher suite in the list with a space.	The default is the securityLevel attribute for cipher suite selection.	com.ibm.ssl.enabledCipherSuites.
jsseProvider	The jsseProvider attribute defines a specific JSSE provider.	The default is IBMJSSE2.	com.ibm.ssl.contextProvider.
sslProtocol	The sslProtocol attribute defines the SSL handshake protocol. Valid options include: SSLv2 (client-side only), SSLv3, SSL, SSL_TLS, TLSv1, and TLS values. The SSL option includes SSLv2 and SSLv3 values. The TLS option includes the TLSv1 value. SSL_TLS, which is the most interoperable protocol, includes all these values and defaults to a Transport Layer Security (TLS) handshake.	The default is SSL_TLS.	com.ibm.ssl.protocol.

security.xml attribute	Description	Default	Associated SSL property
keyStore	The keyStore attribute defines the keystore and attributes of the keyStore instance that the SSL configuration uses for key selection.	The default is NodeDefaultKeyStore.	java.security.KeyStore.
trustStore	The trustStore attribute defines the key store that the SSL configuration uses for certificate signing verification.	The default is NodeDefaultTrustStore.	A trustStore is a logical JSSE term. It signifies a key store that contains signer certificates. Signer certificates validate certificates that are sent to WebSphere Application Server during an SSL handshake.
keyManager	The keyManager attribute defines the key manager that WebSphere Application Server uses to select keys from a key store. A JSSE key manager controls the javax.net.ssl.X509KeyManager interface. A custom key manager controls the javax.net.ssl.X509KeyManager and the com.ibm.wsspi.ssl.KeyManagerExtendedInfo interfaces. The com.ibm.wsspi.ssl.KeyManagerExtendedInfo interface provides more information from WebSphere Application Server.	The default is IbmX509.	com.ibm.ssl.keyManager defines a well-known key manager and accepts the algorithm and algorithm|provider formats, for example, IbmX509 and IbmX509|IBMJSSE2. com.ibm.ssl.customKeyManager defines a custom key manager and takes precedence over the other keyManager properties. This class must implement javax.net.ssl.X509KeyManager and can implement com.ibm.wsspi.ssl.KeyManagerExtendedInfo.

security.xml attribute	Description	Default	Associated SSL property
trustManager	The trustManager determines which trust manager or list of trust managers to use for determining whether to trust the peer side of the connection. A JSSE trust manager implements the javax.net.ssl.X509TrustManager interface. A custom trust manager might also implement com.ibm.wsspi.ssl.TrustManagerExtendedInfo interface to get more information from the WebSphere Application Server environment.	The default is IbmX509. You can specify the IbmPKIX trust manager for certificate revocation list (CRL) verification when the certificate contains a CRL distribution point.	com.ibm.ssl.trustManager defines a well-known trust manager, which is required for most handshake situations. com.ibm.ssl.trustManager performs certificate expiration checking and signature validation. You can define com.ibm.ssl.customTrustManagers with additional custom trust managers that are called during an SSL handshake. Separate additional trust managers with the vertical bar (|) character.

8.3.2 Keystore configurations

Use keystore configurations to define how the runtime for WebSphere Application Server loads and manages keystore types for Secure Sockets Layer (SSL) configurations.

By default, the java.security.Security.getAlgorithms("KeyStore") attribute does not display a predefined list of keystore types in the administrative console. Instead, WebSphere Application Server retrieves all of the KeyStore types that can be referenced by the java.security.KeyStore object, including hardware cryptographic, IBM Java Cryptography Extension (IBMJCE), and Java-based content management system (CMS)-provider keystores. If you specify a keystore provider in the java.security file or add it to the provider list programmatically, WebSphere Application Sever also retrieves custom keystores. The retrieval list depends upon the java.security configuration for that platform and process.

Hardware cryptographic keystores

In certain environments, you can use the hardware cryptographic card for hardware acceleration. Either set the useForAcceleration attribute to true or set the com.ibm.ssl.keyStoreUseForAcceleration custom property. When you set the attribute to true, you are not required to configure a password. However, you cannot use the device to store keys.

8.3.3 Secure Sockets Layer node, application server, and cluster isolation

Secure Sockets Layer (SSL) enables you to ensure that any client that attempts to connect to a server during the handshake first performs server authentication. Using SSL configurations at the node, application server, and cluster scopes, you can isolate communication between servers that should not be allowed to communicate with each other over secure ports.

Before you attempt to isolate communications controlled by WebSphere Application Server, you should have a keen understanding of the deployment topology and application environment. To isolate a node, application server, or cluster, you must be able to control the signers that are contained in the trust stores that are associated with the SSL configuration. When the client does not contain the server signer, it cannot establish a connection to the server. If you use self-signed certificates, the server that created the personal certificate controls the signer, although you do have to manage the self-signed certificates. If you obtain certificates from a certificate authority (CA), you must obtain multiple CA signers because all of the servers can connect to each other if they share the same signer.

Authenticating only the server-side of a connection is not adequate protection when you need to isolate a server. Any client can obtain a signer certificate for the server and add it to its trust store. SSL client authentication must also be enabled between servers so that the server can control its connections by deciding which client certificates it can trust.

Isolation also requires that you use centrally managed SSL configurations for all or most endpoints in the cell. Centrally managed configurations can be scoped, unlike direct or end point configuration selection, and they enable you to create SSL configurations, key stores, and trust stores at a particular scope. Because of the inheritance hierarchy of WebSphere Application Server cells, if you select only the properties that you need for an SSL configuration, only these properties are defined at your selected scope or lower. For example, if you configure at the node scope, your configuration applies to the application server and individual endpoint scopes below the node scope.

When you configure the key stores, which contain cryptographic keys, you must work at the same scope at which you define the SSL configuration and not at a higher scope. For example, if you create a key store that contains a certificate whose host name is part of the distinguished name (DN), then store that keystore in the node directory of the configuration repository. If you decide to create a certificate for the application server, then store that keystore on the application server in the application server directory.

When you configure the trust stores, which control trust decisions on the server, you must consider how much you want to isolate the application servers. You cannot isolate the application servers from the node agents or the deployment manager. However, you can configure the SOAP connector endpoints with the same personal certificate or to share trust. Naming persistence requires IIOP connections when they pass through the deployment manager. Because application servers always connect to the node agents when the server starts, the IIOP protocol requires that WebSphere Application Server establish trust between the application servers and the node agents.

Establishing node SSL isolation

By default, WebSphere Application Server uses a single self-signed certificate for each node so you can isolate nodes easily. A common trust store, which is located in the cell directory of the configuration repository, contains all of the signers for each node that is federated into the cell. After federation, each cell process trusts all of the other cell processes because every SSL configuration references the common trust store.

You can modify the default configuration so that each node has its own trust store, and every application server on the node trusts only the node agent that uses the same personal certificate. You must also add the signer to the node trust store so that WebSphere Application Server can establish trust with the deployment manager. To isolate the node, ensure that the following conditions are met:

- The deployment manager must initiate connections to any process.
- The node agent must initiate connections to the deployment manager and its own application servers.

- The application servers must initiate connections to the applications servers on the same node, to its own node agent, and the deployment manager

As shown in Figure 8-1, Node Agent A contains a key.p12 keystore and a trust.p12 trust store at the node level of the configuration repository for node A.

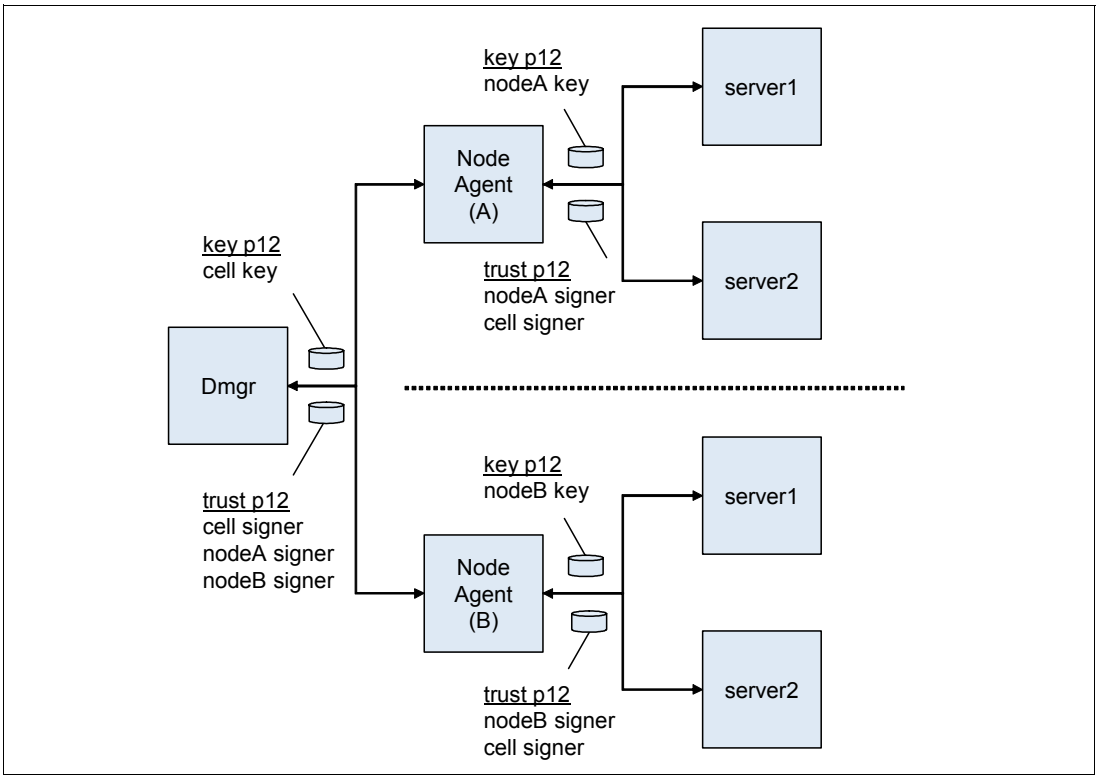


Figure 8-1 SSL node isolation

When you associate an SSL configuration with this keystore and truststore, you break the link with the cell-scoped trust store. To isolate the node completely, repeat this process for each node in the cell. WebSphere Application Server SSL configurations override the cell scope and use the node scope instead so that each process at this scope uses the SSL configuration and certificate alias that you selected at this scope. You establish proper administrative trust by ensuring that nodeA signer is in the common trust store and the cell signer is in the nodeA trust store. The same logic applies to node B as well.

Establishing application server SSL isolation

Isolating application server processes from one another is more challenging than isolating nodes. You must consider the following application design and topology conditions:

- An application server process might need to communicate with the node agent and deployment manager.
- Isolating application server processes from each other might disable single sign-on capabilities for horizontal propagation.

If you configure outbound SSL configurations dynamically, you can accommodate these conditions. When you define a specific outbound protocol, target host, and port for each different SSL configuration, you can override the scoped configuration.

Figure 8-2 shows how you might isolate an application server completely, although in practice this approach would be more complicated.

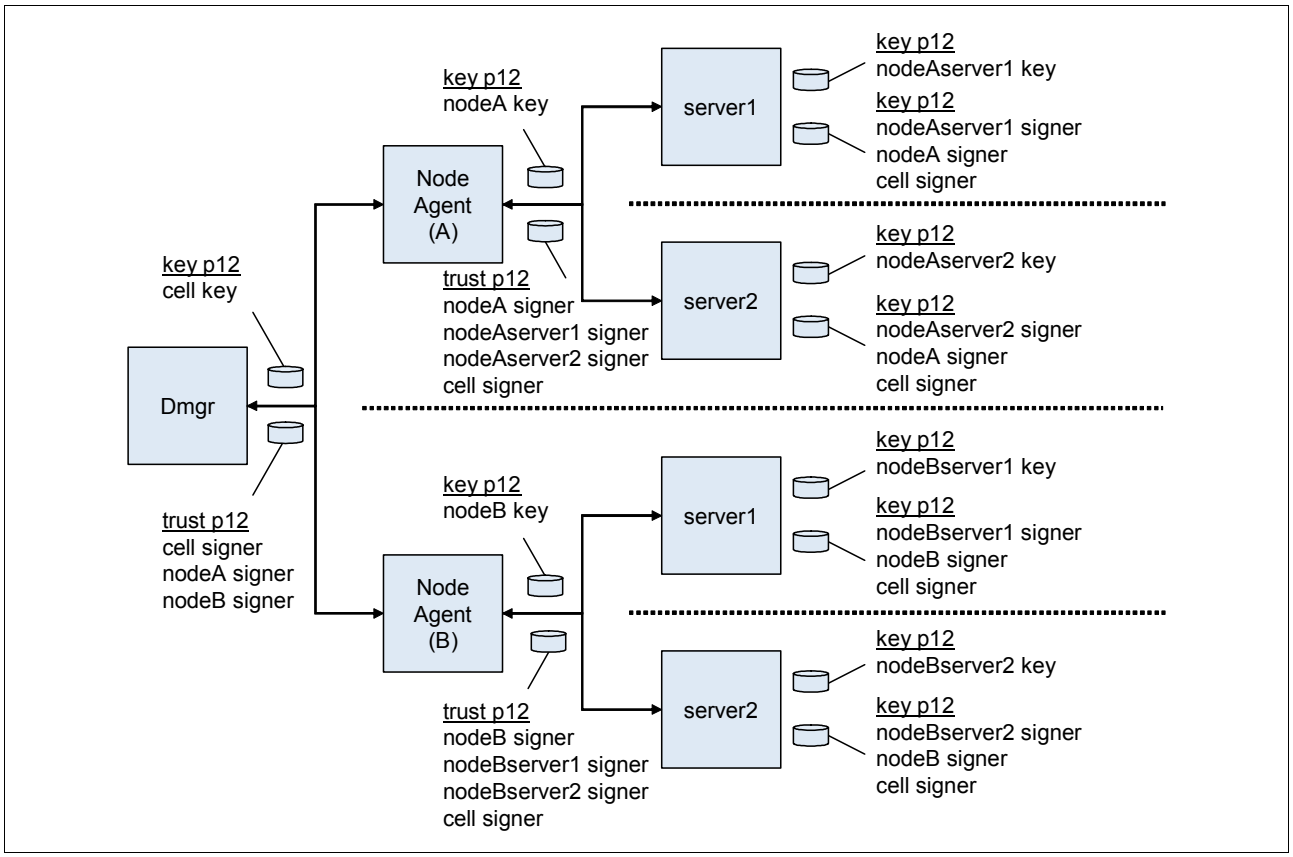


Figure 8-2 SSL application server isolation

The dynamic configuration enables server1 on Node A to communicate with server 1 on Node B only over IIOP. The dynamic outbound rule is IIOP,nodeBhostname,*.

Establishing cluster SSL isolation

You can configure application servers into clusters instead of scoping them centrally at the node or dynamically at the server to establish cluster SSL isolation. While clustered servers can communicate with each other, application servers outside of the cluster cannot communicate with the cluster, thus isolating the clustered servers. For example, you might need to separate applications from different departments while maintaining a basic level of trust among the clustered servers. Using the dynamic outbound SSL configuration method described for servers above, you can easily extend the isolated cluster as needed.

Figure 8-3 on page 295 shows a sample cluster configuration where cluster 1 contains a key.p12 with its own self-signed certificate, and a trust.p12 that is located in the config/cells/<cellname>/clusters/<clustername> directory.

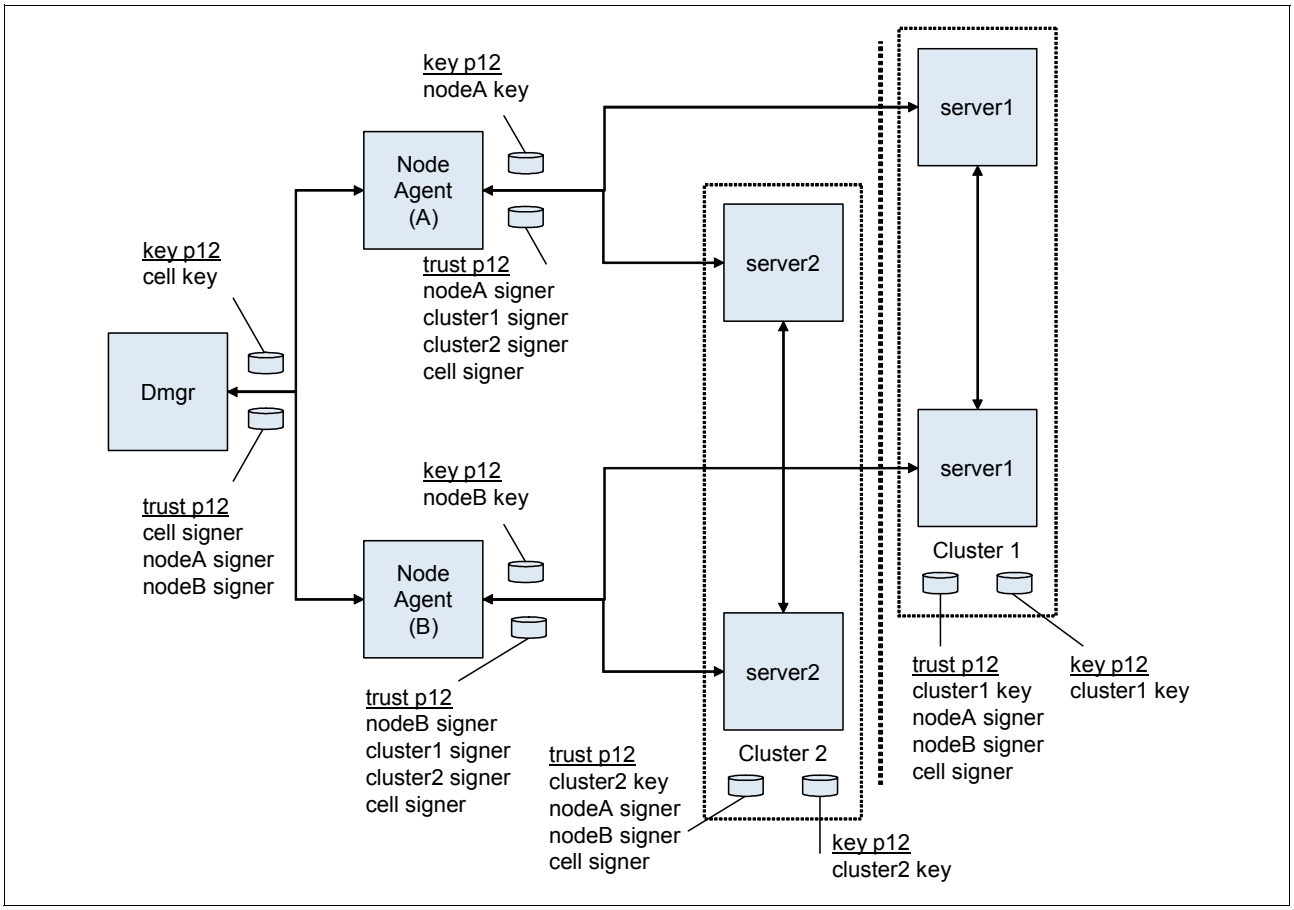


Figure 8-3 SSL cluster isolation

In the example, cluster1 might contain Web applications, and cluster2 might contain EJB applications. Considering the various protocols, you decide to enable IIOP traffic between the two clusters. Your job is to define a dynamic outbound SSL configuration at the cluster1 scope with the following properties:

```
IIOP,nodeAhostname,9403|IIOP,nodeAhostname,9404|IIOP,nodeBhostname,9403|IIOP,nodeB
hostname,9404
```

You have to create another SSL configuration at the cluster1 scope that contains a new trust.p12 file with the cluster2 signer. Consequently, outbound IIOP requests go either to nodeAhostname ports 9403 and 9404 or to nodeBhostname ports 9403 and 9404. The IIOP SSL port numbers on these two application server processes in cluster2 identify the ports.

As you review Figure 8-3, note the following features of the cluster isolation configuration:

- The trust.p12 for cluster1 contains signers that allow communications with itself (cluster1 signer), between both node agents (nodeAsigner and nodeBsigner), and with the deployment manager (cell signer).
- The trust.p12 for cluster2 contains signers that allow communications with itself (cluster2 signer), between both node agents (nodeAsigner and nodeBsigner), and with the deployment manager (cell signer).
- Node agent A and Node agent B can communicate with themselves, the deployment manager, and both clusters.

Even if this section presents an overview of isolation methods from an SSL perspective, you have to guarantee that non-SSL ports are closed or applications require the confidentiality constraint in the deployment descriptor. For example, you can set the CSIv2 inbound transport window to require SSL and disable the channel ports that are not secure from the server ports configuration.

Also, you must enable SSL client authentication for SSL to enforce the isolation requirements on both sides of a connection. Without mutual SSL client authentication, a client can easily obtain a signer for the server programmatically and thus bypass the aim of isolation. With SSL client authentication, the server would require the client's signer for the connection to succeed. For HTTP/S protocol, the client is typically a browser, a Web Service, or a URL connection. For the IIOP/S protocol, the client is typically another application server or a Java client. WebSphere Application Server must know the clients to determine if SSL client authentication enablement is possible. Any applications that are available through a public protocol must not enable SSL client authentication because the client may fail to obtain a certificate to authenticate to the server.

Important: It is beyond the scope of this chapter to describe all the factors you must consider to accomplish complete isolation. For more information about this topic, refer to *IBM WebSphere Application Server V6.1 Security Handbook*, SG24-6316.

8.3.4 Certificate management using iKeyman

Starting in WebSphere Application Server Version 6.1, you can manage your certificates from the administrative console. When using versions of WebSphere Application Server prior to Version 6.1, use iKeyman for certificate management. iKeyman is a key management utility.

WebSphere Application Server certificate management requires that you define the keystores in your WebSphere Application Server configuration. With iKeyman, you need access to the keystore file only. You can read a keystore file with personal certificates and signers that is created in iKeyman. A keystore file can be read into the WebSphere Application Server configuration by using the **`createKeyStore`** command.

The majority of certificate management functions are the same between WebSphere Application Server and iKeyman, especially for personal certificates and signer certificates. However, certificate requests are special. The underlying behavior is different in the two certificate management schemes. Because of this different behavior, when a certificate request is generated from iKeyman, the process must be completed in iKeyman. For example, a certificate that is generated by a certificate request that originated in iKeyman must be received in iKeyman as well.

The same is true for WebSphere Application Server. For example, when a certificate is generated from a certificate request that originated in WebSphere Application Server, the certificate must be received in WebSphere Application Server.

You can perform the certificate operations shown in Table 8-3 on page 297 using iKeyman.

Table 8-3 iKeyman: Certificate operations

Types of certificates	Functions	Description
Personal certificates	Create a self-signed certificate.	Creates a self-signed certificate and stores it in a keystore.
	List personal certificates.	Lists all the personal certificates in a keystore.
	Get information about a personal certificate.	Gets information about a personal certificate.
	Delete a personal certificate.	Deletes a personal certificate from a keystore.
	Import a certificate.	Imports a certificate from a keystore to a keystore.
	Export a certificate.	Exports a certificate from a keystore to another keystore.
	Extract a certificate.	Extracts the signer part of a personal certificate to a file.
	Receive a certificate.	Reads a certificate that comes from a certificate authority (CA) into a keystore.
Signer certificates	Add a signer certificate.	Adds a signer certificate from a file to a keystore.
	List signer certificates.	Lists all the signer certificates in a keystore.
	Get information about a signer certificate.	Gets information about a signer certificate.
	Delete a signer certificate.	Deletes a signer certificate from a keystore.
	Extract a signer certificate.	Extracts a signer certificate from a keystore, and stores the certificate in a file.
Certificate requests	Create a certificate request.	Creates a certificate request that can be sent to a CA.
	List certificate requests.	Lists the certificate requests in a keystore.
	Get information about a certificate request.	Gets information about a certificate request.
	Delete a certificate request.	Deletes a certificate request from a keystore.
	Extract a certificate request.	Extracts a certificate request to a file.

8.3.5 Using Sun Java System Directory Server as the LDAP server

As described in 1.5, “Solaris packaging” on page 9, Solaris includes a license for 200,000 directory entries. In this section, we discuss how you can use Sun Java System Directory Server as your LDAP repository. In Sun Java System Directory Server, the object class is the default groupOfUniqueName when you create a group. For better performance, WebSphere Application Server uses the User object to locate the user group membership from the nsRole attribute. Create the group from the role. If you want to use the groupOfUniqueName attribute to search groups, specify your own filter setting. Roles unify entries. Roles are designed to be more efficient and easier to use for applications. For example, an application can locate the role of an entry by enumerating all the roles that are possessed by a given entry, rather than selecting a group and browsing through the members list. When using roles, you can create a group using the following:

- Managed role
- Filtered role
- Nested role

All of these roles are computable by the nsRole attribute.

To improve performance for LDAP searches, the default filters for Sun Java System Directory Server are defined such that when you search for a user, the result contains all the relevant information about the user (user ID, groups, and so on). As a result, the product does not call the LDAP server multiple times. This definition is possible only in these directory types, which support searches where the complete user information is obtained.

Configuring Sun Java System Directory Server as a stand-alone LDAP registry

The following steps describe how to set up Sun Java System Directory Server as your LDAP server:

1. In the administrative console, select **Security** → **Secure administration, applications, and infrastructure**.
2. Under User account repository, select **Standalone LDAP registry** and click **Configure**. See Figure 8-4 on page 299.

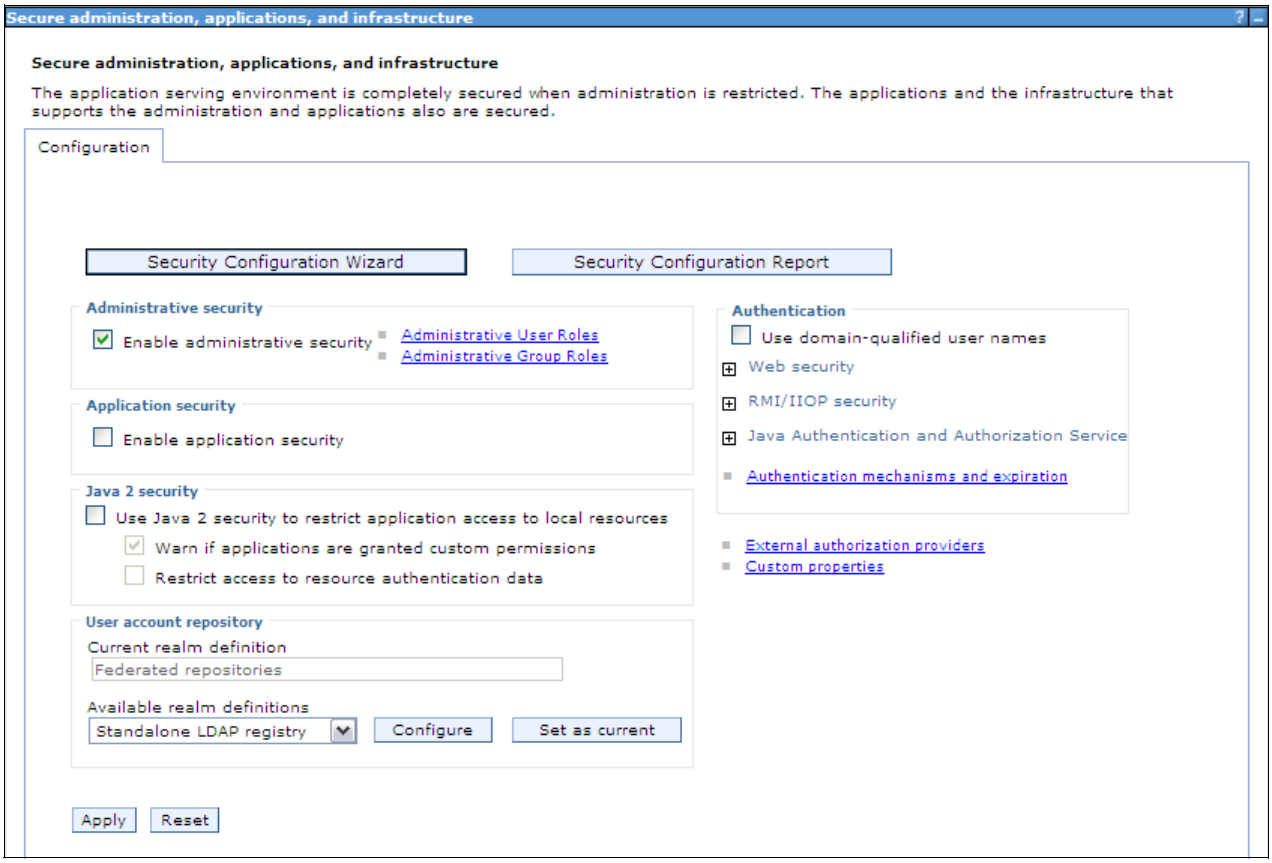


Figure 8-4 Configuring LDAP with Sun Java System Directory Server

3. Use the WebSphere Application Server administrative console to set up the information that is needed to use Sun Java System Directory Server.
4. Based on the information from the previous steps, you can specify the following values on the LDAP settings window(Figure 8-5 on page 301):
   - Primary administrative user name

     Specify the name of a user with administrative privileges that is defined in the registry. This user name is used to access the administrative console or used by wsadmin. You can either enter the complete distinguished name (DN) of the user or the short name of the user, as defined by the user filter in the Advanced LDAP settings window.
   - Type

     Specify Sun Java System Directory Server. The type of LDAP server determines the default filters that are used by WebSphere Application Server.
   - Host

     Specify the fully qualified host name of the machine that is running Sun Java System Directory Server.
   - Port

     Specify the LDAP server port number. The host name and the port number represent the realm for this LDAP server in the WebSphere Application Server cell. So, if servers in different cells are communicating with each other using Lightweight Third Party Authentication (LTPA) tokens, these realms must match exactly in all the cells.

The default value is 389. If multiple WebSphere Application Servers are installed and configured to run in the same single sign-on domain, or if the WebSphere Application Server interoperates with a previous version of the WebSphere Application Server, then it is important that the port number match all configurations.

– Base distinguished name (DN)

  The base DN indicates the starting point for searches in this LDAP directory server. For example, for a user with a DN of cn=itsoUser, ou=ITSO, o=IBM, c=US, specify the base DN as any of the following options assuming a suffix of c=us): ou=ITSO, o=IBM, c=us or o=IBM c=us or c=us. For authorization purposes, this field is case sensitive by default. Match the case in your directory server. If a token is received (for example, from another cell) the base DN in the server must match exactly the base DN from the other cell. If case sensitivity is not a consideration for authorization, enable the **Ignore case for authorization** option.

– Bind distinguished name (DN)

  Optional: Specify the bind DN name in the Bind distinguished name field. The bind DN is required if anonymous binds are not possible on the LDAP server to obtain user and group information. If the LDAP server is set up to use anonymous binds, leave this field blank. If a name is not specified, the application server binds anonymously. See the Base distinguished name (DN) field description for examples of distinguished names.

– Bind password

  Specify password corresponding to the bind DN.

– Search timeout

  Optional: Modify the timeout value. This timeout value is the maximum amount of time that the LDAP server waits to send a response to the product client before stopping the request. The default is 120 seconds.

– Reuse connection

  Ensure that the **Reuse connection** option is selected. This option specifies that the server should reuse the LDAP connection. Clear this option only in rare situations where a router is used to send requests to multiple LDAP servers and when the router does not support affinity. Leave this option selected for all other situations.

– Ignore case for authorization option

  Optional: Verify that the **Ignore case for authorization** option is enabled. When you enable this option, the authorization check is case insensitive. Normally, an authorization check involves checking the complete DN of a user, which is unique in the LDAP server and is case sensitive. However, when you use the Sun Java System Directory Server, you must enable this option because the group information that is obtained from the LDAP servers is not consistent in case. This inconsistency affects the authorization check only. Otherwise, this field is optional and can be enabled when a case sensitive authorization check is required. For example, you might select this option when you use certificates and the certificate contents do not match the case of the entry in the LDAP server. The default is enabled.

– SSL Settings

Optional: Select the **SSL Enabled** option if you want to use Secure Sockets Layer communications with the LDAP server.

If you select the SSL Enabled option, you can select either the **Centrally managed** or the **Use specific SSL alias** option.

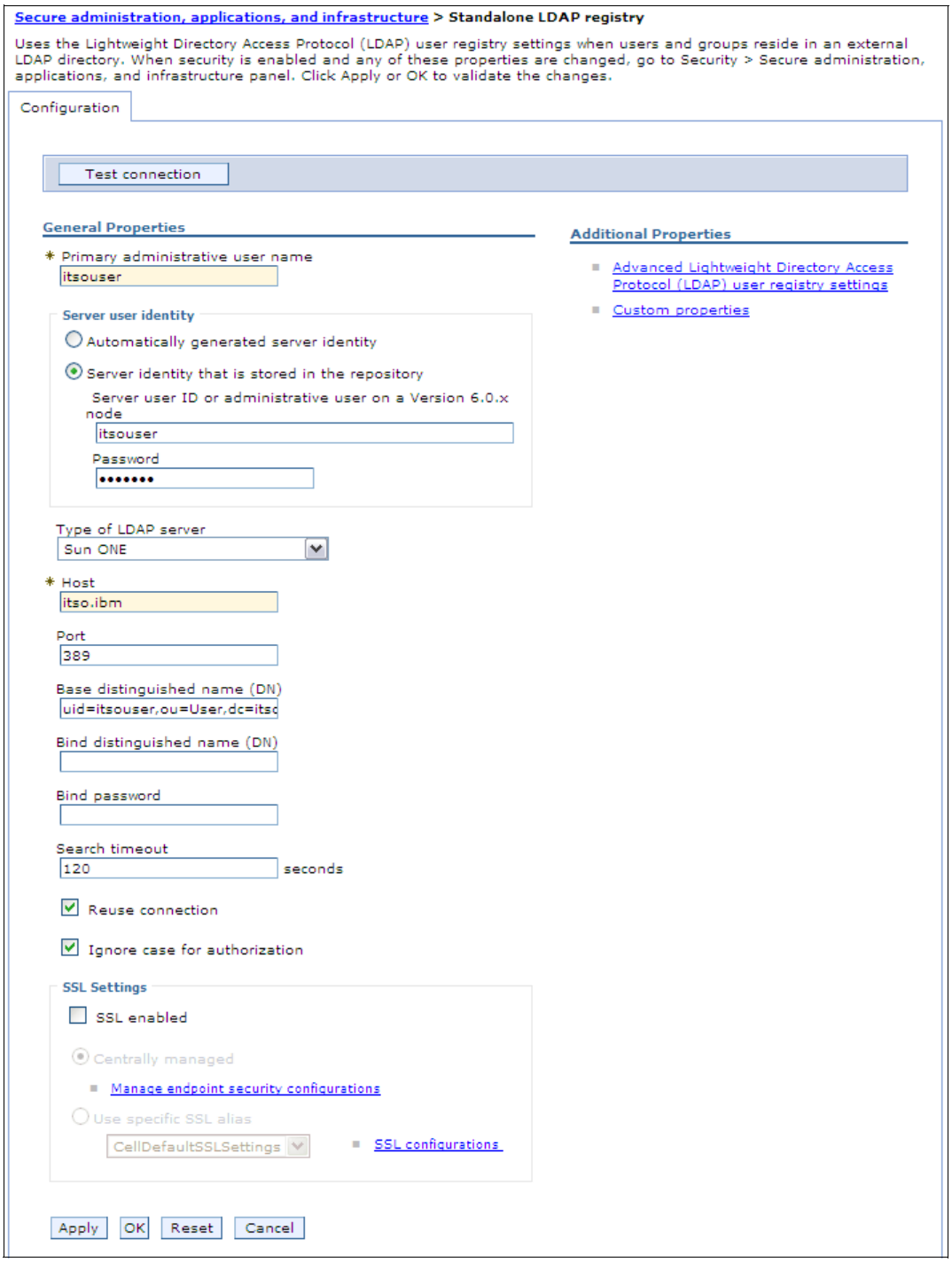


Figure 8-5 Configuring LDAP: Setting values

5. Optional: Set ObjectCategory as the filter in the Group member ID map field to improve LDAP performance. See Figure 8-6.
   a. Under Additional properties, click **Advanced Lightweight Directory Access Protocol (LDAP) user registry settings**.
   b. Add ;objectCategory:group to the end of the Group member ID map field.

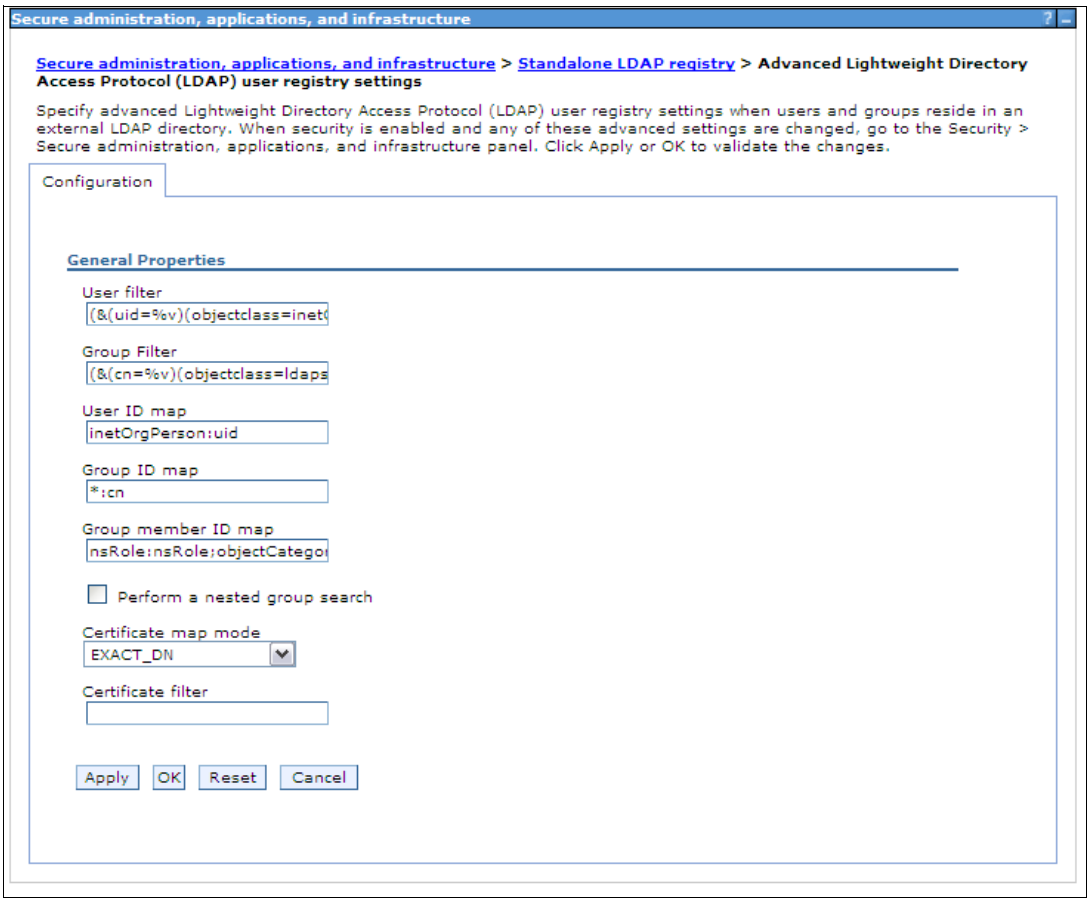


Figure 8-6 Configuring LDAP: Advanced Settings

6. Click **OK** and **Save** to save the changes to the master configuration.
7. Stop and restart the administrative server so that the changes take effect.

8.4 Using WebSphere Application Server with Solaris Kernel SSL proxy

The Kernel SSL proxy implements the SSL protocol for the application servers that do not have SSL support, so that they can receive SSL-based client requests. All SSL processing is done in the kernel and the server program only needs to send and receive data in clear text. WebSphere Application Server already implements the SSL protocol using the HTTPS port defined in the port list. However, due to some technical reasons, it cannot take advantage of Niagara Cryptographic Provider (NCP), so to use this feature, we can treat WebSphere Application Server as just providing the HTTP services and then KSSL can be used to implement SSL. Once we do this, we can take advantage of this provider to offload the SSL processing to the hardware.

KSSL is added during the implementation and is responsible for the server side SSL protocol. It works as the SSL proxy server providing a clear text proxy port to the application server and listening on the SSL port. It manages keys and certificates and it is responsible for the SSL handshake with the clients and managing the SSL session state information. SSL handshake is asynchronous without an application server.

On UltraSPARC T1 and T2 based systems, you can use the KSSL module to take advantage of all the features provided by the OS and the Solaris Cryptographic Framework.

8.4.1 Configuration of the KSSL module on Solaris

In the following example, we will use the Solaris tools to manage the certificate and keys that will enable us to use the KSSL feature in Solaris 10 on Sun CoolThread servers.

Solaris provides a tool called **pktool** to manage the certificates and keys and which can also be used for managing the certificates and keys on multiple keystores, including PKCS#11 tokens (that is, Cryptographic Framework), Netscape Security Services (NSS) tokens, and standard file based keystore for OpenSSL. In this example, we will talk about the PKCS#11 based keystore so we take the advantage of Cryptographic Framework. **pktool** also provides additional capability to manage the Certificate Revocation List (CRL). The details for using **pktool** can be found by issuing the command **man pktool**.

pktool's token subcommand can be used to list all the available PKCS#11 tokens, which can be used to generate the certificate later. An example of this subcommand is shown in Example 8-2.

Example 8-2 Example pktool's tokens subcommand

```
# pktool tokens
Token Label                     Manuf ID         Serial No        PIN State
ncp/0 Crypto Accel Asym 1.0     SUNWncp                           user set
Sun Software PKCS#11 softtoken  Sun Microsystem                   default
```

In Example 8-2, we have the following PKCS#11 tokens:

- `ncp/0 Crypto Accel Asym 1.0`
- `Sun Software PKCS#11 softtoken`

We are going to use the token named `Sun Software PKCS#11 softtoken`, as shown in Example 8-3, to generate the certificate.

Example 8-3 Sample pktool gencert command

```
bash# pktool gencert -i keystore=pkcs11 token="Sun Software PKCS#11 softtoken"
label=cert-label serial=0x0102030405ffeeddee lifetime=2-year

Entering following fields for subject (a DN) ...
Country Name (2 letter code) [US]:
State or Province Name (full name) [Some-State]:CA
Locality Name (eg, city) []:San Jose
Organization Name (eg, company) []:IBM
Organizational Unit Name (eg, section) []:ITSO
Common Name (eg, YOUR name) []:FirstName LastName
Email Address []:FirstName.LastName@itso.com
Enter pin for Sun Software PKCS#11 softtoken:
```

The command in Example 8-3 on page 303 generates a self-signed certificate and installs it and its associated private key to the specified keystore. The **gencert** subcommand also prompts the user to enter a *pin* for token-based keystore. If the pin has not been set using the default system, then it has to be explicitly set using the **setpin** command before the **gencert** subcommand, as shown in Example 8-4.

Example 8-4 Sample pktool setpin subcommand

```
bash# pktool setpin keystore=pkcs11
The default system set pin is : changeme
```

cryptoadm is another command provided by Solaris 10 to manage the cryptographic framework, which includes listing, configuring, and installing and uninstalling a cryptographic provider.

Before we configure the KSSL proxy, we need to disable the metaslot feature in the cryptographic framework, as shown in Example 8-5.

Example 8-5 Sample command to list and disable metaslot

```
bash# cryptoadm list metaslot
bash# cryptoadm disable metaslot
```

Now we need to create the a pin file in which we need to enter the pin in plain text format, as shown in Example 8-6. The proper caution and access rights need to be given to the pin file to secure it.

Example 8-6 Sample pin file

```
bash# cat /var/tmp/pin
changeme
```

Once this is done, we have everything ready to create the proxy. For this task, we need to go to administrative console of WebSphere and create another HTTP port and bind it to default_host virtual host, as shown in Example 8-7. Take note of the port just created and then stop the application server.

Example 8-7 Sample ksscfg command

```
bash# ksslcfg create -f pkcs11 -T "Sun Software PKCS#11 softtoken" -C "cert-label"
-x 9080 -p /var/tmp/pin 443
```

At this point, we need to start the application server so that it is ready to accept client requests.

In Example 8-7, a proxy SSL port has been created using port 443, which will receive the client's SSL request and forward it to the plain text HTTP port running on 9080. For demonstration purposes, we have used the default HTTP port, which needs to be replaced by the new port, which has been created as mentioned previously.

The KSSL proxy services can be enabled and disabled by using the commands shown in Example 8-8 on page 305.

Example 8-8 svcadm disable and enable commands

```
svcadm disable /network/ssl/proxy - disable the kssl proxy
svcs | grep kssl - looks for kssl proxy services

svcadm enable /network/ssl/proxy - enables the kssl proxy
```

8.4.2 Verification of KSSL setup for WebSphere Application Server

Logs for this example KSSL proxy service are written to the /var/svc/log/network-ssl-proxy:kssl-INADDR_ANY-443.log file.

In case of any issues, this log file can be checked for error messages before taking corrective actions.

To check the connectivity on the newly created SSL port on port, 443, do either of the following tasks:

- Run **openssl s_client -connect ws07 443**.
- Open a browser and go to `https://<hostname>:443/<application-url>`.

If everything is configured correctly, then the user will get a response from the server, and, based on the method, the user will be prompted to accept the certificate presented by the server.

Use the **kstat** command to verify that the Solaris 10 kernel is indeed doing the SSL processing for the application. This command lists the different kernel statistics by default, but the output can be made more specific by using different command options. In this case, we can use the **kstat** command, as shown in Example 8-9.

Example 8-9 Sample kstat command

```
# kstat -m kssl
module: kssl                            instance: 0
name:   kssl_stats                      class:    crypto
        crtime                          48.449929752
        kssl_alloc_fails                0
        kssl_appdata_record_ins         0
        kssl_appdata_record_outs        0
        kssl_bad_pre_master_secret      0
        kssl_compute_mac_failure        0
        kssl_fallback_connections       0
        kssl_fatal_alerts               0
        kssl_full_handshakes            0
        kssl_internal_errors            0
        kssl_no_suite_found             0
        kssl_proxy_fallback_failed      0
        kssl_record_decrypt_failure     0
        kssl_resumed_sessions           0
        kssl_sid_cache_hits             0
        kssl_sid_cache_lookups          0
        kssl_sid_uncached               0
        kssl_verify_mac_failure         0
        kssl_warning_alerts             0
        snaptime                        532160.997096896
```

8.4.3 Using WebSphere Application Server with KSSL in the Solaris Zone

Solaris Zone has a limitation where the **ksscfg** command cannot be run in local zones. However, you can run the command from a Global Zone to enable the same functionality, as described in 8.4.1, “Configuration of the KSSL module on Solaris” on page 303. You can point to any host to let that host take advantages of KSSL from the host where it is being configured. All you need to have is the network connection between these two host systems. Any WebSphere Application Server instances running in local zones can be front ended by the KSSL proxy running in the Global Zone. Another important thing to note here is that the Solaris Zones do not need to be physically collocated.

8.5 Integrating third-party HTTP reverse proxy servers

These steps are required to use either a WebSEAL trust association interceptor or your own trust association interceptor with a reverse proxy security server.

WebSphere Application Server enables you to use multiple trust association interceptors. The Application Server uses the first interceptor that can handle the request.

8.5.1 Verify and configure a trust association interceptor

Do the following steps to verify and configure a trust association interceptor:

1. Log in to the administrative console.
2. Select **Security** → **Secure administration, applications, and infrastructure**.
3. Under Web security, click **Trust association**.
4. Select the **Enable trust association** option.
5. Under Additional properties, click **Interceptors**. The default value appears.
6. Verify that the appropriate trust association interceptors are listed. If you need to use a WebSEAL trust association interceptor, refer to 8.5.5, “Configuring single sign-on using the trust association interceptor” on page 308 or “Configuring single sign-on using trust association interceptor ++” on page 309.

   If you are not using WebSEAL and need to use a different interceptor, complete the following steps:

   a. Select both the **com.ibm.ws.security.web.WebSealTrustAssociationInterceptor** and the **com.ibm.ws.security.web.TAMTrustAssociationInterceptorPlus** class names and click **Delete**.

   b. Click **New** and specify a trust association interceptor.
7. If you are enabling security, make sure that you complete the remaining steps for enabling security.
8. Save, stop, and restart all of the product servers (deployment manager, nodes, and application servers) for the changes to take effect.

8.5.2 Trust association settings

Use this page to enable trust association, which integrates application server security and third-party security servers. More specifically, a reverse proxy server can act as a front-end authentication server while the product applies its own authorization policy onto the resulting credentials passed by the proxy server.

To view this administrative console page, complete the following steps:

1. Select **Security** → **Secure administration, applications, and infrastructure**.
2. Under Authentication, expand **Web security** and click **Trust association**.

When security is enabled and any of these properties change, go to the Secure administration, applications, and infrastructure window and click **Apply** to validate the changes.

Enable trust association

Specifies whether trust association is enabled.

- Data type: Boolean
- Default: Disable
- Range: Enable or Disable

8.5.3 Trust association interceptor collection

Use this page to specify trust information for reverse security proxy servers.

To view this administrative console page, complete the following steps:

1. Select **Security** → **Secure administration, applications, and infrastructure**.
2. Under Authentication, expand **Web security** and click **Trust association.**
3. Under Additional Properties, click **Interceptors**.

When security is enabled and any of these properties are changed, go to the Secure administration, applications, and infrastructure window and click **Apply** to validate the changes.

Interceptor class name

Specifies the trust association interceptor class name.

- Data type: String
- Default: com.ibm.ws.security.web.WebSealTrustAssociationInterceptor

8.5.4 Trust association interceptor settings

Use this page to specify trust information for reverse security proxy servers.

To view this administrative console page, complete the following steps:

1. Select **Security** → **Secure administration, applications, and infrastructure**.
2. Under Authentication, expand **Web security** and click **Trust association**.
3. Under Additional Properties, select **Interceptors** → **New**.

When security is enabled and any of these properties are changed, go to the Secure administration, applications, and infrastructure window and click **Apply** to validate the changes.

Interceptor class name

Specifies the trust association interceptor class name.

- Data type: String
- Default: com.ibm.ws.security.web.WebSealTrustAssociationInterceptor

8.5.5 Configuring single sign-on using the trust association interceptor

This task is used to enable single sign-on using the trust association interceptor. These steps involve setting up trust association and creating the interceptor properties.

The following steps are required when setting up security for the first time. Ensure that Lightweight Third Party Authentication (LTPA) is the active authentication mechanism:

1. From the WebSphere Application Server console, select **Security** → **Secure administration, applications, and infrastructure**.
2. Ensure that the Active authentication mechanism field is set to **Lightweight Third Party Authentication (LTPA)**. If not, set it and save your changes.

Lightweight Third Party Authentication (LTPA) is the default authentication mechanism for WebSphere Application Server. You can configure LTPA prior to configure single sign-on (SSO) by selecting **Security** → **Secure administration, applications, and infrastructure** → **Authentication mechanisms and expiration**. Although you can use Simple WebSphere Authentication Mechanism (SWAM) by selecting **Use SWAM**, there is no authenticated communication between servers option on the Authentication mechanisms and expiration window. Single sign-on (SSO) requires LTPA as the configured authentication mechanism:

1. From the administrative console for WebSphere Application Server, select **Security** → **Secure administration, applications, and infrastructure**.
2. Under Web security, click **Trust association**.
3. Select the **Enable** trust association option.
4. Under Additional properties, click the **Interceptors** link.
5. Click **com.ibm.ws.security.web.WebSealTrustAssociationInterceptor** to use the WebSEAL interceptor. This interceptor is the default.
6. Under Additional properties, click **Custom Properties**.
7. Click **New** to enter the property name and value pairs. Ensure the parameters shown in Table 8-4 are set.

Table 8-4 Trust association interceptor parameters

Option	Description
com.ibm.websphere.security. trustassociation.types	Ensure that WebSeal is listed.
com.ibm.websphere.security. webseal.loginId	The format of the user name is the short name representation. This property is mandatory. If the property is not set in the WebSphere Application Server, TAI initialization fails.

Option	Description
com.ibm.websphere.security. webseal.id	The iv-user header, which is com.ibm.websphere.security.webseal.id=iv-user.
com.ibm.websphere.security. webseal.hostnames	Do not set this property if using Tivoli Access Manager plug-in for Web servers. The host names (case sensitive) are trusted and expected in the request header, for example, com.ibm.websphere.security.webseal.hostnames=host1 This includes the proxy host names unless the com.ibm.websphere.security.webseal.ignoreProxy is set to true. Obtain a list of servers using the server list **pdadmin** command.
com.ibm.websphere.security. webseal.ports	Do not set this property if using Tivoli Access Manager Plug-in for Web Servers. The corresponding port number of the host names that are expected are in the request header. This includes the proxy ports unless the com.ibm.websphere.security.webseal.ignoreProxy is set to true, for example, com.ibm.websphere.security.webseal.ports=80, 443
com.ibm.websphere.security. webseal.ignoreProxy	An optional property that if set to true or yes ignores the proxy host names and ports in the IV header. By default, this property is set to false.

8. Click **OK**.
9. Save the configuration and log out.
10. Restart WebSphere Application Server.

Configuring single sign-on using trust association interceptor ++

Perform this task to enable single sign-on using trust association interceptor ++. The steps involve setting up trust association and creating the interceptor properties. Lightweight Third Party Authentication (LTPA) is the default authentication mechanism for WebSphere Application Server. However, you may need to configure LTPA prior to configuring the TAMTrustAssociationInterceptorPlus. LTPA is the required authentication mechanism for all trust association interceptors. You can configure LTPA by selecting **Security** → **Secure administration, applications, and infrastructure** → **Authentication mechanisms and expiration**.

Note: Enabling Web security single sign-on (SSO) is optional when you configure the TAMTrustAssociationInterceptorPlus.

Although you can use Simple WebSphere Authentication Mechanism (SWAM) by selecting the **Use SWAM-no authenticated communication between servers** option on the Authentication mechanisms and expiration window, single sign-on (SSO) requires LTPA as the configured authentication mechanism:

1. From the administrative console for WebSphere Application Server, select **Security** → **Secure administration, applications, and infrastructure**.
2. Under Web security, click **Trust** association.

3. Click **Enable Trust Association**.
4. Click **Interceptors**.
5. Click **com.ibm.ws.security.web.TAMTrustAssociationInterceptorPlus** to use the WebSEAL interceptor. This interceptor is the default.
6. Click **Custom Properties**.
7. Click **New** to enter the property name and value pairs. Verify that the parameters shown in Table 8-5 are set.

Table 8-5 Custom properties

Option	Description
com.ibm.websphere.security. webseal.checkViaHeader	You can configure TAI so that the via header can be ignored when validating trust for a request. Set this property to false if none of the hosts in the via header need to be trusted. When set to false, you do not need to set the trusted host names and host ports properties. The only mandatory property to check when the via header is false is com.ibm.websphere.security.webseal.loginId. The default value of the check via header property is false. When using Tivoli Access Manager plug-in for Web servers, set this property to false. Note: The via header is part of the standard HTTP header that records the server names that the request passed through.
com.ibm.websphere.security. webseal.loginId	The format of the user name is the short name representation. This property is mandatory. If it is not set in WebSphere Application Server, the TAI initialization fails.
com.ibm.websphere.security. webseal.id	A comma-separated list of headers that exists in the request. If all of the configured headers do not exist in the request, trust cannot be established. The default value for the ID property is iv-creds. Any other values set in WebSphere Application Server are added to the list along with iv-creds, separated by commas.
com.ibm.websphere.security. webseal.ports	Do not set this property if using Tivoli Access Manager plug-in for Web servers. This property is a comma-separated list of trusted host ports. Requests that arrive from unlisted ports might not be trusted. If the checkViaHeader property is not set, or is set to false, this property has no influence. If the checkViaHeader property is set to true, and the trusted host ports property is not set in WebSphere Application Server, the TAI initialization fails.

Option	Description

com.ibm.websphere.security. webseal.viaDepth	A positive integer that specifies the number of source hosts in the via header to check for trust. By default, every host in the via header is checked, and if any host is not trusted, trust cannot be established. The via depth property is used when only some of the hosts in the via header have to be trusted. The setting indicates the number of hosts that are required to be trusted.
	As an example, consider the following header:
	`Via: HTTP/1.1 webseal1:7002, 1.1 webseal2:7001`
	If the viaDepth property is not set, is set to 2 or is set to 0, and a request with the previous via header is received, then both webseal1:7002 and webseal2:7001 need to be trusted. The following configuration applies:
	`com.ibm.websphere.security.webseal.hostnames = webseal1,webseal2`
	`com.ibm.websphere.security.webseal.ports = 7002,7001`
	If the via depth property is set to 1, and the previous request is received, then only the last host in the via header needs to be trusted. The following configuration applies:
	`com.ibm.websphere.security.webseal.hostnames = webseal2`
	`com.ibm.websphere.security.webseal.ports = 7001`
	The viaDepth property is set to 0 by default, which means all of the hosts in the via header are checked for trust.

com.ibm.websphere.security. webseal.ssoPwdExpiry	After trust is established for a request, the single sign-on user password is cached, eliminating the need to have the TAI re-authenticate the single sign-on user with Tivoli Access Manager for every request. You can modify the cache timeout period by setting the single sign-on password expiry property to the required time in seconds. If the password expiry property is set to 0, the cached password never expires. The default value for the password expiry property is 600.
com.ibm.websphere.security. webseal.ignoreProxy	This property can be used to tell the TAI to ignore proxies as trusted hosts. If set to true, the comments field of the hosts entry in the via header is checked to determine if a host is a proxy. Remember that not all proxies insert comments in the via header indicating that they are proxies. The default value of the ignoreProxy property is false. If the checkViaHeader property is set to false, then the ignoreProxy property has no influence in establishing trust.

Option	Description
com.ibm.websphere.security. webseal.configURL	For the TAI to establish trust for a request, it requires that the SvrSslCfg run for the Java Virtual Machine on the Application Server and result in the creation of a properties file. If this properties file is not at the default URL, which is file://java.home/PdPerm.properties, the correct URL of the properties file must be set in the configuration URL property. If this property is not set, and the SvrSslCfg-generated properties file is not in the default location, the TAI initialization fails. The default value for the config URL property is file://${WAS_INSTALL_ROOT}/java/jre/PdPerm.properties.

9

WebSphere Application Server performance tuning on Solaris 10

This chapter discusses planning for performance of the deployment as well as tuning an existing environment for performance. While it does not provide specific tuning values, which would be unique to each environment and application, this chapter provides a guide for testing application servers and components of the total serving environment that should be examined and potentially tuned to increase the performance of the application. The topics include:

- Introduction to performance management and why it is important
- System performance management and tuning
- Java performance management and tuning
- WebSphere performance management and tuning
- Public performance benchmarks for application servers

To better understand Sun Java Virtual Machine (JVM) and Solaris performance, readers are suggested to read this chapter in conjunction with the following Sun documentation:

- http://java.sun.com/performance
- http://java.sun.com/performance/reference/whitepapers/5.0_performance.html
- http://java.sun.com/j2se/reference/whitepapers/memorymanagement_whitepaper.pdf
- http://java.sun.com/docs/hotspot/gc5.0/gc_tuning_5.html
- http://docs.sun.com/app/docs/doc/816-5166
- http://docs.sun.com/app/docs/doc/817-6223

9.1 Introduction

In a production environment, performance and availability of Web applications are critical. In this chapter, we describe how a production environment should be monitored, the types of data available for monitoring, the tools available within WebSphere Application Server on Solaris 10 to display that data, and tools included to provide guidance on how to optimize performance. We will also discuss WebSphere performance management, the differences in the Sun JDK on Solaris 10, and the additional tools available on Solaris.

The following topics are discussed and the differences and added value that Solaris 10 offers to any enterprise for WebSphere deployment will be highlighted:

- Performance management challenges
- System performance monitoring and tuning
- JVM performance monitoring and tuning
- WebSphere Application Server performance monitoring and tuning
- Performance benchmarks

9.2 Performance management

Performance management of any complex system requires understanding the holistic view of the system. A bottleneck in any sub-system would not let the system perform optimally even if all other sub-systems are configured or tuned optimally. Therefore, all the sub-systems require similar attention when dealing with performance issues. A particular technology may be suited in one deployment scenario but not suitable in other scenarios. Understanding the need of a complex deployment system requires understanding the challenges of the system and setting a goal about how the system is supposed to perform when moved to production. These changes are induced by either the nature of the differences in the underlying technology. For example, Solaris 10 offers different ways to handle the same task performed by other operating systems. The choice of hardware also brings in additional challenges and requires due consideration for selection of the appropriate hardware for a specific task.

9.2.1 Challenges

Performance problems can be encountered almost anywhere. The problem can be network and hardware related, back-end system related, or it can be in the various layers of the product software stack, or quite often, application design issues.

Understanding the flow used to diagnose a problem helps to establish the monitoring that should be in place for your site to detect and correct performance problems. The first dimension is the "user view," the black box view of your Web site. This is an external perspective of how the overall Web site is performing from a user's point of view and identifies how long the response time is for a user. From this black box perspective, it is important to understand the load and response time on your site. To monitor at this level, many industry monitoring tools allow you to inject and monitor synthetic transactions, helping you identify when your Web site experiences a problem.

The second step is to understand the basic health of all the systems and network that make up a user request. This is the "external" view, which typically leverages tools and utilities provided with the systems and applications running. In this stage, it is of fundamental importance to understand the health of *every system* involved, including Web servers, application servers, databases, back-end systems, and so on. If any of the systems has a problem, it may have a rippling effect and cause something like the "servlet is slow" problem.

This dimension corresponds to the "what resource is constrained" portion of the problem diagnosis. To monitor at this level, WebSphere provides PMI instrumentation and the Tivoli Performance Viewer as a starting point. There are also several industry tools built using PMI instrumentation that provide 24x7 monitoring capabilities. In addition to the capabilities built into WebSphere, there are numerous visual monitoring tools that are included with the JVM, which comes with WebSphere Application Server on the Solaris platform. Solaris 10 also offers many tools to monitor and make appropriate decisions to tune the system for optimum performance.

- System Management
  - Processor Statistics
  - Memory
  - Network
  - I/O Subsystem
- Real-time Tracing
  - DTrace
  - Sun Studio Collector/Analyzer
- JVM
  - jvmstat
  - jmap
  - jstack
  - jhat
  - And many other tools.

The third dimension is the application view. This dimension actually understands the application code that is satisfying the user request. This dimension understands that there are specific servlets that are accessing session beans, entity CMP beans, a specific database, and so on. This dimension typically comes into play in an in-depth internal understanding of who is using the resource. Typically at this stage, some type of time trace through the application or thread analysis under load conditions techniques is deployed to isolate areas of the application, and particular interactions with back-end systems or databases that are especially slow under load. WebSphere provides the Request Metrics technology as a starting point. In many cases, you start using many of the development tools provided, such as IBM Rational Application Developer, Sun Studio, and NetBeans™.

9.2.2 Goals of the WebSphere Application Server administrator

The main goal of any WebSphere Application Server administrator should be to tune the system in such a way that it meets all the service level criteria for which the system has been designed. This would involve tuning the network, hardware, operating system, JVM, and WebSphere Application Server. When all these components are tuned optimally, the system will be able to meet the enterprise's need in terms of deployment. The goals for a typical enterprise deployment would be as follows:

- Is the System Utilization optimal?
- Have all the service level criteria been met?
  - Response Time
  - Peak load Management
  - Customer Satisfaction
- Has the Garbage Collection goal been met?
- Has the system been configured to handle the changes in the load?
- Are all goals defined in quantitative terms?

Keeping the above factors in mind, the administrators must have some basic guidelines for deployment so that the system can address the needs of this application. This would be a starting point for looking into performance management.

9.2.3 Differences

The WebSphere Application Server software is available on a number of different platforms, including Solaris 10. Although WebSphere is designed to operate similarly on all platforms, there are always platform specific differences that cannot be abstracted away, and it is very important that administrators make an effort to understand these platform specific differences. The performance of any application server depends on its core processing engines, which are the JVM and the OS. Once the administrator understands these differences and knows the capability of a particular JVM and OS, the administrator will be able to manage system performance effectively. Each OS has its own tunable parameters to perform optimally with different types of applications. Similarly, each JVM implementation has its own tunable parameters. As said earlier, the JVM plays an important role in the performance of application servers; therefore, it is important to recognize these differences regarding WebSphere on Solaris 10.

9.2.4 Special considerations

The current advances in processor and virtualization technologies impose new challenges on the WebSphere Application Server administrator, which brings in additional complexity. Understanding the system architecture and the application architecture is necessary.

Processors have been traditionally designed to provide a high degree of parallelism at the instruction level using a single hardware thread of execution. Such complex processors have been able to reach extremely high processing frequencies. However, these high clock rates have been achieved only with the unwanted side effects of additional power usage and decreased efficiency through excess generation of heat. What is more, the issue of memory latency causes the otherwise fast single-threaded processors to waste time in an idle state while they wait for memory to become available. Finally, given the type of demanding business workload requirements, applications have not been able to reap the benefits of parallel instructions; rather, they have had to rely on parallel threads of execution.

Sun offers various hardware technology that may be suited to one deployment scenario but may not be appropriate for another deployment situation. Understanding the system behavior when choosing the right hardware saves a lot of deployment and support calls later in the deployment cycle.

Another important thing to note about WebSphere on Solaris is that it is bundled with Sun JDK with some changes to some of the components of a Sun's JDK implementation. This requires additional caution when managing the performance of the system on Solaris 10 and other Solaris platforms. The tuning parameters and behavior of the Sun's JDK may be different when compared to the IBM JDK, even if technically they are the same specification level implementation of Java Virtual Machine. In addition to the JVM specification level, there are additional tools and technologies that may be available with a particular implementation and that can be used to manage the performance.

As explained in 1.7.2, “The Java technology for WebSphere on Solaris” on page 12, the Sun JVM we use here has had changes made to the ORB/XML and Security component. The details of the changes can be found in the following files in WAS_INSTALL_DIR/java/:

- README_FIRST
- ibmorbguide.htm
- securityguide.ref.htm

Refer to these files in order to understand the differences and plan accordingly.

9.3 System performance management

For any complex software system, it is essential to have the infrastructure tuned and configured so that it can optimally support the applications it runs. The infrastructure includes the network, hardware, and operating system. For example, if the network does not have the capability to pass the data that is being generated by applications running in the network, then the software will wait for the network congestion to be cleared. This will result in performance problems. Thus, it is essential to have all the components configured correctly to support the load that is being expected. As the application goes live, proper monitoring and management will result in better performance of the system. In this section, we will discuss some of these system aspects in detail.

Performance management for any system would include the following items:

- Staying up to date with patches for your OS, for example, Solaris 10
- System monitoring tool
  - Collecting data
  - Analyzing
  - Taking corrective actions
- Tuning

9.3.1 Solaris patches for Java

It is important to keep systems up to date with Solaris patches because the patches provide fixes for security vulnerabilities, performance, and other improvements of the OS. Certain patches specifically target the Java platform, which benefits WebSphere Application Server in regards to performance and functionalities.

Different organizations have different policies regarding updated software components from the software and OS vendors. We recommend that JDK and the Solaris OS be updated with a Solaris patch cluster. While updating with patches solves some problems, it may bring in additional complexity. Therefore, you need to plan carefully before updating the system. Some of the implications would be as follows:

- Analyzing the need for a specific patch in your environment
- Addressing Sun notifications for Solaris 10, JDK, and WebSphere Application Server alerts or the IBM Fix Pack guidelines for WebSphere Application Server
- Addressing the need for change in your environment

Solaris delivers patches in different ways to address different needs. Details can be found at:

http://sunsolve.sun.com/show.do?target=patchpage

The details about the WebSphere Application Server requirement for the Solaris 10 update patch can be found at:

http://www.ibm.com/support/docview.wss?rs=180&uid=swg27006921

It is essential that your environment functions correctly, and you must meet the criteria outlined on that Web site. Any deviation from it may result in an unsupported configuration as well as other issues, including performance.

For example, the system requirements for WebSphere Application Server V6.1 on Solaris SPARC can be found at:

http://www.ibm.com/support/docview.wss?rs=180&uid=swg27007662

The same applies to the JVM version as well, which can be found at:

http://www.ibm.com/support/docview.wss?rs=180&uid=swg27005002

The following Web site has information about keeping the Java SE patches up-to-date for Solaris:

http://sunsolve.sun.com/pub-cgi/show.pl?target=patches/JavaSE

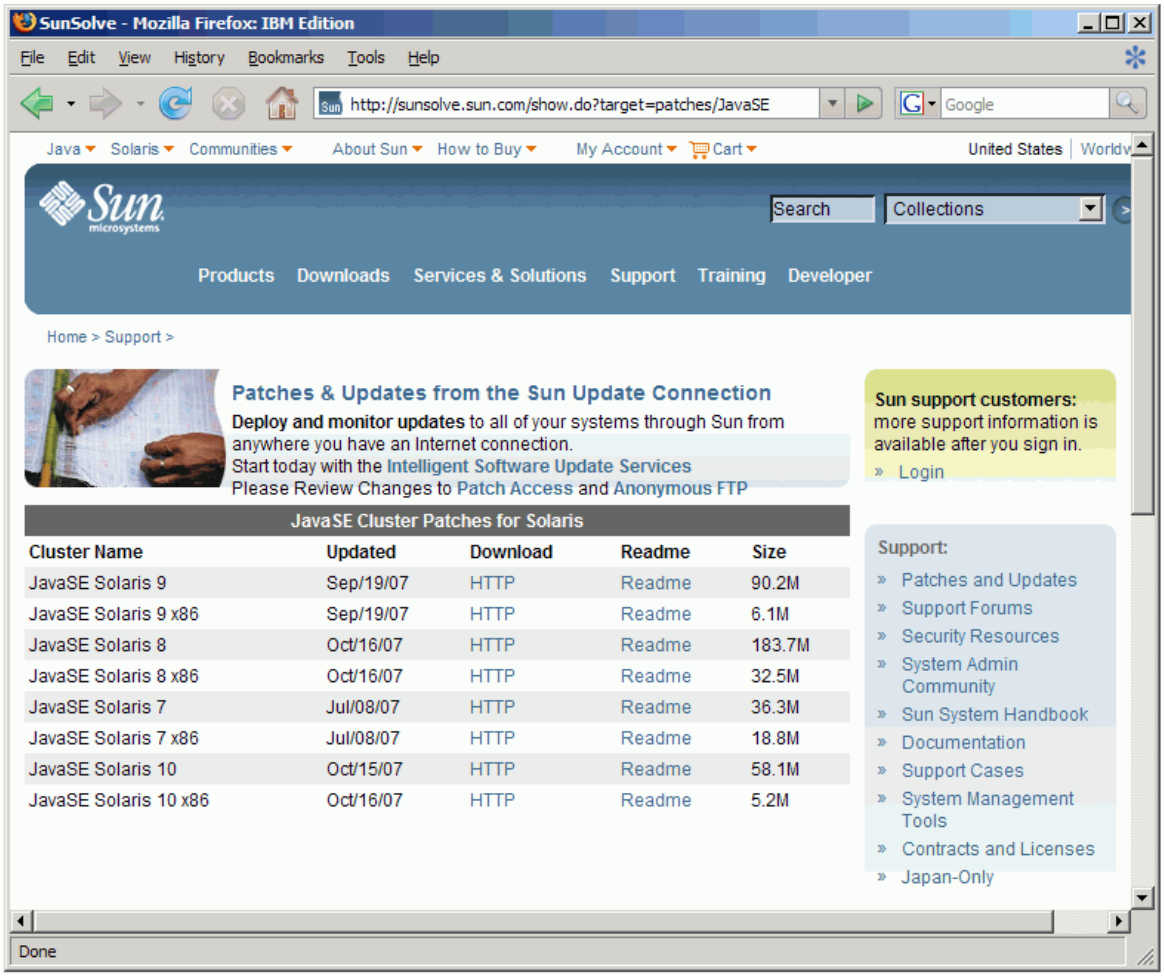


Figure 9-1 The Sun Web site for Solaris updates and patches

9.3.2 Solaris system monitoring

In this section, we discuss how you can monitor your Solaris OS. We recommend using the commands described in this section to monitor the system's behavior. We provide the definition of each command along with the recommended command line option(s), if applicable. We also provide a sample output for each command and explain what you should be paying attention to in the output.

Note: The command definitions are from the Solaris online manual, which we provide a link to at the end of each command section.

Each command is followed by its manual section information inside the parenthesis. For example, **vmstat**(1M) means it is from the System Administration Commands section 1M.

- **vmstat**(1M)

  The **vmstat** command reports virtual memory statistics regarding kernel thread, virtual memory, disk, trap, and CPU activity.

  On multi-processor (MP) systems, **vmstat** averages the number of CPUs into the output. This command can used to get monitoring data about the following items:

  – Kernel Threads
  – Memory
  – Paging
  – Disk
  – CPU

  Example 9-1 shows a sample **vmstat** output. Looking at the last row, it depicts a system that has about 83% of its capacity being used (that is, 17% idle), out of which 64% of the time being spent is user time and 19% is the kernel time. The Kernel Threads in the run queue is at 40, which means there are 40 processes ready in the run queue. Looking at the page and disk columns, we do not see any paging activities and disk contentions for input/output. It is also clear that enough swap and free memory are available. Therefore, the load on this system can be increased to achieve higher resource utilization.

Example 9-1 Example vmstat output

```
kthr      memory            page            disk          faults      cpu
 r b w   swap  free  re  mf pi po fr de sr s0 s2 s4 --   in   sy   cs us sy id
 46 0 0 23158808 8700152 0 5 0 10 10  0  0  0  5  0  0 9750 66324 22325 62 19 18
 46 0 0 23154296 8682592 0 6 0 12 12  0  0  0  4  0  0 8855 66115 21963 63 19 18
 28 0 0 23146520 8653920 0 13 0 3  3  0  0  0  2  0  0 8519 64506 20813 63 19 18
 53 0 0 23143616 8636272 0 7 0  3  3  0  0  0  3  0  0 8987 65462 21436 62 19 19
 47 0 0 23139520 8616152 0 12 0 3  3  0  0  0  3  0  0 7747 66572 20841 65 19 16
 38 0 0 23123136 8583000 0 5 0  9  9  0  0  0  4  0  0 7436 65995 21025 64 19 17
 40 0 0 23119856 8564560 0 9 0  3  3  0  0  0  3  0  0 8624 66608 22553 64 19 17
```

  If the last column marked "id" (idle CPU) reaches the range of zero to less than 10%, the system has reached the saturation point and needs to be tuned to lower CPU utilization.

  In cases where the run queue is very high (the column kthr "r"), it may mean that there are too many threads currently in the system and they are not getting CPU time when they are ready for execution. These conditions can be eliminated by reducing different thread pool sizes within WebSphere Application Server. The optimal thread pool settings are crucial to the system's stability and performance.

For more details on the use of **vmstat**, go to:

http://docs.sun.com/app/docs/doc/816-5166/vmstat-1m

- **mpstat**(1M)

  The **mpstat** command reports processor statistics in tabular form. Each row of the table represents the activity of one processor. The first table summarizes all activity since boot. Each subsequent table summarizes activity for the preceding interval. All values are rates listed as events per second unless otherwise noted.

  Example 9-2 shows an example **mpstat** output. If we look at this data, we find that everything seems right and the system CPUs are still around 18% idle. The system may appear to have some head room for increased load to support more users or improve throughput. However, if we look at the data more carefully, we find that the last row, marked in bold for CPU ID 27, shows that CPU is “0” idle. This indicates that the CPU does not have any processing capacity left. That may be one possible point of contention and we need to discover the cause of this problem. This may be the bottleneck where the system cannot utilize the remaining 18% idle capacity. We will address this problem later in the chapter when we show how a combination of tools can be used to resolve this type of performance problem.

Example 9-2 Example mpstat output

CPU	minf	mjf	xcal	intr	ithr	csw	icsw	migr	smtx	srw	syscl	usr	sys	wt	idl
0	1	0	944	458	180	1472	54	323	683	0	5062	69	17	0	14
1	1	0	82	216	0	1389	55	304	784	0	5172	67	17	0	16
2	1	0	78	223	0	1460	60	302	722	0	4864	68	15	0	17
3	0	0	65	182	0	1394	50	283	641	0	5089	68	14	0	18
8	1	0	60	239	2	1549	64	317	717	0	5319	69	16	0	16
9	1	0	68	183	2	1767	53	293	656	0	5256	69	14	0	17
10	1	0	54	200	0	1558	55	297	691	0	5345	69	15	0	17
11	1	0	74	172	0	1390	53	270	657	0	5314	67	14	0	18
16	1	0	57	232	1	1507	63	303	723	0	5549	69	15	0	16
17	1	0	67	181	0	1565	53	279	683	0	5109	68	15	0	17
18	1	0	85	190	0	1507	51	286	643	0	4847	69	14	0	17
19	1	0	57	171	0	1346	53	267	643	0	4799	67	14	0	19
24	1	0	89	189	0	1445	51	295	565	0	4934	71	12	0	17
25	4	0	58	164	0	1379	48	281	588	0	5066	70	12	0	18
26	1	0	71	145	0	1159	48	254	457	0	4372	70	14	0	16
27	**0**	**0**	**1576**	**4874**	**4840**	**335**	**91**	**34**	**2260**	**0**	**385**	**7**	**93**	**0**	**0**

For more details on the **mpstat** command, see the documentation at:

http://docs.sun.com/app/docs/doc/816-5166/mpstat-1m

- **intrstat**(1M)

  The **intrstat** utility iteratively examines the interrupt distribution over the CPU cores.

  The **intrstat** utility gathers and displays runtime interrupt statistics. The output is a table of device names and CPU IDs, where each row of the table denotes a device, and each column of the table denotes a CPU. Each cell in the table contains both the raw number of interrupts for the given device on the given CPU, and the percentage of absolute time spent in that device's interrupt handler on that CPU.

  The device name is given in the form of {name}#{instance}. The name is the normalized driver name, and typically corresponds to the name of the module implementing the driver (refer to the **ddi_driver_name** command (9F) for more information). Many Sun-delivered drivers have their own manual pages (refer to the **intro** command (7) for more information).

If the standard output is to a terminal, the table contains as many columns of data as can fit within the terminal width. If the standard output is not a terminal, the table contains at most four columns of data. By default, data is gathered and displayed for all CPUs. If the data cannot fit in a single table, it is printed across multiple tables. The set of CPUs for which data is displayed can be optionally specified with the -c or -C option.

By default, **intrstat** displays data once per second and runs indefinitely. Both of these behaviors can be optionally controlled with the interval and count parameters, respectively.

Example 9-3 shows an example **intrstat** output.

Example 9-3 Example intrstat output

```
      device |     cpu16 %tim     cpu17 %tim     cpu18 %tim     cpu19 %tim
-------------+------------------------------------------------------------
    e1000g#0 |         0  0.0         0  0.0         0  0.0         0  0.0
    e1000g#1 |         0  0.0         0  0.0         0  0.0         0  0.0
    e1000g#2 |         0  0.0         0  0.0         0  0.0         0  0.0
       mpt#0 |         0  0.0         0  0.0         0  0.0         0  0.0

      device |     cpu24 %tim     cpu25 %tim     cpu26 %tim     cpu27 %tim
-------------+------------------------------------------------------------
    e1000g#0 |         0  0.0         0  0.0         0  0.0      1726 88.7
    e1000g#1 |         0  0.0         0  0.0         0  0.0         0  0.0
    e1000g#2 |         0  0.0         0  0.0         0  0.0         0  0.0
       mpt#0 |         0  0.0         0  0.0         0  0.0         0  0.0

      device |      cpu0 %tim      cpu1 %tim      cpu2 %tim      cpu3 %tim
-------------+------------------------------------------------------------
    e1000g#0 |         0  0.0         0  0.0         0  0.0         0  0.0
    e1000g#1 |         0  0.0         0  0.0         0  0.0         0  0.0
    e1000g#2 |         0  0.0         0  0.0         0  0.0         0  0.0
       mpt#0 |         1  0.0         0  0.0         0  0.0         0  0.0
```

As we noted in the previous example, there is some problem with the system and we were looking for the way to determine the root cause. So on the same system, we ran **intrstat** for the same duration as **mpstat**. The data from this tool, the row in bold, shows that CPU ID 27 is receiving all the network interrupts for the device marked with id `e1000g#0`. This means that on this system, WebSphere Application Server has been configured to use the e1000g#0 device for its network communications and all interrupts generated due to network activities are going to CPU ID 27, which does not have any more room to accommodate any additional load. The system needs to be tuned and additional network cards need to be used. We may do *link aggregation* here or add another instance of WebSphere Application Server so that the load gets distributed. Another strategy would be to do some tuning to spread the interrupt processing over multiple CPU cores by using the following tunable parameters in /etc/system:

```
set ip:ip_soft_rings_cnt = 8
set ddi_msix_alloc_limit = 8
```

For more details on the **intrstat** command, see the documentation at:

http://docs.sun.com/app/docs/doc/816-5166/intrstat-1m

- **prstat**(1M)

  The **prstat** utility iteratively examines all active processes on the system and reports statistics based on the selected output mode and sort order. **prstat** provides options to examine only processes matching specified PIDs, UIDs, zone IDs, CPU IDs, and processor set IDs.

For more details on the **prstat** command, see the documentation at:

http://docs.sun.com/app/docs/doc/816-5166/prstat-1m

- **iostat**(1M)

  The **iostat** utility iteratively reports terminal, disk, and tape I/O activity, as well as CPU utilization. The first line of output is all the time since boot; each subsequent line is for the prior interval only:

  ```
  # iostat -xnz 5
  ```

  In Example 9-4, we see that the disk c4t0d0 is 77% busy and there is a wait of 2% as well, but despite that, the service time looks to be 1.2 only, so compared to the disk c0t1d0, which has a service time of 19.1, this disk is doing well. If we are doing any logging or using the disk for Message Store, then we need to pay attention to this utility to improve or diagnose the performance problems arising out of this situation.

Example 9-4 Sample iostat command output

```
extended device statistics
r/s    w/s   kr/s    kw/s wait actv wsvc_t asvc_t  %w  %b device
0.0    1.3    0.0     3.7  0.0  0.0    0.0    7.4   0   1 c0t0d0
0.2    1.2    1.6     3.7  0.0  0.0    0.0   19.1   0   0 c0t1d0
0.0  631.3    0.0  2536.0  0.0  0.8    0.0    1.2   2  77 c4t0d0
6.7   31.3   26.8   179.0  0.0  0.1    0.0    2.7   0  10 c5t0d0
```

For more details on the **iostat** command, see the documentation at:

http://docs.sun.com/app/docs/doc/816-5166/iostat-1m

- **netstat**(1M)

  The **netstat** command is used to get different network statistics. It displays the contents of certain network-related data structures in various formats, depending on the options you select. Based on command options, you can get various data points from this command to see how the network is behaving and based on that take appropriate actions.

  For more details on the **netstat** command, see the documentation at

  http://docs.sun.com/app/docs/doc/816-5166/netstat-1m

- **corestat**

  The **corestat** command is a tool that monitors the utilization of UltraSPARC T1 and T2 cores. The **corestat** command reports aggregate core usage based on the instructions executed by a core (that is, by the available Virtual Processors sharing the same core).

  As we discussed earlier in the system performance management section, there are tools that existed before the Chip Multi Threading (CMT) processor came into existence. Some of the advances brought additional complexity, so to understand the characteristics of these systems, we need to look at the collective data from hardware counters and system monitoring tools (that is, **vmstat** and **mpstat**). It is essential to understand the CMT processor and core usages in the right context to be able to make the proper decision for performance management and related tasks.

  When Sun first started shipping the CMT based systems, called UltraSPARC T1, some of the Sun engineers had written the **corestat** utility to monitor the status of the processor cores. The **corestat** utility is based on hardware performance counters and is very helpful in understanding the system behavior under load. UltraSPARC T2 is the second generation CMT processor, which is the most current product at this writing.

  For T1 based processor details, go to:

  http://www.solarisinternals.com/wiki/index.php/CMT_Utilization

For more details, go to the following blogs. These blogs also have the **coretstat** utility, which can be downloaded free of charge and be used *as is* in making performance and capacity planning decisions:

- For UltraSPARC T1 based system:

  http://blogs.sun.com/travi/entry/ultrasparc_t1_utilization_explained

- For UltraSPARC T2 based system:

  http://blogs.sun.com/travi/entry/corestat_for_ultrasparc_t2

- The corestat tool details:

  http://blogs.sun.com/travi/entry/corestat_core_utilization_reporting_tool

Sun's UltraSPARC T1 processor is very radical approach to multicore processing. They are often described as a "system on a chip", and provide very high scalability and throughput. More details about T1 can be found at:

http://sun.com/processors/UltraSPARC-T1/

Due to a radical design shift, we need to understand some of the principle changes behind the processing of these systems. The traditional system monitoring tool may not be adequate to describe the complete details of the system's utilization. This processor is Chip Multiprocessing combined with Chip Multithreading. The thread scheduling and OS treat each of the threads as different CPUs, which results in 32 processors on T1 and 64 processors on T2 based servers, respectively. The conventional idle state and the idle state in the context of these systems would mean different things. Thus, paying attention to this idle state will help resolve performance issues. Utilization of the core will be different from the utilization of the system, and looking at both of them will provide the guidelines for tuning or even increasing the load on the system.

As a result, the meaning of the idle thread has been changed, which requires a different understanding of processor behavior. The blog's previously referenced can help you understand these differences and make the right choice for performance tuning. The tool can be downloaded from:

- For UltraSPARC T1 based systems:

  http://blogs.sun.com/roller/resources/travi/corestat.v.1.1.tar.gz

- For UltraSPARC T2 based systems:

  http://blogs.sun.com/travi/resource/corestat.v.1.2.2.tar.gz

- **plockstat**(1M)

  The **plockstat** utility gathers and displays user-level locking statistics. By default, **plockstat** monitors all lock contention events, gathers frequency and timing data about those events, and displays the data in decreasing frequency order, so that the most common events appear first.

  The **plockstat** utility gathers data until the specified command completes or the process specified with the -p option completes.

  The **plockstat** utility relies on DTrace to instrument a running process or a command it invokes to trace events of interest. This imposes a small but measurable performance impact on the processes being observed. Users must have the dtrace_proc privilege and have permission to observe a particular process with **plockstat**. For more information, see 9.3.3, "Dynamic Tracing (DTrace)" on page 330.

Example 9-5 is a partial output from running **plockstat** on a WebSphere Application Server process.

Example 9-5 Sample plockstat output

```
Mutex block
Count     nsec Lock                         Caller
-------------------------------------------------------------------------------
  1 207832479 libc.so.1`libc_malloc_lock   0xfe5e673c
  1 70760816 libc.so.1`libc_malloc_lock   libjava.so`JNU_ReleaseStringPlatformCha
  1 68851770 libc.so.1`libc_malloc_lock   libjava.so`JNU_ReleaseStringPlatformCha
  1 66458093 0xa326210                    libibmaio.so`Java_com_ibm_io_async_Asyn
  1 63437017 0x60545a8                    libibmaio.so`Java_com_ibm_io_async_Asyn
 14 37501211 libc.so.1`libc_malloc_lock   libjvm.so`__1cKJavaThread2T6M_v_+0x160
9 16309499 libc.so.1`libc_malloc_lock   0xfe5e3f78
1722 13714048 libc.so.1`libc_malloc_lock   libnet.so`NET_AllocSockaddr+0x34
1806 13182818 libc.so.1`libc_malloc_lock   libnio.so`Java_sun_nio_ch_ServerSocketC
4326 13164350 libc.so.1`libc_malloc_lock   libnet.so`Java_java_net_SocketOutputStr
1383 12907046 libc.so.1`libc_malloc_lock   libjava.so`JNU_GetStringPlatformChars+0
496 12873886 libc.so.1`libc_malloc_lock   libibmaio.so`Java_com_ibm_io_async_Asyn
365443 12699974 libc.so.1`libc_malloc_lock libibmaio.so`Java_com_ibm_io_async_Asyn
1756 12687757 libc.so.1`libc_malloc_lock   libjvm.so`Unsafe_FreeMemory+0x234
1131579 12635537 libc.so.1`libc_malloc_lock   libnet.so`Java_java_net_SocketInputStre
330896 12614514 libc.so.1`libc_malloc_lock libibmaio.so`Java_com_ibm_io_async_Asyn
1147174 12565159 libc.so.1`libc_malloc_lock   libnet.so`Java_java_net_SocketInputStre
1963 12530597 libc.so.1`libc_malloc_lock   libibmaio.so`getDevPollEvent+0x4
2051 12414075 libc.so.1`libc_malloc_lock   0xf98eb300
451 12316496 libc.so.1`libc_malloc_lock   libibmaio.so`Java_com_ibm_io_async_Asyn
3989 12295590 libc.so.1`libc_malloc_lock   libjvm.so`__1cCosGmalloc6FI_pv_+0x28
166 11700768 libc.so.1`libc_malloc_lock   libc.so.1`dgettext+0x98
1165 10418987 libc.so.1`libc_malloc_lock   libjava.so`JNU_ReleaseStringPlatformCha
 67 10072643 libc.so.1`libc_malloc_lock   libjvm.so`__1cCosGmalloc6FI_pv_+0x28
  8 10039469 libc.so.1`libc_malloc_lock   libjvm.so`__1cCosLfree_thread6FpnIOSThr
 44  9390603 libc.so.1`libc_malloc_lock   libjvm.so`__1cQChunkPoolCleanerEtask6M_
162  9288702 libc.so.1`libc_malloc_lock   libc.so.1`dgettext+0x98
  9  8859302 libc.so.1`libc_malloc_lock   libjvm.so`__1cGThread2T5B6M_v_+0x40
1  5617493 libc.so.1`libc_malloc_lock   libverify.so`VerifyClassForMajorVersion
399  3293079 libc.so.1`libc_malloc_lock   libnet.so`NET_AllocSockaddr+0x34
951  3183286 libc.so.1`libc_malloc_lock   libnet.so`Java_java_net_SocketOutputStr
414  2814430 libc.so.1`libc_malloc_lock   libnio.so`Java_sun_nio_ch_ServerSocketC
302  2784543 libc.so.1`libc_malloc_lock   libjava.so`JNU_GetStringPlatformChars+0
242409  2676346 libc.so.1`libc_malloc_lock   libnet.so`Java_java_net_SocketInputStre
78806  2645377 libc.so.1`libc_malloc_lock   libibmaio.so`Java_com_ibm_io_async_Asyn
883  2617589 libc.so.1`libc_malloc_lock   libjvm.so`__1cCosGmalloc6FI_pv_+0x28
71292  2615754 libc.so.1`libc_malloc_lock   libibmaio.so`Java_com_ibm_io_async_Asyn
98  2531417 libc.so.1`libc_malloc_lock   libibmaio.so`Java_com_ibm_io_async_Asyn
874  2448724 libc.so.1`libc_malloc_lock   libnet.so`Java_java_net_SocketOutputStr
35  2214630 libc.so.1`libc_malloc_lock   libc.so.1`dgettext+0x98
106  2201580 libc.so.1`libc_malloc_lock   libibmaio.so`Java_com_ibm_io_async_Asyn
```

For more details about the **plockstat** command, go to:

http://docs.sun.com/app/docs/doc/816-5166/plockstat-1m

- **kill**(1)

  The **kill** command sends a signal to one or more process IDs specified by each PID operand.

For each PID operand, the **kill** utility will perform actions equivalent to the **kill(2)** function called with the following arguments:

– The value of the PID operand will be used as the PID argument.

– The sig argument is the value specified by the -s option, the -signal_name option, or the -signal_number option, or, if none of these options is specified, by SIGTERM.

The signaled process must belong to the current user unless the user is the super-user.

For example, to generate a thread dump on Solaris 10 system, get the process ID of the WebSphere Application Server and then issue the following command:

```
kill -3 <PID_OF_WAS_PROCESS>
```

Alternatively, it can be done as follows:

```
kill -SIGQUIT <PID_OF_WAS_PROCESS>
```

This will send the thread dump to the <WAS_LOG_DIR>/native_stdout.log file.

- **pmap**(1)

  The **pmap** command displays information about the address space of a process.

  Example 9-6 shows a sample output of the **pmap** command.

Example 9-6 A sample pmap output

```
# pmap -xs 9465
9465:   /opt/IBM/WebSphere/AppServer/java/bin/java -server -Dwas.status.socket
 Address  Kbytes     RSS    Anon  Locked Pgsz Mode   Mapped File
00010000      64      64       -       -  64K r-x--  java
0002E000      16      16       8       -   8K rwx--  java
00032000      56      56      56       -   8K rwx--    [ heap ]
00040000    3840    3840    3840       -  64K rwx--    [ heap ]
00400000   36864   36864   36864       -   4M rwx--    [ heap ]
196A8000      32      32      32       -   8K rwx-R    [ anon ]
196D8000      32      32      32       -   8K rwx-R    [ anon ]
19708000      32      32      32       -   8K rwx-R    [ stack tid=96 ]
19720000     128     128     128       -  64K rwx--    [ anon ]
.......lines omitted here.......
1993A000      24      24       -       -   8K r--s-  dev:32,30 ino:41276
FF33C000       8       8       8       -   8K rwx--  libsocket.so.1
FF350000       8       8       -       -   8K r-x--  libsched.so.1
FF360000       8       8       -       -   8K r-x--  libc_psr.so.1
FF370000       8       8       8       -   8K rwx--    [ anon ]
FF380000       8       8       -       -   8K r-x--  libdl.so.1
FF392000       8       8       8       -   8K rwx--  libdl.so.1
FF3A0000      24      24      24       -   8K rwx--    [ anon ]
FF3A8000      16      16       -       -   8K r-x--  libpthread.so.1
FF3B0000     192     192       -       -  64K r-x--  ld.so.1
FF3E0000      16      16       -       -   8K r-x--  ld.so.1
FF3EC000      16      16       -       -   8K r-x--  libthread.so.1
FF3F4000       8       8       8       -   8K rwx--  ld.so.1
FF3F6000       8       8       8       -   8K rwx--  ld.so.1
FFBDE000      24       -       -       -    - -----    [ anon ]
FFBE4000      48      48      48       -   8K rwx--    [ stack ]
FFBF0000      64      64      64       -  64K rwx--    [ stack ]
-------- ------- ------- ------- -------
total Kb 3742600  906000  776416       -
```

The **pmap -xs** command prints out an extended process address map, providing the virtual memory, the resident set size, and the anonymous (private) resident set size of each mapping. The virtual memory size is the committed virtual address space of the process. The tool also provides the page sizes for the pages in each mapping. Pages that are committed but not yet touched by the process are marked with a page size of "-". These pages may actually be in physical memory (because some other process touched the shared page), but the process has not yet touched that page. The tool counts such pages as part of the RSS of the process.

- **pstack**(1)

  The **pstack** command reports a hex+symbolic stack trace for each lwp in each process.

Example 9-7 shows a sample **pstack** output on a WebSphere Application Server process.

Example 9-7 Partial pstack output on WebSphere process

```
12230:  /opt/IBM/WebSphere/AppServer/java/bin/java -XX:MaxPermSize=256m -Decli
-----------------  lwp# 1 / thread# 1  --------------------
 ff3412c0 pollsys  (0, 0, ffbfbf98, 0)
 ff2dd068 poll     (0, 0, 7fffffff, 10624c00, 9a41bf, 26906fc0) + 7c
 fe990884 __1cIos_sleep6Fxi_i_ (ff0148f8, 0, 1, 163, 3a350, fefc0000) + 224
 fe9906b8 __1cIos_sleep6Fxi_i_ (7fffffff, ffffffff, 1, 7fffffff, f800c280,
fefc0000) + 58
 fe99064c __1cCosFsleep6FpnGThread_xi_i_ (6f7c, 6c00, feb00dcc, 1, 7, 4) + 25c
 feb094cc JVM_Sleep (8d4, ff00902c, 3a350, ff0151a4, 3af68, fefc0000) + 260
 f800c280 * java/lang/Thread.sleep(J)V+0
 f800c224 * java/lang/Thread.sleep(J)V+0
 f8005764 * com/ibm/ws/runtime/WsServerImpl.main([Ljava/lang/String;)V+76 (line
480)
 f8005c2c * com/ibm/ws/runtime/WsServer.main([Ljava/lang/String;)V+4 (line 59)
 f8000218 * StubRoutines (1)
 fe99a094
__1cJJavaCallsLcall_helper6FpnJJavaValue_pnMmethodHandle_pnRJavaCallArguments_pnGT
hread__v_ (1, 3a350, ffbfc
740, ffbfc520, 4, ffbfc718) + 5a0
 fea33a04
__1cKReflectionGinvoke6FnTinstanceKlassHandle_nMmethodHandle_nGHandle_inOobjArrayH
andle_nJBasicType_4ipnGThr
ead__pnHoopDesc__ (0, 1, 4, 1, ffbfc868, fefc0000) + 14a8
 feaffb98
__1cKReflectionNinvoke_method6FpnHoopDesc_nGHandle_nOobjArrayHandle_pnGThread__2_
(3a950, ffbfc864, 3a948, 3
a350, 3a94c, 0) + 268
 feafdbc4 JVM_InvokeMethod (3a40c, 0, 0, ffbfcac4, ff0151a4, 3a350) + 2bc
 fe670358 Java_sun_reflect_NativeMethodAccessorImpl_invoke0 (3a40c, ffbfca40,
ffbfcacc, 0, ffbfcac4, ffbfca58) + 10
 f800c280 *
sun/reflect/NativeMethodAccessorImpl.invoke0(Ljava/lang/reflect/Method;Ljava/lang/
Object;[Ljava/lang/Objec
t;)Ljava/lang/Object;+0
 f800c224 *
sun/reflect/NativeMethodAccessorImpl.invoke0(Ljava/lang/reflect/Method;Ljava/lang/
Object;[Ljava/lang/Objec
t;)Ljava/lang/Object;+0
```

```
 f8005874 *
sun/reflect/NativeMethodAccessorImpl.invoke(Ljava/lang/Object;[Ljava/lang/Object;)
Ljava/lang/Object;+87 (l
ine 39)
 f8005874 *
sun/reflect/DelegatingMethodAccessorImpl.invoke(Ljava/lang/Object;[Ljava/lang/Obje
ct;)Ljava/lang/Object;+6
 (line 25)
 f8005d3c *
java/lang/reflect/Method.invoke(Ljava/lang/Object;[Ljava/lang/Object;)Ljava/lang/O
bject;+111 (line 585)
 f8005874 *
com/ibm/wsspi/bootstrap/WSLauncher.launchMain([Ljava/lang/String;)V+341 (line 183)
 f8005764 * com/ibm/wsspi/bootstrap/WSLauncher.main([Ljava/lang/String;)V+17 (line
90)
 f8005764 *
com/ibm/wsspi/bootstrap/WSLauncher.run(Ljava/lang/Object;)Ljava/lang/Object;+22
(line 72)
 f8005d3c *
org/eclipse/core/internal/runtime/PlatformActivator$1.run(Ljava/lang/Object;)Ljava
/lang/Object;+219 (line
78)
 f8005d3c *
org/eclipse/core/runtime/internal/adaptor/EclipseAppLauncher.runApplication(Ljava/
lang/Object;)Ljava/lang/
Object;+103 (line 92)
 f8005874 *
org/eclipse/core/runtime/internal/adaptor/EclipseAppLauncher.start(Ljava/lang/Obje
ct;)Ljava/lang/Object;+2
9 (line 68)
 f8005874 *
org/eclipse/core/runtime/adaptor/EclipseStarter.run(Ljava/lang/Object;)Ljava/lang/
Object;+135 (line 400)
 f8005874 *
org/eclipse/core/runtime/adaptor/EclipseStarter.run([Ljava/lang/String;Ljava/lang/
Runnable;)Ljava/lang/Obj
ect;+60 (line 177)
```

- **pldd**(1)

  The **pldd** command lists the dynamic libraries linked into each process, including shared objects explicitly attached using **dlopen**(3DL). This command is very useful when you want to discover which libraries have been loaded by WebSphere Application Server, and when you are switching between different technology implementations, such as switching from aio to nio for WebContainer.

  Example 9-8 shows a sample **pldd** output.

Example 9-8 A sample pldd output

```
-bash-3.00# pldd 9465
9465:   /opt/IBM/WebSphere/AppServer/java/bin/java -server -Dwas.status.socket
/opt/IBM/WebSphere/AppServer/java/bin/java
/lib/libthread.so.1
/lib/libdl.so.1
/lib/libc.so.1
/platform/sun4v/lib/libc_psr.so.1
```

```
/opt/IBM/WebSphere/AppServer/java/jre/lib/sparc/server/libjvm.so
/lib/libsocket.so.1
/usr/lib/libsched.so.1
/usr/lib/libCrun.so.1
/lib/libm.so.1
/lib/libnsl.so.1
/lib/libm.so.2
/lib/libscf.so.1
/lib/libdoor.so.1
/lib/libuutil.so.1
/lib/libgen.so.1
/lib/libmd.so.1
/platform/sun4v/lib/libmd_psr.so.1
/lib/libmp.so.2
/opt/IBM/WebSphere/AppServer/java/jre/lib/sparc/native_threads/libhpi.so
/lib/nss_files.so.1
/opt/IBM/WebSphere/AppServer/java/jre/lib/sparc/libverify.so
/opt/IBM/WebSphere/AppServer/java/jre/lib/sparc/libjava.so
/opt/IBM/WebSphere/AppServer/java/jre/lib/sparc/libzip.so
/usr/lib/locale/en_US.ISO8859-1/en_US.ISO8859-1.so.3
/opt/IBM/WebSphere/AppServer/java/jre/lib/sparc/libnet.so
/opt/IBM/WebSphere/AppServer/java/jre/lib/sparc/libnio.so
/lib/librt.so.1
/lib/libaio.so.1
/lib/libsendfile.so.1
/opt/IBM/WebSphere/AppServer/bin/libWs60ProcessManagement.so
/lib/libpthread.so.1
/opt/IBM/WebSphere/AppServer/bin/libibmaio.so
/opt/IBM/WebSphere/AppServer/bin/libgetClasses.so
```

- **priocntl**(1M)

  The **priocntl** command displays or sets scheduling parameters of the specified process(es). It can also be used to display the current configuration information for the system's process scheduler or execute a command with specified scheduling parameters.

  Processes fall into distinct classes with a separate scheduling policy applied to each class. The currently supported process classes are Real-Time, Time-Sharing, and Interactive classes. The characteristics of these classes and the class-specific options available are described in the "Usage" section of the Solaris manual page for priocntl(1), which can be found at http://docs.sun.com/app/docs/doc/816-0210/6m6nb7mi6?a=view. With the appropriate permissions, the **priocntl** command can change the class and other scheduling parameters associated with a running process.

  Solaris introduced Fair Share (FSS) and Fixed (FX) process schedulers. When running only the application server on a system, it is good to run it in FX scheduler mode so that the WebSphere Application Server process gets fixed priority. When you run your environment or workload with and without these settings, you will notice a difference in the “involuntery context switches (icsw)” **mpstat** output. When this option is enabled, the number in icsw will be low, which will result in performance gain. In the lab environment for some applications, about a 10% improvement was observed. If the application can take adantage of fixed priority, using this scheduler will enhance the perofrmance.

  For example, the scheduler for a WebSphere Application Server process can be changed as follows:

  ```
  /usr/bin/priocntl -s -c FX -m 59 -p 59 -i pid <PID_OF_WAS_PROCESS>
  ```

- **ps**(1)

  The **ps** command displays information about processes. Normally, only those processes that are running with your effective user ID and are attached to a controlling terminal (see **termio**(7I)) are shown. Additional categories of processes can be added to the display using various options. In particular, the -a option allows you to include processes that are not owned by you (that do not have your user ID), and the -x option allows you to include processes without controlling terminals. When you specify both -a and - x, you get processes owned by anyone, with or without a controlling terminal. The -r option restricts the list of processes printed to running and runnable processes.

  The **ps** command displays, in tabular form, the process ID under PID, the controlling terminal (if any) under TT, the CPU time used by the process so far, including both user and system time, under TIME, the state of the process, under S, and finally, an indication of the COMMAND that is running.

  For example, this command can be run to discover what is the actual command that has been invoked to start the WebSphere Application Server process:

  ```
  /usr/ucb/ps -auwx <PID_OF_WAS_PROCESS>
  ```

  This is a very handy command to discover some of the JVM arguments as well as the class path, as shown in Example 9-9.

Example 9-9 Sample /usr/ucb/ps output

```
"/data2/washome/WebSphere/AppServer/java/bin/java" "-d64" "-XX:MaxPermSize=256m"
"-Declipse.security" "-Dosgi.install.area=/data2/washome/WebSphere/AppServer"
"-Dosgi.configuration.area=/data2/washome/WebSphere/AppServer/profiles/AppSrv01/co
nfiguration" "-Djava.awt.headless=true"
"-Xbootclasspath/p:/data2/washome/WebSphere/AppServer/java/jre/lib/ext/ibmorb.jar:
/data2/washome/WebSphere/AppServer/java/jre/lib/ext/ibmext.jar" "-classpath"
"/data2/washome/WebSphere/AppServer/profiles/AppSrv01/properties:/data2/washome/We
bSphere/AppServer/properties:/data2/washome/WebSphere/AppServer/lib/startup.jar:
/data2/washome/WebSphere/AppServer/lib/bootstrap.jar:/data2/washome/WebSphere/AppS
erver/lib/j2ee.jar:/data2/washome/WebSphere/AppServer/lib/lmproxy.jar:/data2/washo
me/WebSphere/AppServer/lib/urlprotocols.jar:/data2/washome/WebSphere/AppServer/dep
loytool/itp/batchboot.jar:/data2/washome/WebSphere/AppServer/deploytool/itp/batch2
.jar:/data2/washome/WebSphere/AppServer/java/lib/tools.jar"
"-Dibm.websphere.internalClassAccessMode=allow" "-Xms2800m" "-Xmx2800m"
"-Dws.ext.dirs=/data2/washome/WebSphere/AppServer/java/lib:/data2/washome/WebSpher
e/AppServer/profiles/AppSrv01/classes:/data2/washome/WebSphere/AppServer/classes:/
data2/washome/WebSphere/AppServer/lib:/data2/washome/WebSphere/AppServer/installed
Channels:/data2/washome/WebSphere/AppServer/lib/ext:/data2/washome/WebSphere/AppSe
rver/web/help:/data2/washome/WebSphere/AppServer/deploytool/itp/plugins/com.ibm.et
ools.ejbdeploy/runtime"
"-Dderby.system.home=/data2/washome/WebSphere/AppServer/derby"
"-Dcom.ibm.itp.location=/data2/washome/WebSphere/AppServer/bin"
"-Djava.util.logging.configureByServer=true"
"-Duser.install.root=/data2/washome/WebSphere/AppServer/profiles/AppSrv01"
"-Djavax.management.builder.initial=com.ibm.ws.management.PlatformMBeanServerBuild
er" "-Dwas.install.root=/data2/washome/WebSphere/AppServer"
"-Dpython.cachedir=/data2/washome/WebSphere/AppServer/profiles/AppSrv01/temp/cache
dir" "-Djava.util.logging.manager=com.ibm.ws.bootstrap.WsLogManager"
"-Dserver.root=/data2/washome/WebSphere/AppServer/profiles/AppSrv01" "-Xmn1200m"
"-XX:+AggressiveHeap" "-XX:+UseParallelGC" "-XX:ParallelGCThreads=2"
"-XX:MaxTenuringThreshold=3" "-XX:LargePageSizeInBytes=2m" "-XX:SurvivorRatio=20"
"-XX:+UseParallelOldGC" "-Dcom.ibm.ws.pm.batch=true"
"-Dcom.ibm.ws.pm.deferredcreate=true" "-Dcom.ibm.CORBA.FragmentSize=3000"
```

```
"-Dcom.ibm.ws.pm.useLegacyCache=false" "-Dcom.ibm.ws.pm.grouppartialupdate=true"
"-Djavax.xml.transform.TransformerFactory=org.apache.xalan.processor.TransformerFa
ctoryImpl"
"-Djavax.xml.parsers.SAXParserFactory=org.apache.xerces.jaxp.SAXParserFactoryImpl"
"-Djavax.xml.parsers.DocumentBuilderFactory=org.apache.xerces.jaxp.DocumentBuilder
FactoryImpl"
"-Dorg.apache.xerces.xni.parser.XMLParserConfiguration=org.apache.xerces.parsers.X
ML11Configuration"
"-Djava.security.auth.login.config=/data2/washome/WebSphere/AppServer/profiles/App
Srv01/properties/wsjaas.conf"
"-Djava.security.policy=/data2/washome/WebSphere/AppServer/profiles/AppSrv01/prope
rties/server.policy" "com.ibm.wsspi.bootstrap.WSPreLauncher" "-nosplash"
"-application" "com.ibm.ws.bootstrap.WSLauncher" "com.ibm.ws.runtime.WsServer"
"/data2/washome/WebSphere/AppServer/profiles/AppSrv01/config" "wsx12Node01Cell"
"wsx12Node01" "server1"
```

More information about this command can be found at:

http://docs.sun.com/app/docs/doc/816-5165/ps-1b?a=view

- **ndd**(1M)

  Solaris has this command to get or set the device driver configuration parameters. The **ndd** command gets and sets selected configuration parameters in some kernel drivers, which can be changed for performance reasons. Currently, **ndd** only supports the drivers that implement the TCP/IP Internet protocol family and each driver has different parameters visible using **ndd**. Since these parameters are usually tightly coupled to the implementation, they are likely to change from release to release. Some parameters may be read-only and cannot be changed. However, from time to time for different systems and implementations, Sun publishes a few of them to be tuned differently; users are encouraged to refer to Sun's Web site to discover the current details about such changes.

  Example 9-10 is an example of tuning Sun UltraSPARC T2 based systems.

Example 9-10 Recommended ndd tuning for Sun SPARC Enterprise T5220 server

```
ndd -set /dev/tcp tcp_conn_req_max_q 16384
ndd -set /dev/tcp tcp_conn_req_max_q0 16384
ndd -set /dev/tcp tcp_xmit_hiwat 131072
ndd -set /dev/tcp tcp_recv_hiwat 131072
ndd -set /dev/tcp tcp_naglim_def 1
```

You can get more information about the **ndd** command on your Solaris system by typing in the **man ndd** command.

9.3.3 Dynamic Tracing (DTrace)

DTrace is a comprehensive dynamic tracing facility that is built into Solaris 10 and can be used by administrators and developers to examine the behavior of both user programs and of the operating system itself. With DTrace, you can explore your system to understand how it works, track down performance problems across many layers of software, or locate the cause of aberrant behavior. It is safe to use on production systems and it does not require restarting either the system or applications.

DTrace can dynamically instrument the operating system kernel and user processes to record data at locations of interest, called *probes*. A probe is an event or activity to which DTrace can bind a request to perform a set of actions, like recording a stack trace, a time stamp, or the argument to a function. Probes are like programmable sensors scattered all over your Solaris

system in interesting places. DTrace probes come from a set of kernel modules called providers, each of which performs a particular kind of instrumentation to create probes.

Important: In a production environment, it is safe to use DTrace. There are many safeguards built into DTrace to make it safe to use in production systems. For example, DTrace will shut itself off if it realizes the script is affecting too much of the system processes.

Having said that, it is still important to make sure that the scripts are well written. It is a best practice to check the probe's effect on your scripts before running it in production.

See Chapter 38, "Performance Considerations", in *Solaris Dynamic Tracing Guide*, found at:

http://docs.sun.com/app/docs/doc/817-6223

DTrace introduces its own scripting language called D. D-scripts are used to interact with the Solaris Dynamic Tracing facility. Also, D-scripts are the portable way of developing standard scripts for collecting data from a live running system.

Figure 9-2 on page 332 explains the basic workings of the DTrace system. DTrace is a Solaris kernel feature. The user space consumers like **`dtrace`** and **`plockstat`** interact with the DTrace system using the **`libdtrace`** interface. Typically, users interact with the DTrace system using D-scripts. There are also C and Java APIs to interact with the DTrace subsystem.

By default, DTrace can only be run as the root user. Non-root users need some extra privileges to run DTrace. These are easily configured by updating the /etc/user_attr files with the necessary privileges. There are three privileges that are specific to DTrace:

- dtrace_proc:: Permits users to use the PID provider for process level tracing
- dtrace_user:: Permits users to use the profile and syscall provider to look at what their process is doing in the kernel
- dtrace_kernel:: Permits users to access almost all DTrace probes except the capability to look into other users' processes.

Edit the /etc/user_attr file with the following format:

```
user-name::::defaultpriv=basic,privilege
```

For example, you can allow the user named "joe" to have this privilege:

```
joe::::defaultpriv=basic,dtrace_proc,dtrace_user,dtrace_kernel
```

Once you have done this edit, the user needs to relogin to get the privileges.

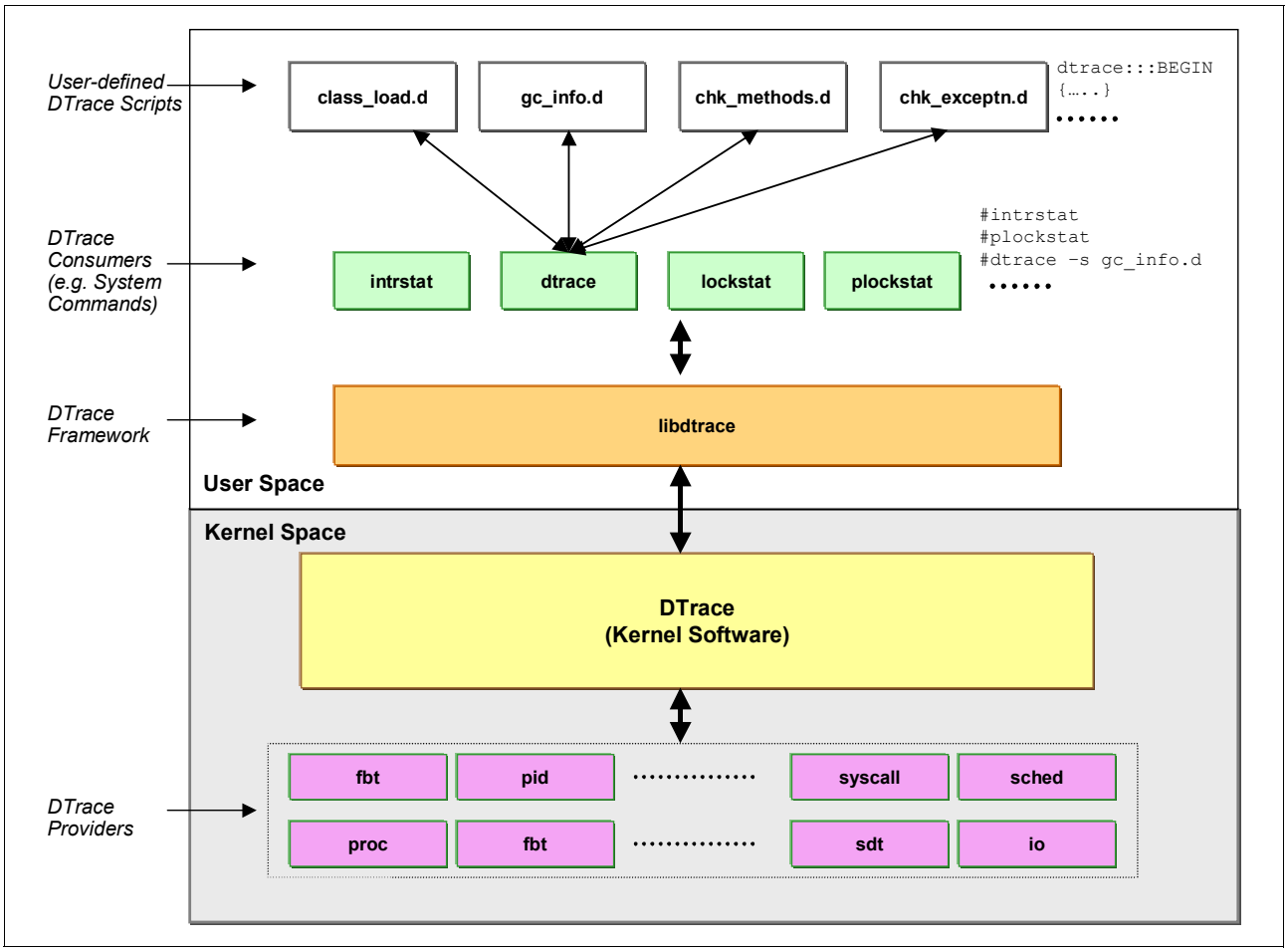


Figure 9-2 Overview of the DTrace architecture and components

Refer to http://docs.sun.com/app/docs/doc/817-6223 for more details on the D language and DTrace.

Introduction to D-scripts

First, we start with a brief discussion of the construction of a D-script. A D-script consists of a probe description, a predicate, and actions, as shown below:

```
probe description
/predicate/
{
   actions
}
```

As you saw earlier, probes are events in DTrace. In the D-script, you can subscribe for a set of one or more events and set up callbacks in the action section. The predicates are limitations that can be set when the actions are executed.

Here is an example of a simple D-script:

```
syscall::open:entry
/execname == "java"/
{
   trace(copyinstr(arg0));
}
```

Here the probe description is the `syscall::open:entry`, which describes the open system call. The predicate is `/execname == "java"/`. This checks to see if the executable that is calling the open system call is the Java process. The trace statement is the action that executes every time Java calls the open system call. D-scripts allow you to look at the arguments. In this case, the first argument of the open system call is the name of the file that is being opened. So this simple script will print out the file name that is opened by any Java process in the system, which includes those that are currently running and those that will be started after the script is run.

To stop the script, press ^c (Ctrl-C).

Probe description

Now let us look a little deeper. A probe is described with four tuples: the provider, module, function, and name.

- Provider: Specifies the instrumentation method to be used. For example, the syscall provider is used to monitor system calls while the io provider is used to monitor the disk I/O.
- Module and function: Describes the module and function you want to observe.
- Name: Typically represents the location in the function. For example, use the entry for name to set when you enter the function.

Note that wild cards like * and ? can be used. Blank fields are interpreted as wildcard. Table 9-1 shows a few examples.

Table 9-1 Examples of probe descriptions

Probe description	Explanation
`syscall::open:entry`	Entry into open system call
`syscall::open*:entry`	Entry into any system call that starts with open (open and open64)
`syscall:::entry`	Entry into any system called

Predicate

A predicate can be any D expression. The action is executed only when the predicate evaluates to true. A few examples are given in Table 9-2.

Table 9-2 Examples of predicates

Predicate	Explanation
`cpu == 0`	True if the probe executes on cpu0
`pid == 1976`	True if the PID of the process that caused the probe to fire is 1976
`execname != “sched”`	True if the process is not the scheduler (sched)
`ppid !=0 && arg0 == 0`	True if the parent process ID is not 0 and the first argument is 0

Action

The action section can contain a series of semi-colon (;) separated action commands. Examples are given in Table 9-3.

Table 9-3 Examples of actions

Action	Explanation
`printf()`	Print something using a C-style **`printf()`** command.
`ustack()`	Print the user level stack.
`trace`	Print the given variable.

Note: Predicates and action statements are optional. If the predicate is missing, then the action is always executed. If the action is missing, then the name of the probe that fired is printed.

Observing running processes

In this section, let us take a quick look at how to use DTrace to observe user processes. DTrace has a PID provider just for this purpose. The PID provider instruments a live running process. You can insert probes into function boundaries and in fact into any assembly line in a live running process. The name of the PID provider includes the process ID of the running process in which you are interested. A few examples of probe descriptions using the PID provider are given in Table 9-4.

Table 9-4 Examples of PID provider and associated probes

Example	Explanation
`pid2401:libc:malloc:entry`	An entry into the malloc function in libc for process ID 2401
`pid2608:a.out:main:return`	A return from the main for process ID 2608
`pid2608:a.out::entry`	An entry into any function in 2608 that is main executable
`pid2608:::entry`	An entry into any function in any library for process ID 2608

Let us look at a simple script using the PID provider:

```
#!/usr/sbin/dtrace -s
/*
```

The previous lines mean that the script that follows need to be interpreted using:

```
dtrace. D uses C-style comments.
*/
pid2401:libc::entry
{}
```

The following steps can be used to build a D-script using the PID provider:

1. The same script can be run from the command line using the -n option of **`dtrace`**:

   ```
   # dtrace -n pid2401:libc::entry
   ```

You can use this command or script by replacing 2401 with the PID of the process in which you are interested, such as WebSphere Application Server's Java process ID. As you notice, this is not too configurable.

2. If you modify the script to take the process ID as a parameter, your script will look like:

```
#!/usr/sbin/dtrace -s
pid$1:libc::entry
{
}
```

3. Now you can provide the process ID as an argument to your script. Note that **dtrace** will produce an error and not execute if you provide more than one argument to this script.

   You may have noticed that the output from this script went by too fast to be useful. D language has a wonderful construct called aggregate to collect all the details in memory and print out a summary. Aggregrations allow you to collect tables of information in memory. Aggregations have the following construction:

```
@name[table index(es)] = aggregate_function()
```

   For example:

```
@count_table[probefunc] = count() ;
```

   This aggregation will collect information into a table with the name of the function (probefunc is a built-in variable that has the name of the function). The aggregation function count() keeps track of the number of times the function was called. The other popular aggregation functions are average, min, max, and sum.

4. The following example adds aggregates in the program we are developing to show a table of summary of user functions:

```
#!/usr/sbin/dtrace -s
pid$1:::entry
{
   @count_table[probefunc]=count();
}
```

   This script will collect information into a table and will continue to run until you press ^c (Ctrl-C). Once you stop the script, DTrace will print out the table of information. Notice that you do not need to write any code to print the table; DTrace automatically does this for you.

   You may have noticed that the probes were created dynamically when running this script. The probes are disabled as soon as the script is stopped; you do not need to do anything special to disable these probes. In addition, DTrace will automatically clean up any memory it allocated; you do not need to write any code for cleanup.

5. You can easily modify this script to create a table with the function and library name by changing the index to `probemod` and `probefunc`.

```
#!/usr/sbin/dtrace -s
pid$1:::entry
{
   @count_table[probemod,probefunc]=count();
}
```

6. Next, find how much time is being spent in each function. This can be done through the built-in variable time stamp. You can create probes in the entry, return the functions, and calculate the time spent by showing the difference in timestamps from entry to return. The time stamp variable reports time in nanoseconds. Example 9-11 contains the modified script.

Example 9-11 Calculating the time spent

```
#!/usr/sbin/dtrace -s
pid$1:libc::entry
{
   ts[probefunc] = timestamp;
}
pid$1:libc::return
{
   @func_time[probefunc] = sum(timestamp - ts[probefunc]);
   ts[probefunc] = 0;
}
```

Note that the ts[] is an array, and D has automatically declared and initialized it for you. It is best practice to save space by setting the variable to 0.

7. This script works for most cases; however, there are a few corner case exceptions:
   - Exception 1: Monitor function entry and return

     Because you are creating the probes on a live running application, it is very likely that you were executing a function when you ran the D-script. Therefore, it is possible to see the return for a function for which you did not see an entry. This case is easily handled by adding a predicate `/ts[probefunc] != 0/` to the return probe section. This predicate will allow you to ignore the above corner case.
   - Exception 2: Multi-threaded applications

     The next corner case is a little more tricky and involves multi-threaded applications. There is a likely race condition where two threads could execute the same function at the same time. In this case, you need one copy of the ts[] for each thread. DTrace addresses this through the self variable. Anything that you add to the self variable will be made thread local.
8. The following script has been modified to handle these two corner cases:

```
#!/usr/sbin/dtrace -s
pid$1:libc::entry
{
   self->ts[probefunc] = timestamp;
}
pid$1:libc::return /self->ts[probefunc]/
{
   @func_time[probefunc] = sum(timestamp - self->ts[probefunc]);
   self->ts[probefunc] = 0;
}
```

Note: `/self->ts[probefunc]/` is the same as `/self->ts[probefunc] != 0/`.

For very large applications, the above script enables a large number of probes (possibly hundreds of thousands). Even though each probe is fairly light weight, adding hundreds of thousands of probes to a live running system will impact the performance of your application.

You can limit the number of probes you enable by modifying the probe description. See Table 9-5 for some examples.

Table 9-5 Modifying probe descriptions

Probe description	Explanation
`pid$1:libc::entry`	Limit to only a given library
`pid$1:a.out::entry`	Limit probes to non-library functions
`pid$1:libc:printf:entry`	Limit probes to just one function

9. Next, you see how to monitor a process from the time it starts until the end. DTrace allows you to do this using the $target variable and the -c option. This script will count the number of times libc functions are called from a given application:

```
#!/usr/sbin/dtrace -s
pid$target:libc::entry
{
   @[probefunc]=count();
}
```

10. Save this script to a file as libc_func.d and run it:

```
# libc_func.d -c "cat /etc/hosts"
```

You can easily replace `cat /etc/hosts` with the command of interest.

Observing Java programs

Now that you have seen how to observe a standard Solaris running process, let us take a look at how to observe a Java process. This is very useful, as a WebSphere Application Server process is a Java process. If you use the PID provider on the Java process, you will only be observing the JVM itself and not the Java program that runs in it. In this section, we will look at how to observe the Java process.

JDK 1.4.2 and 5.0

The *dvm provider* has been developed to observe a Java process running in JDK 1.4.2 or 5.0 (for example, WebSphere Application Server V6.0.2 or V6.1, respectively). You can download the provider from `https://solaris10-dtrace-vm-agents.dev.java.net`. It also has excellent information about using the provider. In this section, we will introduce the use of the dvm provider in Java code. These are generic examples and we provide more specific examples for WebSphere Application Server in the next section:

1. First, download the dvm.zip file from the following URL:

   `https://solaris10-dtrace-vm-agents.dev.java.net/servlets/ProjectDocumentList`

2. The dvmti agent utilizes the Java Virtual Machine Tool Interface (JVMTI), which only works with JDK 5.0 and later and provides the most detail in the DTrace probe arguments. Since WebSphere Application Server V6.1 relies on JVM from the JDK 5.0 version, you can rely on JVMTI.

   The dvmpi agent utilizes the Java Virtual Machine Profiler Interface (JVMPI), which works with 1.4.2 and 5.0, but the JVM PI has always been experimental and can be unreliable at times.

3. To use the dvm provider, you need to use the following command:

   ```
   java -Xrundvm[tp]i[:options] ...
   ```

   Where the options are:

   all Same as unloads,allocs,stats,methods.

   help Print help message.

   unloads Track class unloads.

   allocs Track object allocations and frees.

   stats Generate heap stats after each GC.

   methods Generate method entry exit events.

   exclude=name Exclude class names.

   You can also set the following shell environment variable to get the same effect:

   ```
   export JAVA_TOOL_OPTIONS="-Xrundvmti:all"
   ```

4. You also need to set the LD_LIBRARY_PATH shell variable to point to the location where you have downloaded the dvm provider libraries. You can use the following shell script to set the correct variable. We provide a specific example for WebSphere Application Server in “Setting up DTrace probes in WebSphere Application Server V6.1” on page 340:

   ```
   #!/usr/bin/sh
   cd your_own_dvm_storage_area
   if [ "`uname -p`" = sparc ] ; then
      export LD_LIBRARY_PATH_64=`pwd`/build/sparcv9/lib
      export LD_LIBRARY_PATH_32=`pwd`/build/sparc/lib
   else
      export LD_LIBRARY_PATH_64=`pwd`/build/amd64/lib
      export LD_LIBRARY_PATH_32=`pwd`/build/i386/lib
   fi
   ```

5. The DVM provider has the following probes:

   vm-init and vm-death
   JVM creation and death

   thread-start and thread-end
   Thread begin and end

   class-load and class-unload
   Class loading and unloading

   gc-start and gc-finish
   Garbage collection

   gc-stats gc statistics

   object-alloc and object-free
   Object allocation probes

   method-entry and method-return
   Method entry and return

6. Let us see a few examples of using the dvm provider. For these examples, we will assume you have a Java program on which to run these scripts. If not, use one of the Java demos.

   The script in Example 9-12 on page 339 will print the amount of time GC takes every time garbage collection happens.

Example 9-12 GC example

```
#!/usr/sbin/dtrace -s
dvm$1:::gc-start
{
        self->ts = timestamp;
}
dvm$1:::gc-finish
{
        printf("GC ran for %dnsec\n", timestamp - self->ts);
}
```

Note: This script takes the PID of the Java process as its first and only argument.

The script in Example 9-13 will print all the objects allocated and freed. It will also show you who allocated the object and the size of the object that was allocated.

Example 9-13 Object allocation

```
#!/usr/sbin/dtrace -qs
dvm$1:::object-alloc
{
  printf("%s allocated %d bytes\n",copyinstr(arg0), arg1);
}

dvm$1:::object-free
{
  printf("%s freed %d bytes\n",copyinstr(arg0), arg1);
}
```

Note: This script takes the PID of the Java process as its first and only argument.

The script in Example 9-14 prints the class name and method name of the method that is called. As we generally see many of these method calls in a typical Java process, we choose to look at this as an aggregate. Once you press ^c (Ctrl-C), the script will produce the sorted list and count of all the methods that were called.

Example 9-14 Method calls

```
#!/usr/sbin/dtrace -s

dvm$1:::method-entry
{
        @[copyinstr(arg0),copyinstr(arg1)] = count();
}
```

Note: Again, this script takes the PID of the Java process as its first and only argument.

The script in Example 9-15 shows a sorted list of the time taken to run the various methods in the Java process.

Example 9-15 Sorted list of time taken in method calls

```
#!/usr/sbin/dtrace -s
dvm$1:::method-entry
{
  self->ts[copyinstr(arg0),copyinstr(arg1)] = timestamp;
}
dvm$1:::method-return
{
  @ts[copyinstr(arg0),copyinstr(arg1)] = sum(timestamp -
                self->ts[copyinstr(arg0),copyinstr(arg1)]);
}
```

Using DTrace with WebSphere Application Server V6.1

In this section, we look at observing the WebSphere Application Server V6.1 using DTrace. We introduce the concept with the dvm provider in **"JDK 1.4.2 and 5.0" on page 337** and we will use it for analyzing a "live" WebSphere Application Server runtime environment.

Setting up DTrace probes in WebSphere Application Server V6.1

The following steps are necessary to enable DTrace probes in WebSphere Application Server:

1. Download the dvm probes from the java.net project page at:

   https://solaris10-dtrace-vm-agents.dev.java.net/

2. Create a directory (for example, /export/dvm) to unzip the file:

   ```
   mkdir -p /export/dvm
   cd /export/dvm
   unzip dvm.zip
   ```

3. We are interested in the WebSphere Application Server profile AppSrv01. Edit its start script, the ${WAS_HOME}/AppServer/profiles/AppSrv01/bin/startServer.sh file, to include the following lines before the last line with the startServer.sh script, as shown in Example 9-16 on page 341. Assume we put the dvm provider libraries in the /export/dvm directory:

   ```
   LD_LIBRARY_PATH=/export/dvm/build/sparc/lib
   JAVA_TOOL_OPTIONS="-Xrundvmti:all,dynamic=/tmp/dpipe"
   ```

   For an x86 system, change LD_LIBRARY_PATH to:

   ```
   LD_LIBRARY_PATH=/export/dvm/build/i386/lib
   ```

Note: In general, there is no need to add any flags or restart an application to use DTrace to observe its runtime behavior. You can look at the WebSphere Application Server using the PID provider without any changes on a live production system.

In many cases, you would like to observe not just the application server itself but the Java programs that run in the application server. In Java 5.0, you need an extra library to be able to observe the application and hence you need to modify the WebSphere Application Server's startup script **startServer.sh**. For more details, see **"JDK 1.4.2 and 5.0" on page 337**.

Example 9-16 Modified startServer.sh script for DTrace

```
#!/bin/sh
WAS_USER_SCRIPT=/opt/IBM/WebSphere6.1/AppServer/profiles/AppSrv01/bin/setupCmdLine
.sh
export WAS_USER_SCRIPT

# Set the dvm options and the pipe
JAVA_TOOL_OPTIONS="-Xrundvmti:all,dynamic=/tmp/ddvmpipe"
export JAVA_TOOL_OPTIONS

LD_LIBRARY_PATH=/export/home/DTrace/dvm/build/sparc/lib
export LD_LIBRARY_PATH

/opt/IBM/WebSphere6.1/AppServer/bin/startServer.sh "$@"
```

Note: Why do we need the pipe?

In some cases, the DVM provider can have a significant probe effect. There are ways to reduce the probe effect significantly using a pipe. The DVM provider looks into a named pipe to see if it should enable the probes. You can set the name of the named pipe by setting the dynamic variable in the -Xrundvmti option. For example:

```
JAVA_TOOL_OPTIONS="-Xrundvmti:all,dynamic=/tmp/dpipe"
```

You can turn on the probes by echoing a character into the file /tmp/dpipe. The probes can be turned off by echoing another character to the pipe.

4. Create the pipe file with the following command:

   ```
   mknod /tmp/dpipe p
   ```

5. You can start the server now with the following command:

   ```
   startServer server1
   ```

6. If the server startup time is too slow, you can turn off probes by echoing any character to the pipe using the following command:

   ```
   echo y > /tmp/dpipe
   ```

Example DTrace scripts

In this section, we look at a few example DTrace scripts for observing the WebSphere Application Server, as shown in Example 9-17 on page 342 to Example 9-22 on page 344.

Note: To run the following DTrace scripts, you need to set the execution permission to each script and provide arguments as needed by the script. For example, do the following command to set the execute permission on the script:

```
#chmod +x class_load.d
#./class_load.d
```

Example 9-17 Class loading Script: class_load.d

```
#!/usr/sbin/dtrace -s
dvm*:::class-load
{
   printf("%s loaded\n",copyinstr(arg0));
}
```

Example 9-18 GC Information script: gc_info.d

```
#!/usr/sbin/dtrace -qs

dvm*:::gc-start
{
 self->ts = timestamp;
}

dvm*:::gc-finish
{
 printf("GC ran for %d nsec \n", timestamp - self->ts);
}

dvm*:::gc-stats
{
 printf("gc-stats: used objects: %ld, used object space: %ld \n",
            arg0, arg1);
}
```

Example 9-19 Object Allocation by Classname script: obj_alloc.d

```
#!/usr/sbin/dtrace -qs

dvm*:::object-alloc
{
        @[copyinstr(arg0)] = sum(arg1);
}

BEGIN
{
        printf("%-50s %-10s\n","CLASSNAME","ALLOC SIZE");
}

END
{
        printa("%-50s %-10@d\n",@);
}
```

Important: Remember to press Ctrl-C (^c) to get the output from the script.

Example 9-20 Observing Monitors script: observ_mon.d

```
#!/usr/sbin/dtrace -qs

dvm*:::monitor*
{
        @[probename,copyinstr(arg0)]=count();
}
```

Note: This script will give you a count of the different monitor events and the thread name that caused these events. The events description are as follows:

- **monitor-contended-enter:** Probe that fires as a thread attempts to enter a contended monitor.
- **monitor-contended-entered**: Probe that fires when a thread successfully enters the contended monitor.
- **monitor-contended-exit**: Probe that fires when a thread leaves a monitor and other threads are waiting to enter.
- **monitor-wait**: Probe that fires as a thread begins a wait on a monitor through Object.wait().
- **monitor-waited**: Probe that fires when a thread completes an Object.wait().
- **monitor-notify**: Probe that fires when a thread calls Object.notify() to notify waiters on a monitor.
- **monitor-notify**: Probe that fires when a thread calls Object.notifyAll() to notify waiters on a monitor.

Example 9-21 Observing method calls script: observ_method.d

```
#!/usr/sbin/dtrace -s

dvm*:::method-entry
{
        @[copyinstr(arg0),copyinstr(arg1)]=count();
}

END
{
        printa("%30s::%30s %10@d\n",@);
}
```

Note: You need to press ^c (Ctrl-C) to see the results of the script. It will print the name of the class and method name followed by the number of times that method was called.

Example 9-22 Observing Exceptions script: observ_excep.d

```
#!/usr/sbin/dtrace -qs

dvm*:::exception-throw
{
        printf("%s thrown\n",copyinstr(arg0));
        jstack();
}
```

This script will print out all the exceptions that are thrown and the stacktrace from where the exception is thrown. Be aware this is all the exceptions that are thrown as well as the exceptions that are handled.

JDK 6.0 and the future

Although JDK 6.0 is not yet supported in any of the WebSphere products at this writing, it is worth mentioning the DTrace support in it. Unlike in JDK 1.4.2 and 5.0, JDK 6.0 has built-in DTrace support. There is no need to download special libraries or set special flags as described in the previous section. This simplifies the process of observing the Java applications. The name of the provider in JDK 6.0 will be called *hotspot*. The hotspot provider has similar probes to the dvm provider. The hotspot provider will greatly simplify using DTrace on a Java process.

In general, the types of probes in Java 6.0 are similar to the dvm probes. The name of the provider has been changed to hotspot.

For more details on the hotspot provider, go to:

http://java.sun.com/javase/6/docs/technotes/guides/vm/dtrace.html

In a later version of Java, the ability to add your own custom probes will be introduced. See http://blogs.sun.com/kamg/ for more details on future enhancements.

Additional DTrace resources

Here are a few good resources to get further details on DTrace:

1. DTrace community page: http://opensolaris.org/os/community/dtrace

   This is the best place to get information about DTrace. There are links to many resources on this page.

2. Solaris 10 Dynamic Tracing Guide: http://docs.sun.com/app/docs/doc/817-6223

   This is the official document for DTrace. It has comprehensive information about DTrace.

3. DTraceToolkit: http://opensolaris.org/os/community/dtrace/dtracetoolkit/

   The DTraceToolkit has over 200 useful D-scripts. This is a easy way to start using DTrace.

4. DTrace Java API: http://opensolaris.org/os/project/dtrace-chime/java_dtrace_api/

   This page contains details about the Java API for DTrace. This allows you to develop Java programs to interact with the Solaris 10 DTrace.

5. DTrace Visualization Chime: http://opensolaris.org/os/project/dtrace-chime/

   This is a an excellent tool that uses the Java API for DTrace to visualize the output from a D-script. The source code and details are also present at this site.

6. DTrace Web cast: http://www.snpnet.com/sun_DTrace/dtrace.html

   This Web cast is a quick introduction to DTrace for beginners.

7. Solaris Internals DTrace topic:
   http://www.solarisinternals.com/wiki/index.php/DTrace_Topics

   This is a compilation of the various DTrace resources focused on Solaris internals.
8. DTrace Bigadmin page: http://www.sun.com/bigadmin/content/dtrace/

   This is a compilation of the various DTrace resources.
9. Interesting blogs for DTrace:

   http://blogs.sun.com/roller/page/bmc
   http://blogs.sun.com/mws
   http://blogs.sun.com/roller/page/ahl
   http://blogs.sun.com/kamg/
   http://blogs.sun.com/brendan/

9.3.4 Sun Studio Analyzer

Developing any high performance application requires the right tools to be used, which would include the compiler, the platform that provides the infrastructure services to the application, and the right tools for peformance data collection and analysis.

On the Solaris platform, Sun provides the Sun Studio Analyzer (which we hereafter refer to as Analyzer), which helps you collect as well as analyze the performance data from a variety of applications. The applications can be based on anything from C, C++, and FORTRAN as well as profiling applications written in Java. The real advantage of this is that you do not need to have the source code of the application or compile and build it with any special option to use this tool.

WebSphere Application Server applications are just one type of Java application in regards to Analyzer. The data can be collected from any application written to run in the WebSphere Application Server environment. Analyzer has two parts:

- Collector

  The *Collector* part of the Sun Studio Analyzer tool collects performance data by profiling and tracing Java function calls. The data can include call stacks, microstate accounting information, thread-synchronization delay data, hardware-counter overflow data, MPI function call data, memory allocation data, and summary information for the operating system and the process.
- Analyzer

  The *Analyzer* part of the Sun Studio Analyzer tool displays the data recorded by the Collector in a GUI. It reads and processes the data collected by the Collector and displays various metrics of performance at the level of the program, the functions, the source lines, and the instructions. These metrics are classified into five groups:

  – Timing metrics
  – Hardware counter metrics
  – Synchronization delay metrics
  – Memory allocation metrics
  – MPI tracing metrics

  The Analyzer also displays the raw data in a graphical format as a function of time.

The details about how to set up and use this tool with WebSphere Application Server has been listed at:

http://developers.sun.com/sunstudio/articles/profiling_websphere.html

More details about the Analyzer can be found at:

http://developers.sun.com/solaris/articles/performance_tools.html

9.4 Java Virtual Machine (JVM) Performance Management

Performance is very critical for the success of any enterprise applications built on Java technology and it impacts all levels of the software stack as well as the hardware stack. Software stack performance involves server-side programs, client-side programs, Web services, operating systems, servers, and desktops. Hardware stack performance depends on the hardware design and technology.

In this section, we provide tools and techniques to help you improve the performance of Java Virtual Machine, which will in turn result in better performance of WebSphere Application Server.

The Sun Java performance team maintains the performance page at the Java Web site, and it contains up-to-date information. Readers are encouraged to read the latest performance tips and techniques at:

http://java.sun.com/performance/

9.4.1 WebSphere Application Server and Java versions

A WebSphere Application Server version is an implementation of a particular Java EE specification version with some additional features, while Java SE provides the Java Application Programming Interface (API), runtime environment (JRE), and other supporting facilities, such as tools. When tuning or planning for tuning of the WebSphere Application Server, it is necessary to understand the versions of WebSphere Application Server in terms of implementation of JDK/JVM they support. Like any other technology, the JVM technology also changes from version to version in which the performance tuning strategy may change. Therefore, it is important to note the version of JDK and then plan for the tuning, keeping in mind the tuning options available with the particular version of the JDK, as shown in Table 9-6.

Table 9-6 WebSphere and Java versions

WebSphere Application Server	Java EE	Java SE (JDK)
V6.1	1.4	5
V6.0.2	1.4	1.4.2
V5.1.1	1.3	1.4.2
V5.1	1.3	1.4.1

WebSphere Application Server is the core for other WebSphere products, such as Portal Server, Process Server, Customer Center, and Commerce Server.

Be aware of the specific versions bundled with WebSphere Application Server based products.

JDK versions for WebSphere

JDK bundled with WebSphere Application Server on Solaris (SPARC and x64) is Sun JDK with IBM modifications (For example, ORBs, XML processors, and so on).

JIT and JVM (Memory Management and GC) operations can have a significant impact on WebSphere Application Server performance.

With proper Fix Packs or WebSphere SR, JDK can be updated to later versions, as shown in Table 9-7.

Table 9-7 JDK versions for WebSphere

WebSphere Application Server	JDK version on Solaris
6.1.0.x	1.5.0_06
6.0.2.x[a]	1.4.2_08*
6.0.1.x	1.4.2_07
6.0	1.4.2_05
5.1.1.x	1.4.2_05
5.1.1.x	1.4.2_11
5.1	1.4.1_05
5.0.2.x	1.3.1_08
5.0.1	1.3.1_07
5.0	1.3.1_05

a. WebSphere Application Server support for Solaris Containers begins with V6.0.2.

For JDK versions on all supported platforms, consult the Web site at:

http://www.ibm.com/support/docview.wss?rs=180&context=SSEQTP&uid=swg27005002JavaSE5.0

9.4.2 JVM ergonomics (self tuning)

J2SE 5.0 includes a number of new features related to performance, including enhancements to existing features, specific optimizations for x86/x64, and enhancements to automatically tune the JVM based on the resources the system provides. When you start an application or server (for example, a stand-alone Java application or WebSphere) without specifying any of the JVM tuning options, the Java process starts with some of the JVM's default settings. The ergonomics feature is introduced to change some of these defaults based on the class of the machine, which is described in "Automatic selection of tuning parameters" on page 348. However, these default JVM settings may not be adequate for all applications; therefore, additional changes or tuning are necessary based on the need of the application. In this section, we discuss the ergonomics feature of JVM and how it can impact the initial behavior of the system. We discuss some of the relevant features in the context of WebSphere Application Server here; interested users are encouraged to check other performance enhancement features at:

http://java.sun.com/performance/reference/whitepapers/5.0_performance.html

In Java SE 5.0, ergonomics is used to automatically select the default JVM tuning settings for applications based on the platform and operating system on which they are deployed. These default values include the following:

- Garbage collector
- Heap size
- JVM (client or server)

These default settings attempt to optimize the JVM for some generic applications without having to tune the JVM using numerous command-line options.

Automatic selection of tuning parameters

For the Java SE 5.0, ergonomics machines are classified as either server-class or client-class.

- Server-class machine

  A machine is considered server-class if it has the following resources:

  – Two or more *logical* processors, as reported on Solaris by the **`psrinfo`** command

  – Two or more GBs of physical memory

  Note: Special cases:

  - The 32-bit Windows platform is never considered “server class”; thus, the default -client JVM is used.
  - The 64-bit JVMs always default to -server.

  The default values for a server-class machine are as follows:

  – The *`server`* Just-In-Time (JIT) compiler (can be changed to client JVM using the -client option)

  – The *`Throughput`* collector (ParallelGC)

  – Initial heap size (server or client JVM) = 1/64th of the physical memory, up to 1GB

  – Maximum heap size (server or client JVM) = 1/4th of the physical memory, up to 1GB

- Client-class machine

  A machine is considered client-class if it does not have the minimum resources of a server-class machine.

  The default values for a client-class machine are as follows.

  – The *`client`* JVM

  – The *`serial`* garbage collector

  – Initial heap size = 4 MB

  – Maximum heap size = 64 MB

  Note: Default values can always be overridden using command-line options.

Automatic tuning of the Throughput collector based on desired behavior

As of Java SE 5.0, the Throughput collector can be tuned based on an application’s desired maximum pause time and throughput. Command-line options can be used to specify the desired behavior of the collector. The automatic tuning will attempt to meet two goals: *maximum pause time* and *application throughput*.

Maximum pause time goal

Use the following command-line option to specify the maximum pause time goal:

`-XX:+MaxGCPauseMillis=`*n*

This option is effectively a suggestion to the Throughput collector that pause times should not exceed *n* milliseconds. The Throughput collector will attempt to achieve this goal by dynamically adjusting the heap size and other related garbage collection parameters. These adjustments may affect the throughput of the application, and it may be that the desired goal cannot be achieved.

Maximum pause time goals are applied separately to each generation. Generations may need to be decreased in size to meet the goal. There is no default value for the maximum pause time.

Throughput goal

The throughput goal is established as the percentage of total time spent performing garbage collection. Use the following command-line option to specify the throughput goal:

`-XX:+GCTimeRatio=`*n*

The ratio of GC time to application processing time is given by the following formula:

1 / (1 + *n*)

For example, `-XX:GCTimeRatio=9` sets a goal of 10% of the total to be spent performing garbage collection. The default goal is 1% (n=99). This is the total time spent garbage collecting all generations. If the goal cannot be achieved, the generation sizes are increased. Since the generations would then take longer to fill up, the applications can run longer between collections in an attempt to achieve the throughput goal.

Footprint goal

Once both throughput and maximum pause time goals have been met, the garbage collector begins to decrease the heap size. As the heap size is decreased one of the goals, typically the throughput goal, can no longer be met. At that point the collector begins tuning to achieve the goal that is currently not met.

Goal priorities

The maximum pause time goal has the highest priority, so the collector tries to achieve that goal first. The throughput goal will not be attempted until the maximum pause time goal is met. The footprint goal has the lowest priority. Only after the maximum pause time and the throughput goal have been achieved will the collector try to meet the footprint goal.

For more details on JVM tuning, refer to “Recommendations for tuning the JVM” on page 356.

9.4.3 Class data sharing

The class data sharing feature is aimed at reducing application startup time and footprint. The installation process loads a set of classes from the system jar file into a private, internal representation, then dumps that representation to a "shared archive" file. During subsequent JVM invocations, the shared archive is memory-mapped into the process address space, saving the cost of loading those classes in the traditional way and allowing much of the JVM's metadata for these classes to be shared among multiple JVM processes. For more information, see the Class Data Sharing documentation, found at:

`http://java.sun.com/j2se/1.5.0/docs/guide/vm/class-data-sharing.html`

JVM performance improvements in Java SE 5.0, including Class Data Sharing, have contributed to improved startup and footprint performance. Some of the startup experiments have shown that:

- Windows XP Java SE 5.0 starts 8% faster compared to J2SE 1.4.2.
- Sun Java Desktop System Java SE 5.0 starts 22% faster than J2SE 1.4.2.

Here we explain the memory footprint related changes between the J2SE 1.4.2 and Java SE 5.0 applications.

Measuring the real memory impact of a Java application is often quite difficult. Perhaps the first hurdle in understanding the footprint is that conventional system utilities, such as Task Manager on Windows, only tell part of the footprint story. Memory reported depends on whether application data and programs are read as conventional files or memory mapped files. In other words, often the true memory footprint of an application includes all the files that have been brought into the operating system's file system memory cache. Often memory pages that are shared by other processes or in the file system cache are not reported by conventional tools. Getting consistent footprint measurements is further complicated by accurately measuring the same moment in an application's lifetime. Clearly the longer an application operates, the more likely it is to perform classloading, compilation, or other activities that affect footprint. The great news for users of Java SE 5.0 is that despite adding massive new functionality Java engineering has actually pared down core JVM memory usage and leveraged Class Data Sharing to make the actual memory impact on your system lower than with J2SE 1.4.2.

Note: As of the writing of this book, Class Data Sharing is only applicable to the -client runtime; it is not available with the -server runtime.

9.4.4 JVM performance tuning

This section will describe the memory management function of the JVM, commonly referred to as *garbage collection*. Several types of garbage collectors are available with the Java SE 5.0; HotSpot JVM will be discussed and recommendations will be made as to which collectors will provide improved performance for various hardware platforms and application scenarios.

Introduction to garbage collector concepts

The memory management function contains the garbage collector and the allocator. It is responsible for allocating memory in addition to collecting garbage. Because the task of memory allocation is small, compared to that of garbage collection, the term garbage collection usually also means memory management. The garbage collector allocates areas of storage in the *heap*. These areas of storage define Java objects. When allocated, an object continues to be live while a reference (pointer) to it exists somewhere in the JVM; therefore, the object is *reachable*. When an object ceases to be referenced from the active state, it becomes *garbage* and can be reclaimed for reuse. When this reclamation occurs, the garbage collector must process a possible finalizer and also ensure that any internal JVM resources that are associated with the object are returned to the pool of such resources.

When the JVM cannot allocate an object from the heap because of a lack of contiguous space, a memory allocation fault occurs, and the garbage collector is invoked. The first task of the garbage collector is to collect all the garbage that is in the heap. This process starts when any thread calls the garbage collector either indirectly as a result of allocation failure, or directly by a specific call to **`System.gc()`**. Usually the full heap or part of it is garbage collected when it becomes full or exceeds a threshold limit of occupancy (usage).

Garbage collection proceeds through three phases:

- Mark

  In this phase, all live or reachable objects are found. Any object that is not reachable is considered garbage and will be removed during the next phase.

- Sweep

  During this phase, the garbage found in the Mark phase is removed from the heap.

- Compaction

  If after the Sweep phase there is still not enough contiguous free memory, the heap can be compacted. During compaction all live objects are moved to one end of the heap.

The job of fulfilling a request for space on the heap requires that a large enough block of contiguous memory be available. If the space is available but not contiguous due to heap fragmentation, then two things usually happen: The heap can be *compacted*, meaning that all live objects are moved to one end of the heap leaving the other end free to be allocated. If the process of compaction still does not create enough space, then the heap can be expanded up to its configured maximum heap size.

Garbage collector designs

Garbage collectors can be implemented in a number of ways in order to achieve optimum performance:

- Serial

  During serial collection, only one phase can happen at a time. Even if the machine has multiple CPUs, only one CPU will be performing the collection.

- Parallel

  During parallel collection, the job can be sub-divided into multiple parts that can be performed at the same time (in parallel) resulting in faster collection.

- Concurrent

  Concurrent collection means that collection happens simultaneously (concurrent) with application execution. Otherwise applications would experience a pause in execution during the collection.

- Stop-the-world

  If the collection is stop-the-world, then application execution is paused until the collection completes.

- Compacting

  During compaction, all live objects are moved to one end of the heap, freeing the rest of the memory for new allocation requests.

- Non-compacting

  If the collector is non-compacting, the space freed from removing the garbage remains *in-place*. Live objects are not moved as in compaction, resulting in faster collection, but potentially more fragmentation.

- Copying

  A copying collector actually copies the live objects to another area of memory, making the original object locations free to be allocated for new objects. Additional time and extra space are required for this method to be effective.

JVM performance metrics

There are a number of metrics that can be gathered in ordered to monitor and evaluate the performance of garbage collection. These metrics are defined as follows:

- Throughput

  The amount of a JVM's total time that is spent processing applications and *not* spent doing garbage collection. It is measured over long periods of time, and the ideal throughput should be 80% or greater.

- Garbage collection overhead

  The opposite of throughput. Ideally, garbage collection overhead should be a small percentage of a JVM's total time.

- Pause time

  The amount of time during which application processing is stopped while garbage collection is happening. For most applications and real-time applications in particular, pause time should be minimized.

- Frequency of GCs

  How often garbage collections occur.

- Java Heap Memory footprint

  Typically refers to the size of the Java heap.

- Promptness

  Measures the interval between the time an object is marked as garbage and the time the object's memory becomes available for reuse.

Generational garbage collection

The technique known as generational collection partitions the heap into generations. The generations are separate parts of the heap that contain objects with different ages. A common configuration is to use two generations: a young generation (Eden) for newly created objects, and an old or tenured generation for older objects.

Various algorithms for performing garbage collection can be implemented to take advantage of the different characteristics of the objects in each generation. Generational collectors can take advantage of the following observations about Java applications:

- The majority of objects on the heap are not long lived. They are only referenced for a short period of time and then become garbage.
- Very few references from older to younger objects are used.

The frequency of young generation garbage collections is high, but the young generation footprint can be kept small, resulting in faster collections. The collections are more efficient because most of the objects are garbage.

When an object in the young generation survives a few garbage collections, it can be promoted (tenured) to the older generation, as displayed in Figure 9-3 on page 353. The older generation usually has a larger chunk of the heap than the younger generation so it will fill up over a longer period of time. The frequency of old generation collections is low, but they will require a longer time to complete.

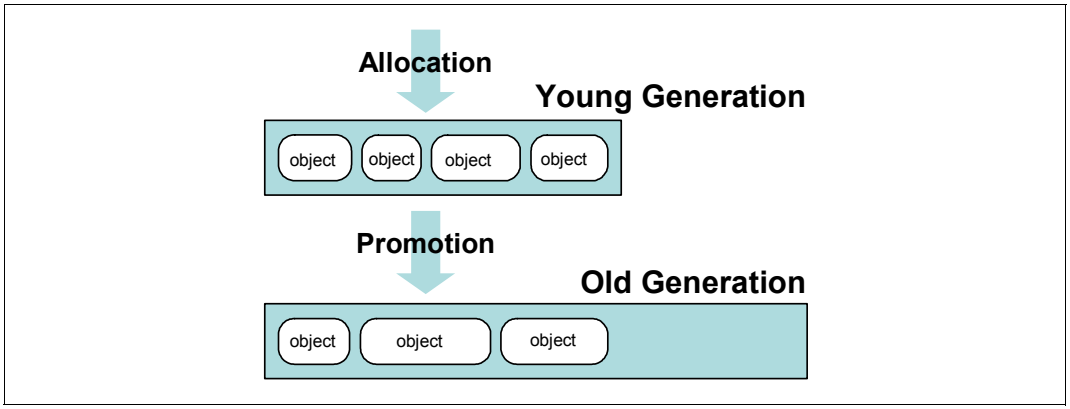


Figure 9-3 Generational garbage collection

Young generation versus old generation garbage collection

The algorithm implemented for the young generation collector is optimized for speed because the frequency of collections is high, while the algorithm implemented for the old generation collector is optimized to use heap space more efficiently since the old generation will use most of the heap space.

Note: As of Java SE 5.0 Update 6, the combination of a parallel young generation and parallel old generation can be enabled with -XX:+UseParallelOldGC.

Java SE 5.0 Sun HotSpot JVM garbage collectors

The Sun HotSpot JVM supports the following garbage collectors as of Java SE 5.0 update 6:

- Serial collector
- Throughput collector
  - Parallel collector
  - Parallel Compacting collector
- Concurrent Mark-Sweep (CMS) collector

This section will briefly describe each collector and provide recommendations for when each collector should be used.

For a more detailed discussion of the HotSpot collectors, refer to the following white paper *Memory Management in the Java HotSpot Virtual Machine*, found at:

`http://java.sun.com/j2se/reference/whitepapers/memorymanagement_whitepaper.pdf`

HotSpot generations

In the HotSpot JVM, memory is partitioned into three generations.

- Young generation

  Most objects are initially allocated in the young generation unless they are too large.
- Old generation

  The old generation contains objects that have survived a few young generation garbage collections and have been promoted to the old generation. Some large objects may have been directly allocated to the old generation.

- Permanent generation

  The permanent generation typically contains objects that may survive the entire life of the JVM, but may also benefit from being managed by the garbage collector. Such objects include the following:

  – Objects describing classes and methods

  – Actual classes and methods

The young generation is further divided into three parts: an area called *Eden* and two smaller areas referred to as *survivor spaces*, as shown in Figure 9-4.

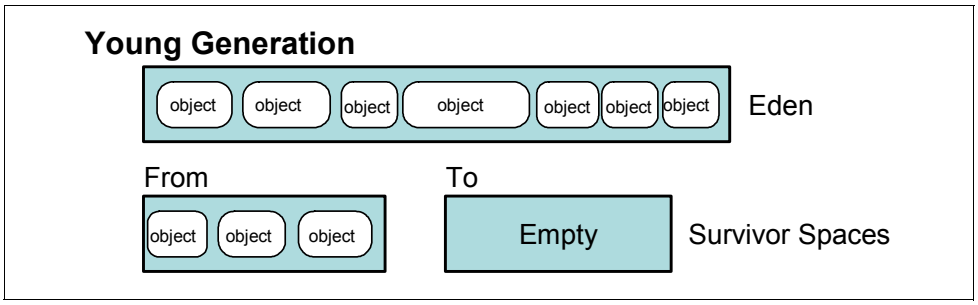


Figure 9-4 Young generation memory areas

As discussed earlier, most objects begin life in Eden with some being directly allocated to the old generation. The survivor spaces contain objects from Eden that have survived at least one garbage collection. At any point in time, one of the survivor spaces (called *From* in the figure) contain such surviving objects and the other survivor space is empty and remains empty until another garbage collection occurs.

Types of garbage collections

Eventually all generations will fill up with objects. So there are a number of collection types that can occur.

- Young generation collection (minor collection)

  When the young generation fills up, only the young generation is collected.

- Full collection (major collection)

  When the old or permanent generations fill up, then a full collection is performed. All generations are collected starting with the young generation. The young generation is collected using the young generation algorithm. The old and permanent generations are collected with an *old generation collection algorithm*. If compactions occur, each generation is compacted separately.

Serial collector

A serial collector collects all generations in sequence using only one CPU. This is a *stop-the-world* collection, meaning that all application execution is paused until the collection is complete.

As shown in Figure 9-5 on page 355, live objects in the Eden space are copied to the empty survivor space (called *To* in the figure). If an Eden object is too large for the survivor space, it is copied directly to the old generation. Live objects in the other survivor space (called *From* in the figure) that are comparatively young are also copied to the empty survivor space, while the comparatively older objects are copied to the old generation. The objects marked with an X are garbage and do not get copied.

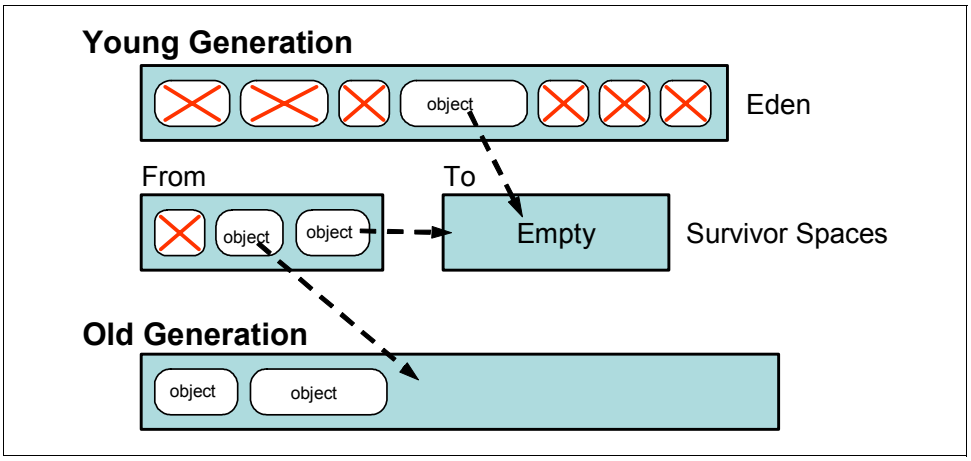


Figure 9-5 Serial young generation garbage collection

When the garbage collection is complete, the Eden space and the previously occupied survivor space are now empty, as shown in Figure 9-6, and the two survivor spaces “flip” or switch roles.

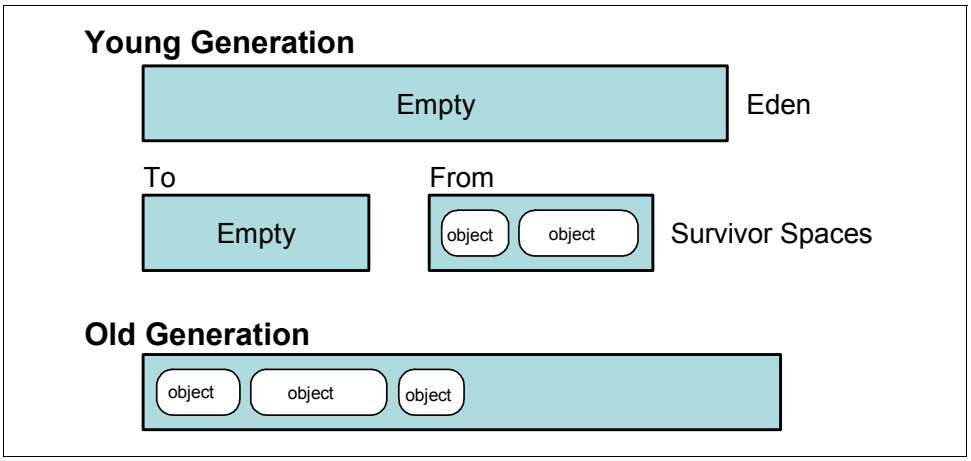


Figure 9-6 After young generation garbage collection

Use the serial collector if the following conditions are true:

- Most of your applications are running on a client-style machine.
- You have applications that do not require short pause times.
- You have applications with small Java heap requirements (for example, less than 256 MB)

In Java SE 5.0, the serial collector is the default on machines that are not server-class. You can explicitly select the serial collector by using the -XX:+UseSerialGC command-line option.

Parallel collector

The parallel collector is also referred to as the *throughput collector* and it is designed to take advantage of a machine with multiple CPUs and a large amount of physical memory. Multiple CPUs are used to perform garbage collection tasks in parallel.

Use the parallel collector if the following conditions are true:

- The machine has multiple CPUs and plenty of physical memory.
- Applications do *not* have any constraints on pause times. For example:
  - Billing
  - Payroll
  - Scientific computing

In addition, you should use the parallel collector if the serial collector is not giving you short enough pause times on minor collections or if the GC impact (which is a function of frequency and pause time) is too high. Furthermore, we recommend a large young generation for that parallel collector, or you will end up with threads contending for work, which could reduce efficiency.

In Java SE 5.0, the parallel collector is the default on server-class machines. You can explicitly select it with the -XX:+UseParallelGC command-line option.

Parallel compacting collector

The parallel compacting collector was new in Java SE 5.0 Update 6. It differs from the parallel collector in that it uses a new algorithm for performing the old generation garbage collection.

Use the parallel compacting collector if the following conditions are true:

- The machine has multiple CPUs and plenty of physical memory.
- To further reduce full gc times for throughput oriented applications.

The parallel compacting collector is not yet a default on server-class machines. If you want to use it, you must explicitly select the parallel compacting collector by using the -XX:+UseParallelOldGC command-line option.

Concurrent Mark-Sweep (CMS) collector

This collector is more about reducing pause times than it is about preserving throughput in the face of full gcs. If you are looking for maximum throughput regardless of pause times, then UseParallelGC and UseParallelOldGC are the appropriate garbage collectors. If you are trying to reduce pause times while trying to preserve the throughput at the same time, CMS may be the right collector.

Use the CMS collector if the following conditions are true.

- The machine has multiple CPUs (for example, less than four).
- The application needs shorter garbage collection pauses.
- The application can share processor resources with the collector while the application is executing.
- Applications have large and long-lived data sets.

The CMS collector is not the default on server-class machines. If you want to use it, you must explicitly select the CMS collector by using the -XX:+UseConcMarkSweepGC command-line option. For pause time sensitive applications, running the concurrent collector in incremental mode may be useful. To enable incremental mode, use the -XX:+CMSIncrementalMode option.

Recommendations for tuning the JVM

The ergonomics described in 9.4.2, “JVM ergonomics (self tuning)” on page 347 provides default values for JVM tuning parameters, such as heap size, garbage collector type, and JVM options. The initial recommendation is to accept the default values and test your application with this as a baseline. If you find that the application performs reasonably well in

terms of required throughput and acceptable pause times, you may not need to further optimize the tuning.

In many cases, things may not be this simple. If load testing indicates performance problems associated with garbage collection, you should consider the following actions:

- Set the minimum and maximum values of the Java heap size.
- Evaluate whether the default garbage collector is appropriate for the application.
- If necessary, explicitly select another garbage collector.
- Load test the application and evaluate whether the performance is acceptable with the new garbage collector.

Section 9.4.6, “Tools for performance tuning” on page 364 will describe tools for measuring and analyzing performance. The results produced by these tools will enable you to better select alternative tuning options.

Several tables of “Key garbage collection options”, showing commonly used settings, is provided in 9.4.5, “Key options related to garbage collection” on page 361.

Attention: Avoid the tendency to overtune or overoptimize, because your environment is sure to change over time. Changes can be expected in applications, application data sets, system hardware, and possibly the implementation of the garbage collector itself.

Selecting a different garbage collector

Refer to “Types of garbage collections” on page 354, which describes the garbage collectors and their recommended usage. If the results from load testing your application indicate that a garbage collector other than the default should be used, you should explicitly select another using the appropriate command-line options. Refer to Table 9-8 on page 361 for more details.

Heap sizing recommendations

Sometimes, changing the default heap size may have the desired effect without changing the garbage collector. If the results of load testing your application indicate poor performance due to insufficient heap size, or the applications are throwing `OutOfMemoryError` exceptions, you may need to modify the default heap sizes. Refer to Table 9-10 on page 362 for the command-line options associated with heap size. A sufficiently sized heap is critical to ensure efficient garbage collection. On the other hand, if you are experiencing `OutOfMemoryError` exceptions, it may be because the application itself is having memory leaks. You should then analyze your application for unintentional object retention. Refer to 10.4.3, “Troubleshooting memory leaks” on page 443 for more information.

Strategy for tuning the parallel collector

If you are using the parallel collector or parallel compacting collector, you should specify a throughput goal that is appropriate for your application. You can follow these guidelines:

- Do not select a maximum heap size value (unless you know that the default is not sufficient for your application).
- Let the heap size expand or shrink to support your throughput goal.
- Expect oscillations in heap sizes during startup and changes in application behavior.

If the heap size expands to its maximum, this usually means that the throughput goal cannot be achieved with that maximum heap size. Then, you can follow these guidelines:

- Set the maximum heap size close to total physical memory. (Be careful not to cause swapping of the application.)
- Test the application with the new max heap size.
- If the throughput goal is still not achieved, evaluate whether the goal is to high for the available memory.

If the throughput goal can be achieved, but pause times are too long, you can follow these guidelines:

- Select a maximum pause time goal.
- Expect that achieving a maximum pause time goal may mean that the throughput goal will not be achieved.
- Consequently, choose values that are a reasonable compromise for your application.

Trying to satisfy both goals will cause oscillations in the heap size, even after the application appears to reach a steady state. There are opposing forces at work. To achieve a throughput goal may require a larger heap, while the goal to achieve a maximum pause time and a minimum footprint may require a smaller heap.

Note: There are times when manually tuning the size of the young generation and the size of the survivor spaces and the number of GC threads will have significant impact. There are also times when making the initial and maximum heap sizes the same (for example, -Xmx == -Xms) is appropriate to use, instead of letting the heap grow and shrink.

Choosing a garbage collector for young and old (tenured) generations

Use the *Serial Copying Collector* (Default) for co-locating multiple instances of WebSphere Application Server (JVMs) on multi-CPU servers:

-XX:+UseSerialGC

Note: The serial collector is not the default on server class machines unless -client is specifically added to the command line.

Use the *Parallel* or *Throughput Collector* for better performance on multi-CPU servers (For example, T2000 or T5220):

- -XX:+UseParallelGC.
- Set the proper number of threads for parallel garbage collection (for example, -XX:ParallelGCThreads=<threads>, where <threads> is equal to total number of CPUs on the system or hardware threads on a multi-core system).

Use the Concurrent Low Pause Collector for constant response time on multi-CPU servers:

-XX:+UseConcMarkSweepGC

Use the *Incremental Low Pause Collector* for constant response time on servers that have few processors:

-Xincgc

Note: Do not use -XX:+UseTrainGC, as it will be obsoleted in Java SE 6.

The *AggressiveOpts* option turns on point performance compiler optimizations that are available in v1.5.0_06:

-XX:+AggressiveOpts

Use *Fixed Size Young Generation* to better handle stress. For example:

- -XX:NewSize=512m
- -XX:MaxNewSize=512m (for 1.3) -Xmn512m (for 1.4)

Note: If you are using a fixed size young generation, you should also be using a fixed sized heap (that is, -Xms == -Xmx). It is not strictly necessary, but you should also set NewSize == MaxNewSize unless -Xms == -Xmx.

Permanent Generation:

- -XX:PermSize=<initial size>
- -XX:MaxPermSize=<max size>

Reduce Full GC cycle time:

- See the GC Tuning Guide for details:

  http://java.sun.com/docs/hotspot/gc5.0/gc_tuning_5.html
- Choose other suitable options. Note that the following values are shown as an example:

  -XX:MaxTenuringThreshold=<value> (for example, value=3)

  -XX:SurvivorRatio=<value> (for example, value=8)

WebSphere Application Server V6.1 JVM tuning parameters on Solaris 10

In order to simplify WebSphere Application Server V6.1 performance tuning on the Solaris 10 platform, one needs to understand all the tuning knobs that are available with the version of Java that is shipped with WebSphere Application Server. Once you know the version details, you can take a look at the tunable parameters that are available and take corrective action accordingly. All the tunable parameters options are listed here:

http://java.sun.com/javase/technologies/hotspot/vmoptions.jsp

Before choosing the command-line flags, you should monitor and observe the behavior of the JVM under commonly expected application workloads or in stressed conditions. Then, analyze the performance data and change the default JVM configuration(s) as needed.

The following list shows the typical JVM tuning parameters that can be used for WebSphere Application Server V6.1:

initialHeapSize This is a WebSphere specific JVM option and is similar to the -Xms option. The values are numeric in MB. For example, a value of 2048 MB would mean 2 GB of memory for the initial JVM heap.

maximumHeapSize This is a WebSphere specific JVM option and is similar to the -Xmx option. The values are numeric in MB. For example, a value of 2048 MB would mean 2 GB of memory for the maximum JVM heap.

-server Sets the Java runtime environment to use the server optimized virtual machine. Generally, “server” is recommended for high throughput Application Servers. Due to the ergonomics feature, this may be the default option.

-Xmn<value & unit> Size of the young generation. Sets both the NewSize and MaxNewSize options to the same values (for example, -Xmn1024m). When using the throughput collector, we can set this value to around 50% of the Java heap size as a starting point for tuning JVM. Based on GC logs (for example, printing out GC details), you can further adjust this value for an optimal setting.

-XX:-ScavengeBeforeFullG
Do young generation GC prior to a full GC.

-XX:MaxTenuringThreshold
Specifies how many collections an object can remain in while it is in the young generation, after which it is promoted to the old generation.

-XX:+UseParallelGC Use parallel garbage collection for scavenges.

-XX:ParallelGCThreads=n
This option is used to specify a different number of GC threads from the default.

Note: The number of GC threads setting will depend on the underlying processor architecture of the deployment system. On the Chip Multithreading processor based (for example, UltraSPARC-T1 and T2) systems, you should *not* set the threads higher than one quarter of the available hardware threads. For example, a system with the UltraSPARC-T2 (8-core) processor has 64 hardware threads; thus, the maximum recommended value is -XX:ParallelGCThreads=16.

On other processors, the maximum recommended value is the total number of *"logical" processors* on the system (for example, reported by the **`psrinfo`** command). For example, the threads value is 8 on a system with four UltraSPARC IV+ (Dual Core) processors (for example, =8 processors). Setting the number of GC threads too high would have an adverse effect on overall system performance.

In addition, you need to consider the number of WebSphere instances that you are running on the system. Since this thread setting is for each JVM, you must distribute these threads accordingly (that is, the sum of the GC threads for all WebSphere instances is no more than the number of *processors*).

-XX:+AggressiveHeap
Available in JDK 1.4.2, this parameter attempts to set various parameters based on the system's physical memory and number of processors, but going forward, you should use AggressiveOpts instead.

Attention: We strongly recommend using -XX:+AggressiveOpt instead of -XX:+AggressiveHeap.

-XX:MaxPermSize Size of the permanent generation. (This is for Sun JVM 5.0 and newer; 64-bit VMs are scaled 30% larger, 1.4 amd64: 96m, and 1.3.1 -client: 32m.) It is important to note here that this value is further scaled up with WebSphere Application Server and the default setting sets it to 256m on a 32-bit platform.

-XX:PermSize It is the initial size of the permanent generation and is used along with the -XX:MaxPermSize flag.

-XX:+UseParallelOldGC

Use parallel garbage collection for the full collection. Enabling this option automatically sets -XX:+UseParallelGC. (Introduced in 5.0 update 6.)

Using a combination of the above flags can lead you to your desired system throughput. In some cases, JVM tuning can be simplified using just a few flags to achieve similar throughput:

-XX:+AggressiveOpts

Turn on point performance compiler optimizations that are expected to be the defaults in upcoming releases. (Introduced in 5.0 update 6.)

-XX:+UnlockDiagnosticVMOptions -XX:-EliminateZeroing

These two options must be used with +AggressiveOpts for optimizing WebSphere Application Server. Because the EliminateZeroing option is considered a diagnostic VM option, you must set +UnlockDiagnosticVMOptions first in order to use EliminateZeroing. This -EliminateZeroing option disables the initialization of newly created character array objects.

Besides JVM tuning, you also need to invest the time in setting the WebSphere thread pool sizes appropriately.

9.4.5 Key options related to garbage collection

A number of command-line options can be used to select a garbage collector, specify heap or generation sizes, modify garbage collection behavior, and obtain garbage collection statistics. This section shows some of the most commonly used options. Note: The numbers you specify can end with “m” or “M” for megabytes, “k” or “K” for kilobytes, and “g” or “G” for gigabytes.

Garbage collector selection

Table 9-8 displays the different Garbage collector types in JDK 5.

Table 9-8 Garbage collector types

Option	Garbage collector selected
–XX:+UseSerialGC	Serial
–XX:+UseParallelGC	Parallel
–XX:+UseParallelOldGC	Parallel compacting
–XX:+UseConcMarkSweepGC	Concurrent mark–sweep (CMS)

Garbage collector statistics

Table 9-9 displays the options to produce the garbage collector output.

Table 9-9 GC log output

Option	Description
–XX:+PrintGC	Outputs basic information at every garbage collection.
–XX:+PrintGCDetails	Outputs more detailed information at every garbage collection.
–XX:+PrintGCTimeStamps	Outputs a time stamp at the start of each garbage collection event. Used with –XX:+PrintGC or –XX:+PrintGCDetails to show when each garbage collection begins.
-XX:+PrintGCApplicationConcurrentTime	Measures the amount of time the applications runs between collection pauses.
-XX:+PrintGCApplicationStoppedTime	Measures the length of the collection pauses.s

Heap and generation sizes

Table 9-10 displays the initial and maximum heap size setting options.

Table 9-10 Initial and maximum heap sizes

Option	Default	Description
–Xms*n*	See 9.4.2, “JVM ergonomics (self tuning)” on page 347.	Initial Heap size, in bytes, of the heap.
–Xmx*n*	See 9.4.2, “JVM ergonomics (self tuning)” on page 347.	Maximum Heap size, in bytes, of the heap.
–XX:MinHeapFreeRatio=*minimum* –XX:MaxHeapFreeRatio=*maximum*	40 (min). 70 (max).	Target range for the proportion of free space to total heap size. These are applied per generation. For example, if the minimum is 30 and the percent of free space in a generation falls below 30%, the size of the generation is expanded so as to have 30% of the space free. Similarly, if maximum is 60 and the percent of free space exceeds 60%, the size of the generation is shrunk so as to have only 60% of the space free.
–XX:NewSize=*n*	Platform dependent.	The default initial size of the new (young) generation, in bytes.

Option	Default	Description
–XX:NewRatio=*n*	2 on client JVM and 8 on server JVM.	Ratio between the young and old generations. For example, if n is 3, then the ratio is 1:3 and the combined size of Eden and the survivor spaces is one fourth of the total size of the young and old generations.
–XX:SurvivorRatio=*n*	32.	Ratio between each survivor space and Eden. For example, if n is 7, each survivor space is one–ninth of the young generation (not one–eighth, because there are two survivor spaces).
–XX:MaxPermSize=*n*	Platform dependent.	Maximum size of the permanent generation.

Options for the Parallel and Parallel compacting collectors

Table 9-11 displays options for the Parallel and Parallel compacting collectors.

Table 9-11 Parallel options

Option	Default	Description
–XX:ParallelGCThreads=*n*	The number of CPUs	Number of garbage collector threads.
–XX:MaxGCPauseMillis=*n*	No default	Indicates to the collector that pause times of n milliseconds or less are desired.
–XX:GCTimeRatio=*n*	99	Number that sets a goal that 1/(1+n) of the total time be spent on garbage collection.

Options for the CMS collector

Table 9-12 shows the options for the CMS collector.

Table 9-12 Concurrent mark and sweep options

Option	Default	Description
–XX:+CMSIncrementalMode	Disabled	Enables a mode in which the concurrent phases are done incrementally, periodically stopping the concurrent phase to yield back the processor to the application.
–XX:+CMSIncrementalPacing	Disabled	Enables automatic control of the amount of work the CMS collector is allowed to do before giving up the processor, based on application behavior.

Option	Default	Description
–XX:ParallelGCThreads=*n*	The number of CPUs	Number of garbage collector threads for the parallel young generation collections and for the parallel parts of the old generation collections.
-XX:CMSInitiatingOccupancyFraction=*n*	~68%	Sets the percentage of the current tenured generation size.
-XX:+UseCMSInitiatingOccupancyOnly	Disabled	CMS uses more than just the occupancy of the old generation to start a cycle. However, when setting this flag, it will use only the occupancy as its trigger.

9.4.6 Tools for performance tuning

This sections discusses tools for performance tuning.

Generate JVM GC output

You can enable verbose garbage collection for WebSphere Application Server by checking the box in the administrator console and restarting the server. The following property is set to true in the server.xml file:

```
verboseModeGarbageCollection="true"
```

By default on Solaris, the GC data will be written to the server's native_stdout.log file in the log directory.

Example 9-23 shows the output generated when `verboseModeGarbageCollection` is set to true.

Example 9-23 Portion of native_stdout.log showing GC data

```
[GC 50565K->34417K(101312K), 0.0241116 secs]
[GC 50417K->34424K(101120K), 0.0233317 secs]
[GC 50232K->34428K(100928K), 0.0220433 secs]
[GC 50108K->34456K(100736K), 0.0248518 secs]
[GC 50007K->34500K(100608K), 0.0243897 secs]
[GC 49924K->34515K(100416K), 0.0257724 secs]
[GC 49811K->34552K(100288K), 0.0221639 secs]
[GC 49720K->34526K(100096K), 0.0243514 secs]
[GC 49566K->34583K(99904K), 0.0225725 secs]
[GC 49495K->34586K(99776K), 0.0218356 secs]
```

In Example 9-23, if we look at the GC details at the line in bold:

`[GC 50108K->34456K(100736K), 0.0248518 secs]`

This line indicates the following:

- GC: Indicates that it was a minor collection (in young generation). A Full GC would indicate that it was a major collection (tenured generation).
- 50108K: Combined size of live objects before GC.

- 34456K: The combined size of live objects after GC.
- 100736K: The total available space, excluding the permanent generation, which is the total heap minus one of the survivor spaces.
- 0.0248518 secs: Time it took for garbage collection.

You can get more GC details by adding any of the following arguments to the server's Generic JVM arguments configuration:

- -XX:+PrintGC
- -XX:+PrintGCDetails
- -XX:+PrintGCTimeStamps

Refer to Table 9-9 on page 362 for a description of each of these options.

Example 9-24 shows output from when the -XX:+PrintGCDetails argument is used.

Example 9-24 More detailed GC data

```
[GC [PSYoungGen: 34081K->3533K(56832K)] 46314K->16756K(93696K), 0.0600419 secs]
[GC [PSYoungGen: 54669K->2139K(51840K)] 67892K->18351K(88704K), 0.0536013 secs]
[GC [PSYoungGen: 51803K->1359K(55552K)] 68015K->19439K(92416K), 0.0543981 secs]
[GC [PSYoungGen: 51919K->1385K(56448K)] 69999K->20508K(93312K), 0.0211575 secs]
[GC [PSYoungGen: 51945K->1424K(84032K)] 71068K->21592K(120896K), 0.0304477 secs]
[GC [PSYoungGen: 79504K->1569K(84096K)] 99672K->22905K(120960K), 0.0489358 secs]
[GC [PSYoungGen: 79649K->1744K(84096K)] 100985K->24565K(120960K), 0.0238232 secs]
[GC [PSYoungGen: 79888K->1010K(84160K)] 102709K->25503K(121024K), 0.0204290 secs]
[GC [PSYoungGen: 79346K->4083K(83776K)] 103839K->29306K(120640K), 0.0620245 secs]
[GC [PSYoungGen: 82035K->5154K(80512K)] 107258K->33265K(117376K), 0.0515145 secs]
```

In Example 9-24, look at the line marked in bold:

[GC [PSYoungGen: 79504K->1569K(84096K)] 99672K->22905K(120960K), 0.0489358 secs]

This line indicates the following:

- GC: A garbage collection event has occurred.
- PSYoungGen: Young generation GC stat.
- 79504K: Total size of live objects in young generation before GC.
- 1569K: Total size of live objects in young generation after GC.
- 84096K: Size of the young generation.
- 99672K: Combined size of live objects before GC.
- 22905K: Combined size of live objects after GC.
- 120960K: Total heap.

Example 9-25 shows an example of a full garbage collection.

Example 9-25 Full GC data

```
[Full GC [PSYoungGen: 38184K->0K(176182K)] [ParOldGen:
1427048K->439878K(1539996K)] 1473633K->449978K(3300736K) [PSPermGen:
4634K->4631K(16384K)], 5.3205801 secs]
```

A line similar to the one in Example 9-25 on page 365 gets logged into nativestd_out.log when the JVM of WebSphere Application Server runs Full GC. The output tells us the following:

- A Full GC ran for 5.3205801 seconds, which means all the WebSphere Application Server processing was paused for this duration, as Full GC has taken over everything.
- From the output, it is clear that the WebSphere Application Server is running with the PrintGCDetails flag and has logged information about:
  - Young Generation
    - Marked by PSYoungGen.
    - 38184K: Size of the live objects in young generation before Full GC.
    - 0K: Live objects in young generation after Full GC.
    - 176182K: Total size allocated to young generation.
  - Old Generation
    - Marked by ParOldGen.
    - 1427048K: Size of the live objects in old generation before Full GC.
    - 439878K: Size of the live objects in old generation after Full GC.
    - 1539996K: Total size allocated to young generation.
  - Heap Summary.
    - 1473633K: Size of live objects before Full GC.
    - 449978K: Size of live objects after Full GC.
    - 3300736K: Total size of heap.
  - Permanent Generation
    - Marked by PSPermGen.
    - 4634K: Size of the live object in perm gen before Full GC.
    - 4631K: Size of the live object in perm gen before Full GC.
    - 16384K: Total size of perm gen.

Avoiding Full GC results in better performance, as it severely affects response time. Full GC is usually an indication that the Java heap is too small or many short-lived objects have been created. Increase the heap size by using the MaximumHeap (-Xmx) and InitialHeap (-Xms) options until Full GCs are no longer triggered or it can be controlled to minimize the impact. For a long running high volume Web site, it may be impossible to eliminate the full GC completely; however, the time it takes to do the Full GC and the frequency of it can be controlled by looking at the GC log and applying the appropriate tuning. For example, if an application generates many short-lived objects that get collected by a minor GC, you can increase the young generation (-Xmn). The reason is that these objects do not get promoted and they get collected during minor GC.

If your application tends to generate the long-lived objects that are going to persist for some time, you can increase the old generation. It is ideal to pre-allocate the heap by setting -Xmx and -Xms to the same value (that is, the fixed heap size). For example, to set the Java heap to 3.5 GB, you can use the following Java options:

-Xmx3550m -Xms3550m -Xmn2g -Xss128k

The J2SE 1.5.0_06 release also introduced parallelism in garbage collection for old generation. Add the -XX:+UseParallelOldGC option to the standard JVM flags to enable this feature.

For young generation, the number of parallel GC threads is the number of *logical processors* presented by the Solaris OS. For example, on a fully populated UltraSPARC T1 processor-based system (that is, eight cores and four threads/core), the processors equate to the number of threads which, in this case, is 32. It may be necessary to scale back the number of threads involved in young generation GC to achieve response time constraints. To reduce the number of threads, you can set `XX:ParallelGCThreads=number_of_threads`. Similarly, the UltraSPARC T2 processor has 64 threads and for these systems the number Parallel GC threads would be 64, which may be too much and may affect performance adversely. Therefore, proper caution needs to be taken when working with these types of hardware. At the same time, US IV+, for example, may have few parallel GC threads and increasing this number would improve the performance of the system.

Generate thread dumps

Thread dumps or Java dumps can be generated by issuing the **`kill -3 <pid>`** command. Java dumps enable you to examine real-time thread activity within the server. For example, if you suspect that an application is having a hung thread problem, try performing the following steps:

1. Take three or four Java dumps a few minutes apart.
2. Examine the Java dump data for patterns or problem trends.
3. The thread dumps are a snapshot of a point in time, so one dump would not mean anything. When looking at multiple dumps, we see the pattern and then take corrective actions accordingly.

4. Use the ThreadAnalyzer tool, which can be downloaded using the IBM Support Assistant to help analyze the Java dump data, as shown in Figure 9-7.

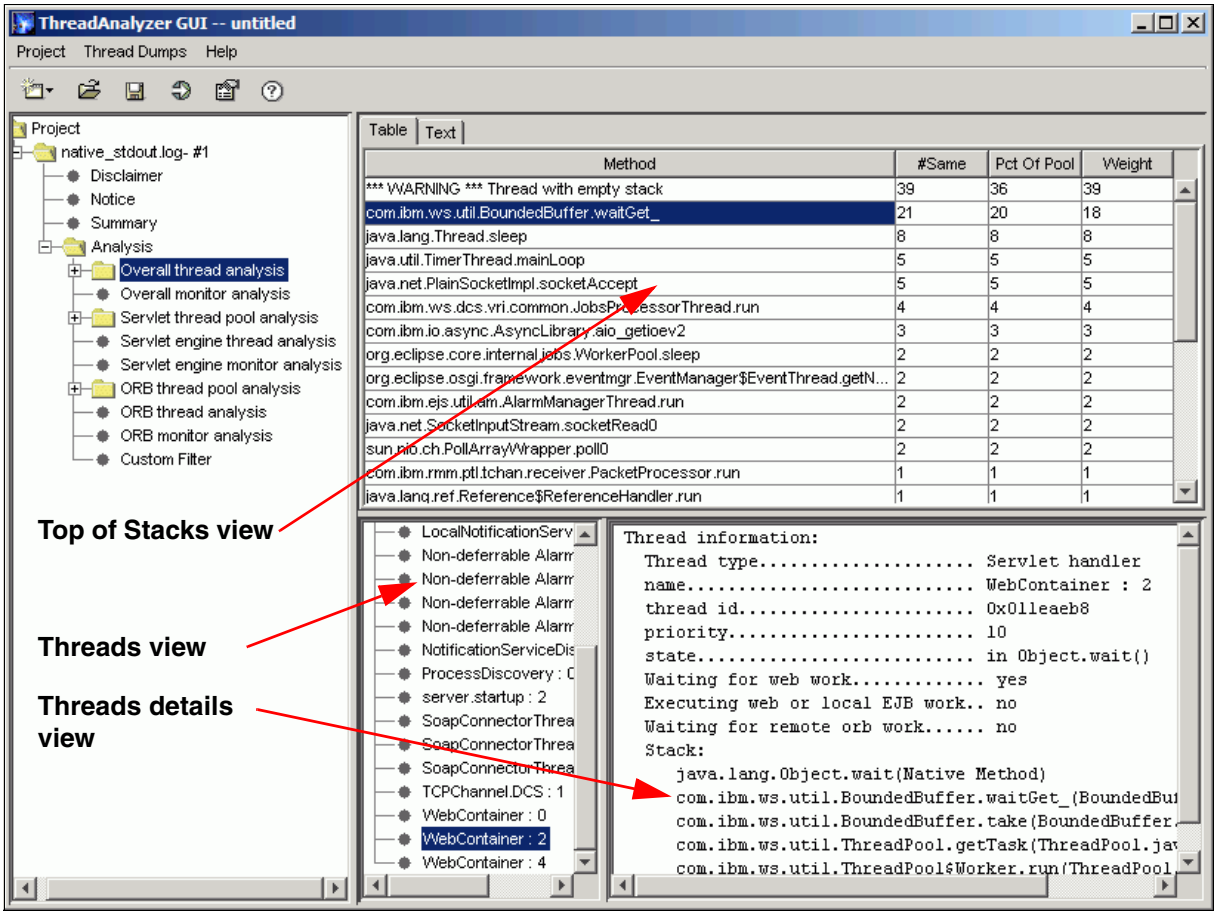


Figure 9-7 Thread Analyzer tool

There are three important views that we should note:

- Top of Stacks view: Displays the top of stacks for all threads currently executing in the JVM. This view indicates the class and method in which the threads are currently executing. When analyzing your thread dumps, you should look critically at methods where a majority of the threads in your heap are executing.
- Threads view: Displays the threads that were executing the method highlighted in the Top of Stacks view.
- Thread Detail view: Displays details of the threads that were selected in the Threads view.

Using the jmap command

The **jmap** command is included in the Solaris and Linux releases of the JDK. It is not included in the JDK5.0 release on Windows. The **jmap** command prints memory related statistics for a running VM or core file. If **jmap** is used with a process or core file without any command-line options, then it prints the list of shared objects loaded (the output is similar to the Solaris **pmap** utility). For more specific information, the -heap, -histo, or -permstat options can be used. These are described in the following sections.

Heap configuration and usage

The -heap option is used to obtain Java heap information, including:

1. GC algorithm specific information: This includes the name of the GC algorithm (Parallel GC for example) and algorithm specific details (such as number of threads for parallel GC).
2. Heap configuration: The heap configuration may have been specified as command-line options or selected by the VM based on the machine configuration.
3. Heap usage summary: For each generation, it prints the total capacity, in-use, and available free memory. If a generation is organized as a collection of spaces (the new generation, for example), then a space-wise memory size summary is included.

Example 9-26 shows an output from **`jmap -heap`**.

Example 9-26 Output from jmap -heap

```
edge-bash# jmap -heap 8134
Attaching to process ID 8134, please wait...
Debugger attached successfully.
Server compiler detected.
JVM version is 1.5.0_06-erdist-20060404

using thread-local object allocation.
Parallel GC with 32 thread(s)

Heap Configuration:
   MinHeapFreeRatio = 40
   MaxHeapFreeRatio = 70
   MaxHeapSize      = 268435456 (256.0MB)
   NewSize          = 2228224 (2.125MB)
   MaxNewSize       = 4294901760 (4095.9375MB)
   OldSize          = 1441792 (1.375MB)
   NewRatio         = 2
   SurvivorRatio    = 32
   PermSize         = 16777216 (16.0MB)
   MaxPermSize      = 268435456 (256.0MB)

Heap Usage:
PS Young Generation
Eden Space:
   capacity = 44761088 (42.6875MB)
   used     = 15615640 (14.892234802246094MB)
   free     = 29145448 (27.795265197753906MB)
   34.88664082517386% used
From Space:
   capacity = 6094848 (5.8125MB)
   used     = 5237680 (4.9950408935546875MB)
   free     = 857168 (0.8174591064453125MB)
   85.93618741599462% used
To Space:
   capacity = 6094848 (5.8125MB)
   used     = 0 (0.0MB)
   free     = 6094848 (5.8125MB)
   0.0% used
PS Old Generation
   capacity = 180355072 (172.0MB)
```

```
   used     = 27175992 (25.91704559326172MB)
   free     = 153179080 (146.08295440673828MB)
   15.068049763524256% used
PS Perm Generation
   capacity = 142606336 (136.0MB)
   used     = 78661728 (75.01766967773438MB)
   free     = 63944608 (60.982330322265625MB)
   55.16005123362822% used
```

Heap histogram

The -histo option can be used to obtain a class-wise histogram of the heap. For each class, it prints the number of objects, memory size in bytes, and fully qualified class name. Note that internal classes in the HotSpot VM are prefixed with an "*". The histogram is useful when trying to understand how the heap is used. To get the size of an object, you need to divide the total size by the count of that object type.

Example 9-27 shows an output from **jmap -histo**:

Example 9-27 Output from jmap -histo

```
edge-bash# jmap -histo 8134
Attaching to process ID 8134, please wait...
Debugger attached successfully.
Server compiler detected.
JVM version is 1.5.0_06-erdist-20060404
Iterating over heap. This may take a while...
Object Histogram:

Size    Count   Class description
-------------------------------------------------------
19871280        166678  * ConstMethodKlass
13686368        19034   byte[]
12479328        270864  * SymbolKlass
12005792        166678  * MethodKlass
8803944 14473   * ConstantPoolKlass
8744672 95124   char[]
6663312 14473   * InstanceKlassKlass
5954776 21062   int[]
5364912 12512   * ConstantPoolCacheKlass
2453736 102239  java.lang.String
1483336 18017   short[]
1382128 15706   java.lang.Class
1219216 20670   java.lang.Object[]
1155272 26173   java.lang.Object[]
756576  31524   java.util.HashMap$Entry
742344  2660    * MethodDataKlass
658112  6173    java.util.HashMap$Entry[]
463488  14484   java.util.TreeMap$Entry
366016  1204    * ObjArrayKlassKlass
350016  14584   java.util.Hashtable$Entry
307680  19230   com.ibm.ejs.util.Bucket
[...lines removed here to reduce output...]
Heap traversal took 49.395 seconds.
```

Getting information about the permanent generation

The permanent generation is the area of the heap that holds all the reflective data of the virtual machine itself, such as class and method objects (also called the “method area” in the Java Virtual Machine Specification).

Configuring the size of the permanent generation can be important for applications that dynamically generate and load a very large number of classes (Java Server Pages or Web containers, for example). If an application loads too many classes, then it is possible it will abort with an `OutOfMemoryError` error. The specific error is `Exception in thread XXXX java.lang.OutOfMemoryError: PermGen space`.

To get further information about permanent generation, use the -permstat option. It prints statistics for the objects in the permanent generation.

Example 9-28 shows the output from the **`jmap -permstat`** command.

Example 9-28 Output from jmap -permstat

```
edge-bash# jmap -permstat 8134
Attaching to process ID 8134, please wait...
Debugger attached successfully.
Server compiler detected.
JVM version is 1.5.0_06-erdist-20060404
finding class loader instances ..
done.
done.
computing per loader stat ..done.
please wait.. computing liveness.................................done
class_loader    classes bytes   parent_loader   alive?  type

<bootstrap>     3225    7380912   null          live    <internal>
0xf52719d8      1       1416    0xe7bed8f8      dead    sun/reflect/DelegatingClassLoader@0xd7842898
0xe8995b78      1       1408      null          dead    sun/reflect/DelegatingClassLoader@0xd7842898
0xe9161c70      3       8488      null          dead    com/sun/jmx/remote/util/OrderClassLoaders@0xdb9c1d70
0xf5271e00      3       8488      null          dead    com/sun/jmx/remote/util/OrderClassLoaders@0xdb9c1d70
0xe91f9ca8      3       8488      null          dead    com/sun/jmx/remote/util/OrderClassLoaders@0xdb9c1d70
0xe901b008      1       1416      null          dead    sun/reflect/DelegatingClassLoader@0xd7842898
0xe93895b8      3       8488      null          dead    com/sun/jmx/remote/util/OrderClassLoaders@0xdb9c1d70
0xe86d8b60      1       1408      null          dead    sun/reflect/DelegatingClassLoader@0xd7842898
0xe8d03398      1       1408    0xe7c5c9c0      dead    sun/reflect/DelegatingClassLoader@0xd7842898
0xe911bca8      3       8488      null          dead    com/sun/jmx/remote/util/OrderClassLoaders@0xdb9c1d70
[...lines removed here to reduce output...]
total = 629     18997   48392896          N/A         alive=53, dead=576          N/A
```

For each class loader object, the following details are printed:

1. The address of the class loader object in the snapshot when the utility was run.
2. The number of classes loaded (defined by this loader with the method java.lang.ClassLoader.defineClass).
3. The approximate number of bytes consumed by meta-data for all classes loaded by this classloader.
4. The address of the parent class loader (if any).
5. A “live” or “dead” indication, which indicates whether the loader object will be garbage collected in the future.
6. The class name of this class loader.

For more information, refer to the **`jmap`** manual page at:

`http://java.sun.com/j2se/1.5.0/docs/tooldocs/share/jmap.info`

The jstack command

The **jstack** command-line utility is included in the Solaris and Linux releases of the JDK. It is not included in the JDK 5.0 release on Windows. The utility attaches to the specified process (or core file) and prints the stack traces of all threads that are attached to the virtual machine (this includes Java threads and VM internal threads).

A stack trace of all threads can be useful when trying to diagnose a number of issues, such as deadlocks or hangs. In many cases, a thread dump can be obtained by pressing Ctrl-\ at the application console (standard input) or by sending the process a QUIT signal. Thread dumps can also be obtained programmatically using the Thread.getAllStackTraces method, or in the debugger using the debugger option to print all thread stacks (the **where** command in the case of the jdb sample debugger). In these examples, the VM process must be in a state where it can execute code. In rare cases (for example, if you encounter a bug in the thread library or HotSpot VM), this may not be possible, but it may be possible with the **jstack** utility, as it attaches to the process using an operating system interface.

Example 9-29 shows the output from the **jstack** command.

Example 9-29 Output from jstack command

```
edge-bash# jstack 8134
Attaching to process ID 8134, please wait...
Debugger attached successfully.
Server compiler detected.
JVM version is 1.5.0_06-erdist-20060404

Thread t@1353: (state = BLOCKED)
 - java.lang.Object.wait(long) @bci=0 (Compiled frame; information may be
imprecise)
 - com.ibm.ws.util.BoundedBuffer.waitGet_(long) @bci=22, line=192 (Compiled frame)
 - com.ibm.ws.util.BoundedBuffer.take() @bci=86, line=543 (Compiled frame)
 - com.ibm.ws.util.ThreadPool.getTask() @bci=137, line=819 (Compiled frame)
 - com.ibm.ws.util.ThreadPool$Worker.run() @bci=264, line=1544 (Interpreted frame)


Thread t@1352: (state = IN_NATIVE)
 - java.net.SocketInputStream.socketRead0(java.io.FileDescriptor, byte[], int,
int, int) @bci=0 (Compiled frame; information may be imprecise)
 - java.net.SocketInputStream.read(byte[], int, int) @bci=84, line=129 (Compiled
frame)
 - java.io.BufferedInputStream.fill() @bci=175, line=218 (Compiled frame)
 - java.io.BufferedInputStream.read() @bci=12, line=235 (Compiled frame)
 - java.io.FilterInputStream.read() @bci=4, line=66 (Compiled frame)
 - sun.rmi.transport.tcp.TCPTransport$ConnectionHandler.run() @bci=685, line=701
(Interpreted frame)
 - java.lang.Thread.run() @bci=11, line=595 (Interpreted frame)


Thread t@1350: (state = BLOCKED)
 - java.lang.Object.wait(long) @bci=0 (Compiled frame; information may be
imprecise)
 - com.ibm.ws.util.BoundedBuffer.waitGet_(long) @bci=22, line=192 (Compiled frame)
 - com.ibm.ws.util.ThreadPool.getTask() @bci=121, line=819 (Compiled frame)
 - com.ibm.ws.util.ThreadPool$Worker.run() @bci=264, line=1544 (Compiled frame)
```

```
Thread t@1341: (state = BLOCKED)
 - java.lang.Object.wait(long) @bci=0 (Compiled frame; information may be
imprecise)
 - com.ibm.ws.util.BoundedBuffer.waitGet_(long) @bci=22, line=192 (Compiled frame)
 - com.ibm.ws.util.BoundedBuffer.take() @bci=86, line=543 (Compiled frame)
 - com.ibm.ws.util.ThreadPool.getTask() @bci=137, line=819 (Compiled frame)
 - com.ibm.ws.util.ThreadPool$Worker.run() @bci=264, line=1544 (Interpreted frame)
[...lines removed here to reduce output...]
```

For more information, the manual page for **jstack** is available at:

http://java.sun.com/j2se/1.5.0/docs/tooldocs/share/jstack.html

9.4.7 Startup and footprint

When you upgrade to one version from another version of WebSphere Application Server, you may be upgrading the underlying Java technology versions as well. As technology keeps changing, applications demand more resources and are becoming more complex with new features. To address these changes from release to release, some of the default parameters, or more specifically in this context, the JVM options keep changing to address the demands of the application. The WebSphere Application Server administrator needs to pay close attention to the technology change so that they can address the demands of their enterprise and take care of the changes in WebSphere Application Server and Java. When WebSphere Application Server adds new features, more Java classes are needed to implement those features. The addition of these classes results in additional startup impact that ultimately increases startup time.

One of the interesting things to note here is the startup, which is defined by the time it takes to start the application, and footprint, which is defined by the memory needed by WebSphere Application Server, also changes with the above changes. The default parameters are usually defined to address the need for more generalized cases of deployment, which may include the developer environment to a default production deployment. So some of the settings may not be right *per se* for developer or some special deployment cases. When you observe such a change, you may consult the relevant documentation and take the appropriate action to minimize the impact of technology change and take advantage of the new technology.

For example, when using the WebSphere Application Server V6.1, you may notice the increase in memory usage by WebSphere Application Server, but this is one result of the JVM option that has been changed in Java SE 5.0. To take control of this memory increase, some of these options can be reverted to get memory usage back to the level it was with the previous version. Some recommendations on managing memory usage can be found at the following link:

http://www.ibm.com/support/docview.wss?rs=180&context=SSEQTP&q1=footprint&uid=swg21240768&loc=en_US&cs=utf-8&lang=en

In summary, the information at that link suggests overriding a default setting that is causing the JVM to use more memory by using the following options:

-client -XX:MaxPermSize=256m -XX:-UseLargePages -XX:+UseSerialGC

As you may notice, the JVM has been forced to run in -client mode, which used to be the default for WebSphere Application Server V6.0.2 or earlier. Due to ergonomics and the hardware configuration, the JVM normally chooses to run in -server mode.

Additionally, as we stated earlier, class data sharing (CDS) has been added to JVM and it is available with WebSphere Application Server V6.1 Fix Pack 7. The class data sharing mode makes your application startup faster.

To enable your WebSphere Application Server installation to make use of this feature, you need to have this JVM option added to you generic JVM argument:

-Xshare:on

Other related options are:

- -Xshare:off to turn off class data sharing
- -Xshare:auto to enable class data sharing when possible

Another important thing to note here is that this feature in JDK 1.5 has been made available to the client VM only, so you need to have the -client option as well genericJVMArguments, because ergonomics is going to start the server VM if the underlying hardware is classified as a server class machine (for example, greater than 2 CPUs and 2 GB of RAM).

To speed up the startup process, WebSphere Application Server has additional parameters that can be adjusted to help speed up the startup process:

- Parallel Start

  Select this field to start the server on multiple threads. This might shorten the startup time. This can be accessed by selecting **Applications** → **Enterprise Applications** → ***application_name*** → **Startup behavior**.

- Run in development mode

  Enabling this option may reduce the startup time of an application server. This may include JVM settings, such as disabling bytecode verification and reducing JIT compilation costs. Do not enable this setting on production servers.

  Specify that you want to use the JVM properties -Xverify and -Xquickstart on startup. Before selecting this option, add the -Xverify and -Xquickstart properties as generic arguments to the JVM configuration.

  If you select this option, you must save the configuration and restart the server before this configuration change takes effect.

  The default setting for this option is false, which indicates that the server will not be started in development mode. Setting this option to true specifies that the server will be started in development mode (with settings that will speed server startup time).

Other factors that affect the WebSphere Application Server startup are that major portions of the server startup cannot be done over multiple threads, such as classloading. The system will have a great deal of processing capability that the WebSphere Application Server process cannot take advantage of due to the current single threaded nature of the class loader. This can be verified by running **`mpstat 5`** during server startup; you will notice that only one core, thread, or CPU is being used while the rest of them are idle.

9.4.8 Additional performance tuning tools

The following tools are also useful for JVM tuning and troubleshooting. These will be discussed and used in Chapter 10, “Problem determination” on page 395.

The jvmstat tools

The **jvmstat** tools add light-weight performance and configuration instrumentation to the HotSpot JVM and provide a set of monitoring APIs and tools for monitoring the performance of the JVM in production and test environments. The instrumentation has been designed in such a way that it is “always on”, yet has a negligible performance impact.

Detailed information can be found at Sun’s Java Web site at:

http://java.sun.com/performance/jvmstat

Here we talk some of the important features of these tools and discuss in detail the part of **jvmstat** that is bundled with JVM and is already available with WebSphere Application Server installations. These tools are located in WAS_JAVA_DIR/bin. They are:

- **jps**
- **jstat**
- **jstatd**

The **jps** tool can be used to list all the instrumented JVM processes running on the system. Running **jps** on a system with WebSphere Application Server would generate output similar to that shown in Example 9-30.

Example 9-30 Sample jps command output

```
# ./jps
9465 WSPreLauncher
9485 Jps
```

The **jstat** tool, also known as the JVM Statistics Monitoring Tool, attaches to a running HotSpot Java virtual machine and collects and logs performance statistics specified by the command-line options. It was formerly known as **jvmstat**.

A more detailed description, usage, and samples can be found at:

http://java.sun.com/javase/6/docs/technotes/tools/share/jstat.html

This is very helpful and more readable compared to the PrintGCDetails option of verbose gc. Example 9-31 shows an output from the **jstat** command.

Example 9-31 Sample jstat output

```
# ./jstat -gcutil 9465 10000 5
  S0     S1     E      O      P     YGC     YGCT    FGC    FGCT     GCT
  0.00  15.29  71.80   0.00  71.02      1    0.123     0    0.000    0.123
  0.00  15.29  71.80   0.00  71.02      1    0.123     0    0.000    0.123
  0.00  15.29  71.80   0.00  71.02      1    0.123     0    0.000    0.123
  0.00  15.29  71.80   0.00  71.02      1    0.123     0    0.000    0.123
  0.00  15.29  72.05   0.00  71.02      1    0.123     0    0.000    0.123
```

In Example 9-31, the -gcutil option tells the **jstat** tool to collect and print a summary of Garbage Collection Statistics at an interval of 10000 ms and to do it five times.

A description of each column in the **`jstat -gcutil`** output is given in Table 9-13.

Table 9-13 jstat output column descriptions

Column	Description
S0	Survivor space 0 utilization is a percentage of the space's current capacity.
S1	Survivor space 1 utilization is a percentage of the space's current capacity.
E	Eden space utilization is a percentage of the space's current capacity.
O	Old space utilization is a percentage of the space's current capacity.
P	Permanent space utilization is a percentage of the space's current capacity.
YGC	Number of young generation GC events.
YGCT	Young generation garbage collection time.
FGC	Number of full GC events,
FGCT	Full garbage collection time.
GCT	Total garbage collection time.

There are various other options available as well, such as class, compiler, gccapacity, gcold, and so on.

The monitoring interfaces added to the HotSpot JVM are proprietary and may or may not be supported in future versions of the HotSpot JVM, but for the current version, they can be used to look at different stats to make the right tuning decisions.

Java heap analysis tool (HAT)

Information about the Java heap analysis tool (HAT) can be found at:

`http://hat.dev.java.net`

To profile WebSphere applications, use IBM Rational Application Developer V7; information about it can be found at:

`http://www.ibm.com/developerworks/rational/products/rad`

9.5 WebSphere Application Server Performance Management

WebSphere Application Server provides tunable settings for its major components to enable you to make adjustments to better match the runtime environment to the characteristics of your application. Many applications can run successfully without any changes to the default values for these tuning parameters. Other applications might need changes, for example, a larger heap size, to achieve optimal performance.

Performance tuning can yield significant gains in performance even if an application is not optimized for performance. However, correcting shortcomings of an application typically results in larger performance gains than are possible with just altering tuning parameters. Many factors contribute to a high performing application.

9.5.1 Adjusting the WebSphere Application Server system queues

WebSphere Application Server establishes a queuing network, which is a group of interconnected queues that represent various components. There are queues established for the network, Web server, Web container, EJB container, Object Request Broker (ORB), data source, and possibly a connection manager to a custom back-end system. Each of these resources represents a queue of requests waiting to use that resource. Queues are load-dependent resources. As such, the average response time of a request depends on the number of concurrent clients.

As an example, think of an application, consisting of servlets and EJBs, that accesses a back-end database. Each of these application elements reside in the appropriate WebSphere component (for example, servlets in the Web container) and each component can handle a certain number of requests in a given time frame.

A client request enters the Web server and travels through the WebSphere components in order to provide a response to the client. Figure 9-8 illustrates the processing path this application takes through the WebSphere components as interconnected pipes that form a large tube.

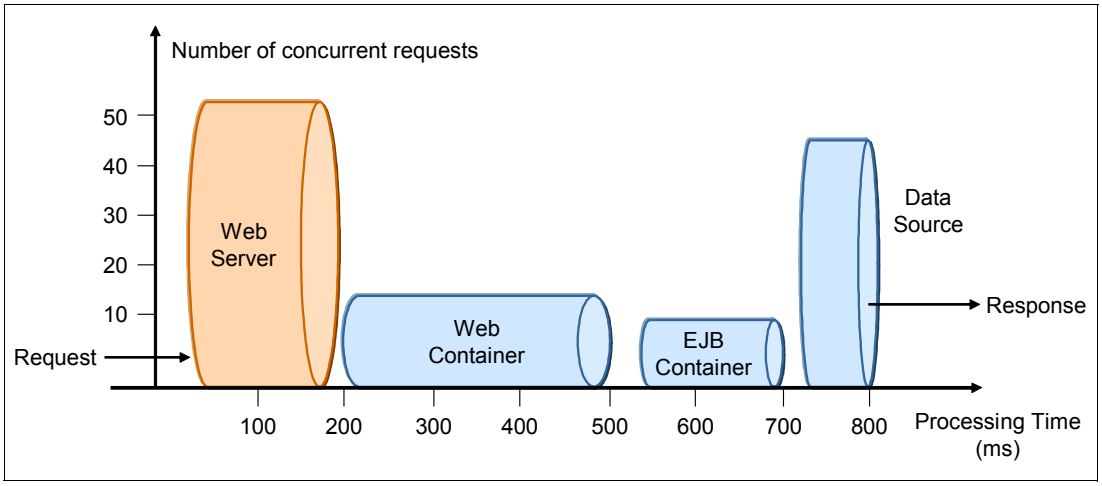


Figure 9-8 Queuing network

The width of the pipes (illustrated by height) represents the number of requests that can be processed at any given time. The length represents the processing time that it takes to provide a response to the request.

In order to find processing bottlenecks, it is useful to calculate a transactions per second (tps) ratio for each component. Here are some ratio calculations for a fictional application:

The Web server can process 50 requests in 100 ms = $\frac{50 req}{0.1 s} = 500 tps$

The Web container parts can process 18 requests in 300 ms = $\frac{18 req}{0.3 s} = 60 tps$

The EJB container parts can process nine requests in 150 ms = $\frac{9 req}{0.15 s} = 60 tps$

The datasource can process 40 requests in 50 ms = $\frac{40 req}{0.05 s} = 800 tps$

These sample ratio calculations show that the application elements in the Web container and in the EJB container process requests at the same speed. Nothing would be gained from increasing the processing speed of the servlets or Web container, because the EJB container would still only handle 60 transactions per second. The requests normally queued at the Web container would simply shift to the queue for the EJB container.

This example illustrates the importance of queues in the system. Looking at the operations ratio of the Web and EJB containers, each is able to process the same number of requests over time. However, the Web container could produce twice the number of requests that the EJB container could process at any given time. In order to keep the EJB container fully utilized, the other half of the requests must be queued.

It should be noted that it is common for applications to have more requests processed in the Web server and Web container than by EJBs and back-end systems. As a result, the queue sizes would be progressively smaller moving deeper into the WebSphere components. This is one of the reasons queue sizes should not solely depend on operation ratios.

9.5.2 WebSphere Application Server performance tuning

This section lists some sample tuning parameters for various WebSphere components. This is a generic starting guideline based on the size, default values, and tuning options. You will need to test your environment with proper workloads and adjust these parameters accordingly. As the workload behavior changes, you will need to have additional adjustments over time.

Thread pool sizes

- A fixed size pool gives better performance.
- Lower sizes are better.
- Example tuning parameters:
  - WebContainer Min/Max: 30/30
  - Default Min/Max: 10/10
  - ORB Min/Max: 22/22

Default thread pool

- Observe the spikes.
- The default value may be adequate, as it does not require many resources.

EJB container

- Bean Cache: 30530
- Timeout: 3000

Web container HTTP transport

- Keep Alive
- Buffer Size
- Example tuning parameters:
  - **`set keepAliveEnabled true`**
  - **`set maxPersistentRequests -1`**
  - **`set maxPersistentTimeout 3000`**
  - **`set maxReadTimeout 6000`**

- **set maxWriteTimeout 6000**

JDBC

- Type 4 driver (pure Java).
- Keep it low.
  - Connections Min/Max: 100/100
- Set statement cache size properly.
  - Statement Cache: 60

Persistence manager (CMP)

- -Dcom.ibm.ws.pm.batch=true -Dcom.ibm.ws.pm.deferredcreate=true
- -Dcom.ibm.CORBA.FragmentSize=3000

Service integration bus

- Default settings are usually adequate.

Disable unnecessary features for production

- Performance Monitoring Infrastructure (PMI)
- Debug logs
- GC details/verbose

WebSphere Application Server performance issues

A WebSphere Application Server deployment is typically a multi-tier environment with many interdependencies among resources and components.

Performance issues can be caused by any of the following:

- User applications
- Underlying JVM
- Underlying operating systems
- Infrastructure: Network, firewall, storage, and so on
- Other dependent tiers: Web server, data base servers, ERP, Web services, and messaging servers

Performance tuning check list

The following check list contains recommendations that have improved performance, scalability, or both for many applications.

WebSphere Application Server provides several tunable parameters and options to match the application server environment to the requirements of your application:

- Review the hardware and software prerequisites for WebSphere Application Server.
- Install the most current refresh pack, Fix Pack, and recommended interim fixes.
- Check the hardware configuration and settings.
- Review your application design.
- Tune the Solaris Operating System.
- Tune the Java Virtual Machine settings.
- Use a Type-4 (or pure Java) JDBC driver.

- Tune WebSphere Application Server JDBC resources and associated connection pools.
- Enable the Pass by Reference option, if applicable.
- Ensure that the transaction log is on a fast disk.
- Tune related components, such as data bases, messaging providers, and so on.
- Disable functions that are not needed.

Review hardware and software prerequisites for WebSphere Application Server

Review the hardware and software requirements at the following Web site:

http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg27006921

Install the most current refresh pack, Fix Pack, and recommended interim fixes

The list of recommended updates is maintained on the Support site at:

http://www-1.ibm.com/support/docview.wss?uid=swg27004980

Check hardware configuration and settings

Sometimes transient errors can cause Ethernet adapters to shift down to a lower speed. Verify that the system has adequate memory and that the number and position of memory dual inline memory modules (DIMMs) are optimal. With some systems, there are some memory DIMM configurations that permit higher performance than other DIMM configurations. Verify that the hardware that is used is the hardware that is supposed to be used.

Review your application design

You can track many performance problems back to the application design. Review the design to determine if it causes performance problems.

Tune the Solaris Operating System

Operating system configuration plays a key role in performance. In many cases, adjustments to some TCP/IP parameters might be necessary for your application.

Tune the Java Virtual Machine settings

Many applications need a larger heap size for best performance.

Use a Type-4 (or pure Java) JDBC driver

In general, the type 2 JDBC driver is recommended. However, the type 4 JDBC drivers perform faster in the area of multi-row fetch. Use the link above to view a list of database vendor-specific requirements, which can tell you if a Type-4 JDBC driver is supported for your database.

Tune WebSphere Application Server JDBC resources and associated connection pools

The JDBC data source configuration might have a significant performance impact. For example, the connection pool size and prepared statement cache need to be sized based on the number of concurrent requests being processed and the design of the application.

Enable the Pass by Reference option if applicable

Use applications that can take advantage of the Pass by Reference option to avoid the cost of copying parameters to the stack.

Ensure that the transaction log is on a fast disk

Some applications generate a high rate of writes to the WebSphere Application Server transaction log. Locating the transaction log on a fast disk or disk array can improve response time.

Tune related components, such as data bases, messaging providers, and so forth

In many cases, some other component, for example, a database, needs adjustment to achieve higher throughput for your entire configuration.

Disable functions that are not needed

For example, if your application does not use the Web services addressing (WS-Addressing) support, disabling this function can improve performance.

9.5.3 WebSphere component monitoring

For monitoring WebSphere components, use Tivoli Performance Viewer to help identify application server performance issues.

From the TPV shown in Figure 9-9, you can view current activity or log Performance Monitoring Infrastructure (PMI) performance data for the following:

- System resources, such as CPU utilization
- WebSphere pools and queues, such as a database connection pool
- Customer application data, such as average servlet response time

By viewing PMI data, administrators can determine which part of the application and configuration settings to alter in order to improve performance. For example, in order to determine what part of the application to focus on, you can view the servlet summary report, as shown in Figure 9-10 on page 382, enterprise beans, and Enterprise JavaBeans (EJB) methods, and determine which of these resources has the highest response time. You can then focus on improving the configuration for those application resources with the longest response times.

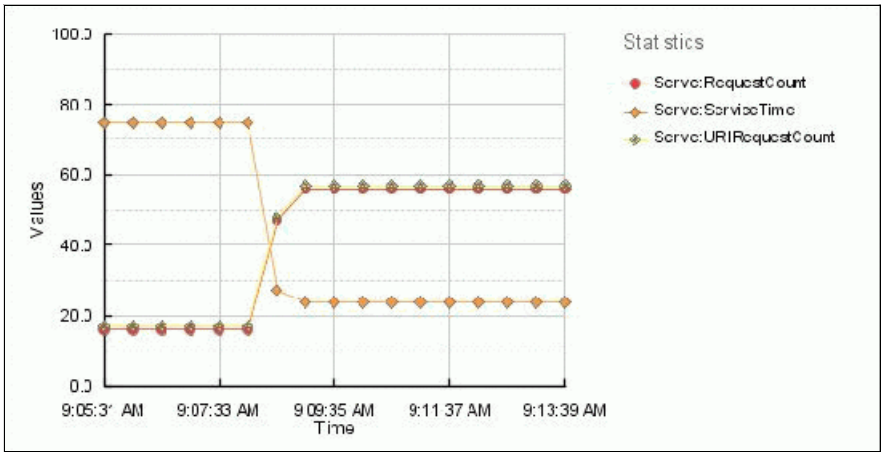


Figure 9-9 Tivoli Performance Viewer chart view of servlet statistics

Time	Servlets RequestCount	Servlets ServiceTime	Servlets URIRequestCount	Servlets URIServiceTime
9:21:16 AM	56.00	237.94	57.00	380.82
9:20:46 AM	56.00	237.94	57.00	380.82

Figure 9-10 Tivoli Performance Viewer chart table of servlet statistics

For details on using the Tivoli Performance Viewer, read the WebSphere Application Server V6.1 Information Center article, *Monitoring performance with the Tivoli Performance Viewer (TPV)*, found at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tprf_tpvmonitor.html

9.5.4 Tivoli Performance Advisor

Use the Tivoli Performance Advisor in addition to Tivoli Performance Viewer. Together, these tools will help you to tune your system, and the Tivoli Performance Advisor will give you recommendations on inefficient settings.

View recommendations and data from the Performance Advisor by clicking **Advisor** in the Tivoli Performance Viewer.

9.5.5 Performance monitoring guidelines

Before tuning measures can be taken, there is one important question that has to be answered first: “What part of the system shall I tune?”. Simple random tuning acts will seldom provide the desired effect, and sometimes will even produce worse results. This section discusses which components to examine first, and gives a practical hotlist of the top ten monitors and which tools to use. Finally, it makes a short excursion into performance analysis and load testing best practices.

Top ten monitoring hotlist

If you experience performance problems, start using Tivoli Performance Viewer and make your way through the following top ten monitoring items checklist. It will help you more easily find those significant parts of your system where the most performance impact can be gained by tuning. It cannot be stated often enough that performance tuning is not a science, but an art, so do not expect miracles by adjusting a few “knobs” inside WebSphere Application Server: Every system and every application behaves differently, and requires different tuning measures! But the Performance Monitoring Infrastructure (PMI), Tivoli Performance Viewer, and the Performance Advisors will assist you with achieving your goal: a system configured for optimum performance.

Because of the sheer magnitude of monitors and tuning parameters, knowing where to start, what to monitor, and which component to tune first is difficult. Follow these top ten monitoring steps to check the most important counters and metrics of WebSphere Application Server. See Figure 9-11 on page 383 for a graphical overview of the most important resources to check. Consider the following:

- If you recognize something out of the ordinary, for example, an overutilized thread pool or a JVM that spends 50% in garbage collection at peak load time, then concentrate your tuning actions there first.
- Perform your examination when the system is under typical production level load and make note of the observed values.

- Alternatively, save Tivoli Performance Viewer sessions to the file system, where the monitoring data will be stored for recurring analysis.
- Keep in mind that one set of tuning parameters for one application will not work the same way for another.

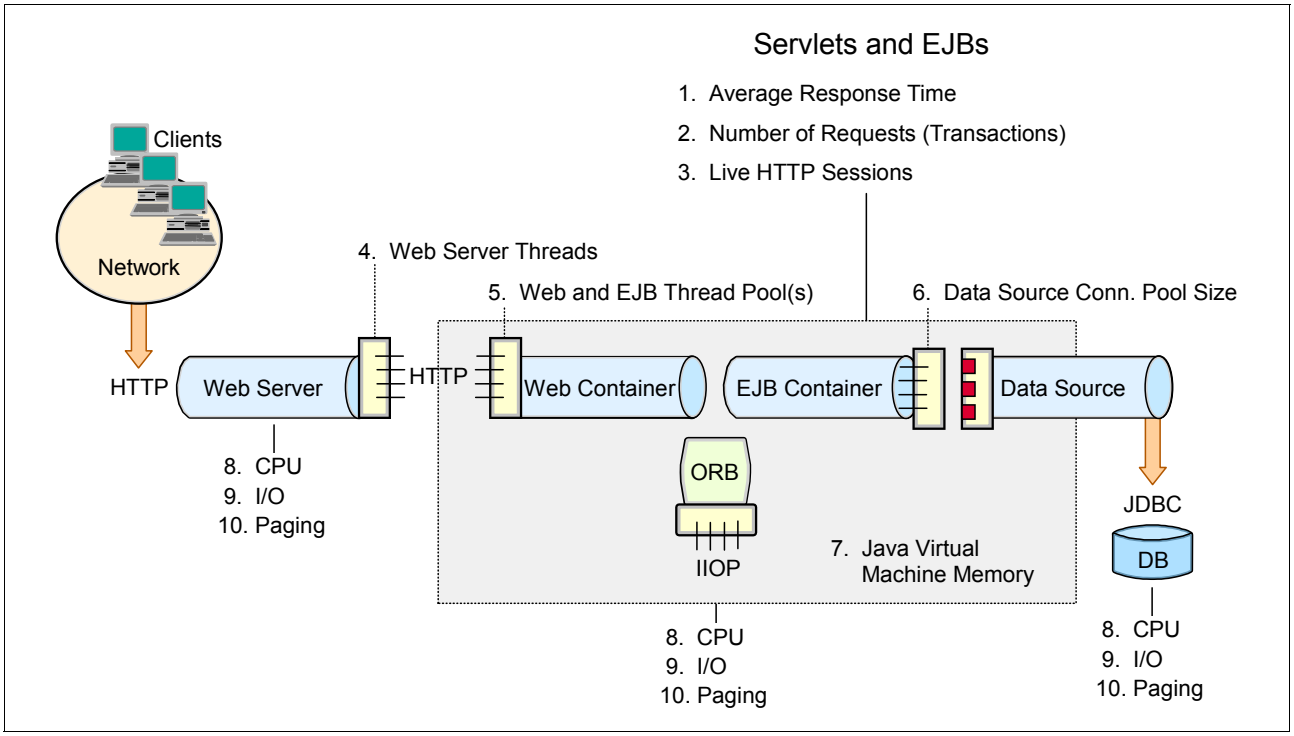


Figure 9-11 Top ten monitoring items checklist

Servlets and Enterprise JavaBeans

Here are some items to consider for servlet and Enterprise JavaBeans monitoring:

1. Average response time

   To find this value in the Tivoli Performance Viewer, select **Performance Modules** → **Web Applications** and look at the value named ServiceTime.

2. Number of requests (Transactions)

   To find this value in the Tivoli Performance Viewer, select **Performance Modules** → **Web Applications** and look at the value named RequestCount.

3. Live number of HTTP Sessions

   To find this value in the Tivoli Performance Viewer, select **Performance Modules** → **Servlet Session Manager** and look at the value named Live Count.

Thread pools

Here are some items to consider for thread pool monitoring:

1. Web server threads

   For information about how to monitor the IBM HTTP Server, refer to Chapter 14, “Server-side performance and analysis tools”, in *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392.

For information about how to monitor the Sun Java System Web server, refer to the *Sun Java System Web Server 7.0 Performance Tuning, Sizing, and Scaling Guide*, found at:

http://dlc.sun.com/pdf/819-2635/819-2635.pdf

For all other Web servers, refer to the documentation for your Web server product.

2. Web container and EJB container thread pool

   All WebSphere Application Server thread pools can be viewed in the Thread Pools summary report in the Tivoli Performance Viewer. The thread pool summary shows the usage of all thread pools in the application server over time.

   Tip: When the application is experiencing normal to heavy usage, the pools used by that application should be nearly fully utilized. Low utilization means that resources are being wasted by maintaining connections or threads that are never used. Consider the order in which work progresses through the various pools. If the resources near the end of the pipeline are underutilized, it might mean that resources near the front are constrained or that more resources than necessary are allocated near the end of the pipeline.

Data sources

All datasource connection pools can be viewed in the Connection Pools summary report in the Tivoli Performance Viewer. The connection pool summary lists all data source connections that are defined in the application server and shows their usage over time.

Tip: When the application is experiencing normal to heavy usage, the pools used by that application should be nearly fully utilized. Low utilization means that resources are being wasted by maintaining connections or threads that are never used. Consider the order in which work progresses through the various pools. If the resources near the end of the pipeline are underutilized, it might mean that resources near the front are constrained or that more resources than necessary are allocated near the end of the pipeline.

Java Virtual Machine

Here are some items to consider for Java Virtual Machine monitoring:

Java Virtual Machine (JVM) memory and garbage collection statistics

Information provided about the JVM runtime depends on the debug settings of the JVM and can be viewed in the Tivoli Performance Viewer by selecting **Performance Modules** → **JVM Runtime**. Use the Java virtual machine Tool interface (JVMTI) for profiling the JVM. JVMTI is new in JVM V1.5. For WebSphere Application Server V6.0.2 and earlier, use the Java virtual machine Profiling interface (JVMPI).

For details on how to enable JVMTI data reporting, read the Information Center article, *Enabling the Java virtual machine profiler data*, found at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/tprf_jvmpidata.html

Runtime Performance Advisor

The Runtime Performance Advisor service runs inside each application server process and has to be enabled explicitly. It uses the same rules engine as Tivoli Performance Advisor and will produce similar tuning recommendations. In contrast to Performance Advisor in Tivoli Performance Viewer, the Runtime Performance Advisor does not provide the full spectrum of advice types, but is suited for long-term analysis.

Tip: It is important to configure the correct number of processors in the Runtime Performance Advisor configuration window, otherwise the recommendations can be inaccurate.

However, keep in mind that both Tivoli Performance Advisor and Runtime Performance Advisor advice can only be accurate if your application runs with a production level load, and without exceptions and errors. This is best done by running a performance test, where a production level load can be simulated in a reproducible manner. Both advisors need the CPU utilization to be high to provide good (and complete) advice, which is best accomplished during a load and stress test. If this is not the case, the resulting output may be misleading and most certainly contradictory.

Performance analysis

Web performance analysis describes the process of finding out how well your system (a Web application or generally a Web site) performs, and pinpointing performance problems caused by inadequate resource utilization, such as memory leaks or over- or underutilized object pools, to name just a few. Once you know the trouble spots of your system, you can take counter-measures to reduce their impact by taking appropriate tuning actions.

Terminology

Performance analysis is a very comprehensive topic. It fills entire books and is surely out of the scope of this book. We attempt to provide a short introduction to that subject. What follows is a concise definition of the three most important concepts used in performance analysis literature:

- Load
- Throughput
- Response Time

Load

A Web site, and especially the application that is running behind it, typically behaves and performs differently depending on the current load, that is, the number of users that are concurrently *using* the Web site at a given point in time. This includes clients who actively perform requests at a time, but also clients who are currently reading a previously requested Web page. Peak load often refers to the maximum number of concurrent users using the site at some point in time.

Response time

Response time refers to the time interval from the time the client initiates a request until it receives the response. Typically, the time taken to display the response (usually the HTML data inside the browser window) is also accounted for in the response time.

Throughput

A Web site can only handle a specific number of concurrent requests. Throughput depends on that number of requests and on the average time a request takes to process; it is measured in requests per second. If the site can handle 100 concurrent requests and the average response time is one second, the Web site's throughput is 100 requests per second.

Performance testing

To discover the performance of your system in the first place, you have to run a performance test to get an idea of how many users will be able to access the site at the same time without noteworthy delays. Even at a later time, load tests are most helpful to discover how much performance was gained by a specific tuning measure. After each tuning step, the load test has to be repeated (10-15 iterations of it are not uncommon), until the system meets your performance objectives. For performance assessment and tuning, the following test requirements can be stated:

- The load expected on the system has to be defined.

  This implies that you have to create a model of your expected real-world, production level workload, based on your experience and knowledge of your clients' behavior. Using this representative model in combination with a stress test tool, you will create as many test cases as needed, and combine them into a test suite that simulates the desired user behavior and performs the necessary interactions with your site during a load test. In some cases, when no good estimate of such a model can be given, even a crude approximation is better than performing no load test at all, and it can always be refined for future test cycles.

- The load test has to be repeatable.

  Tuning without the ability to perform repeatable tests is very hard, because of the lack of comparable performance data. To be repeatable, it is necessary to restore the initial system state after each test iteration. Usually, persistent application or transaction data is stored in databases or back-end systems, so that implies that a second, staging database or back-end system has to be used for testing purposes. Changes to the data on that system will not have consequences in the real world, meaning that the placement of a thousand orders in an online shop during load testing will not activate the order fulfillment process in the back end.

- Find the optimum load level.

  The goal is to find the site's saturation point. First, the system has to be driven over the point where performance begins to degrade. This is commonly referred to as *drawing the throughput curve* (see Figure 9-12 on page 387).

  Start a series of tests to plot the throughput curve. Begin testing with wide open server queues (pools) to allow maximum concurrency in your system. Record the throughput (average requests per second) and the response time (average time for requests), increasing the concurrent user load after each test. You will find the system's saturation point where the throughput becomes constant (the maximum throughput point is reached), but the response time begins to grow proportionally with the user load.

  Refer to 9.5.1, "Adjusting the WebSphere Application Server system queues" on page 377 for more information about the queuing network.

  Important: The goal here is to drive CPU utilization to nearly 100 percent. If you cannot reach that state with opened up queues, there is most likely a bottleneck in your application, in your network connection, some hard limit on a resource, or possibly any other external component you did not modify.

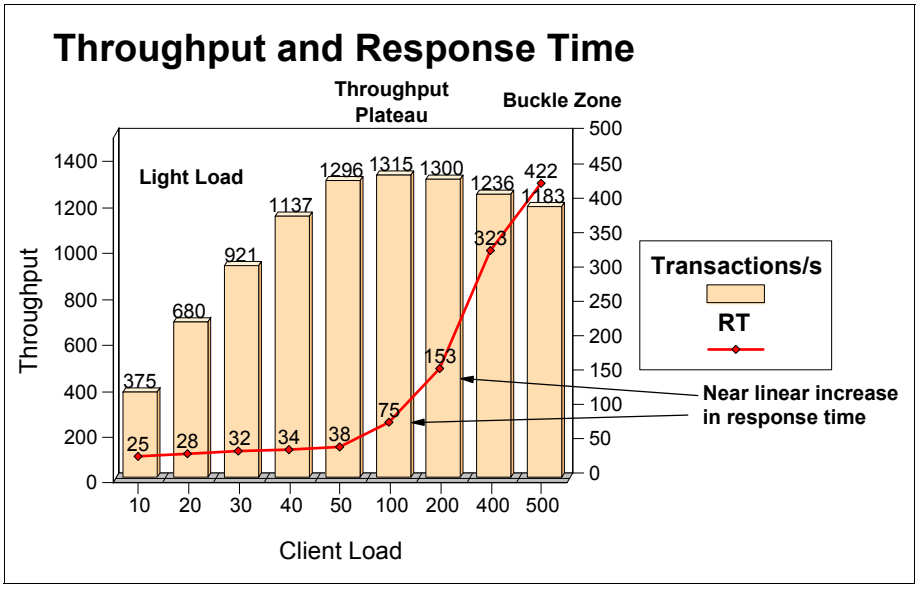


Figure 9-12 How to find the optimum load level for a system: the saturation point

The monitoring, tuning, and testing cycle

Perform an initial load test, to get a *baseline* that you can refer to later, with the Performance Advisors facility enabled. Use the number of users at your saturation point as the load parameter in all future load tests. Check the monitors in Tivoli Performance Viewer according to the top ten hotlist and the Performance Advisors. After you have implemented an advisor's recommendation, perform another test cycle to discover if you have gained throughput (because of a decreased average response time) or freed resources (for example, memory by reducing thread pools), which you could then possibly spend on other components that need more memory resources. Repeat this procedure to find an optimum configuration for your system and application. Keep in mind that performance tuning is an iterative process, so do not rely on results from a single run.

Here is a short overview of the steps necessary to perform a load test for tuning purposes with the Runtime Performance Advisor:

1. Enable the PMI service in all application servers and the Node Agent (if using a WebSphere Network Deployment environment), and restart both. In WebSphere Application Server V6.1, PMI is enabled by default with the Basic monitoring set, so unless you disabled it previously, there should be no need to restart the servers. However, you might want to configure the statistics to be monitored for your test.
2. Enable Runtime Performance Advisor for all application servers.
3. Select **Monitoring and Tuning** → **Performance Viewer** → **Current Activity** and select the server you want to monitor in the WebSphere Administrative Console.
4. Simulate your representative production level load using a stress test tool.
   - Make sure that there are no errors or exceptions during the load test.
   - Record throughput and average response time statistics to plot a curve at the end of all testing iterations.

5. Check the Runtime Performance Advisor and Tivoli Performance Viewer. Apply advice and follow your own intuition. You will get messages that provide suggestions about which settings might improve the performance on your system:
   - Restart components or services if necessary.
   - Reset all components (for example, the database) to the initial state.
6. Retest (go back to step 4 on page 387).

Attention: Advice provided by Tivoli Performance Viewer Performance Advisor or the Runtime Performance Advisor is not applied automatically by WebSphere. You will have to decide which changes make sense for your environment, apply them to your configuration, and test the effects these changes had during a load test.

Data capture best practices

To get accurate results, be mindful of the following best practices for data capturing:

- Measure during steady-state of the load test.

  Do not include ramp-up/ramp-down times in your performance data measurements and analysis (see Figure 9-13). Measure during the steady-state when the maximum number of users are concurrently performing requests.

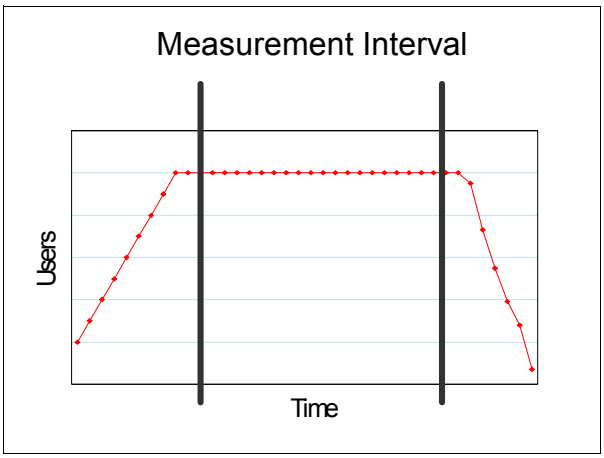


Figure 9-13 Measurement interval: concurrently active users versus elapsed time

- Monitor machine logs and resources.

  Monitor important log files for exceptions or errors. Be sure that there are no exceptions or deadlocks in the database. Keep an eye on system resources like memory, paging activity, CPU, disk IO, network utilization, socket states, and so on, for bottlenecks.

  Important log files are SystemOut.log and SystemErr.log. Monitor these logs to make sure your application runs without errors. SystemErr.log should typically remain free of entries. Errors logged there *must* be solved before you can capture meaningful performance data. Likewise, any sort of exception in SystemOut.log during the performance run should be solved before another run, because exception handling and the I/O necessary to write stacks to the log are expensive operations that impact performance.

For a Network Deployment cluster, you do not need to repeat all the monitoring steps for each cluster member if they are all set up identically; monitoring one or two representative cluster members should be sufficient. What is essential, however, is to check the CPU statistics for each node to make sure that all cluster member processes are using similar amounts of CPU and there are no extra processes consuming CPU on any nodes that can interfere with the application server CPU efficiency.

- Test on idle systems.

  Make sure you do not perform test runs during database backups, maintenance cycles, or while other people perform tests on these systems.

- Use isolated networks.

  Load driving machines should be attached to the same network switch as your first-layer machines, such as Web servers or network dispatchers, to be able to apply the highest load possible on the front network interfaces of your infrastructure.

- Performance tuning is an iterative process.

  10-15 test runs are quite usual during the tuning phase. Perform long lasting runs to detect resource leaks, for example, memory leaks, where the load tested application runs out of heap space only after a given time.

9.6 64-bit considerations

More customers now require that their applications as well a WebSphere Application Server be able to exceed the 32-bit limitations of 4 GB of addressable physical memory. Due to this increased customer demand, IBM has made WebSphere Application Server available with 32-bit and 64-bit JVMs to address specific customer needs. For Solaris 10, WebSphere Application Server V6.0.2 became available with 64-bit JVM starting with x64 systems.

Following that release, as of this writing, WebSphere Application Server in 64-bit format is available on all supported Solaris 10 platforms. The platform support for the WebSphere Application Server can be found at the following IBM Web site:

http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg27006921

IBM has tested WebSphere Application Server 64-bit performance and has published a report with the results. Although the report does not cover Solaris 10 itself, the results should apply in principle to all platforms. You can read the document, *IBM WebSphere Application Server and 64-bit platforms*, found at:

ftp://ftp.software.ibm.com/software/webserver/appserv/was/64bitPerf.pdf

Applications that require a large amount of memory can now leverage 64-bit WebSphere Application Server and achieve virtually unlimited heap space. However, proper JVM tuning is still required to address the challenges that are discussed in the previously referenced document. The proper sizing of the heap's old and new generations can be a greater challenge on 64-bit WebSphere Application Server. Since WebSphere Application Server on Solaris 10 uses Sun's JVM, the information found at the following Web site may be useful for examining the performance comparison of 32-bit and 64-bit in the context of performance benchmarks:

http://blogs.sun.com/dagastine/entry/no_tuning_required_java_se

As you will observe from the information provided in the previously referenced link, a minimal tuning of Sun's Java SE can get you started and may provide some performance advantages without getting into more complex tuning procedures.

9.7 Performance benchmarks

When characterizing the performance of WebSphere Application Server on any platform, it is always better if you can use your own application and do so. But in reality it is just not possible, as we may have to make the choice of system during the planning phase only. Cases like this get complicated when the application changes from the initial plan. So to help characterize the systems in context of WebSphere Application Server, there are some publicly available benchmark results that can be used to do some guesswork to arrive at the system configuration you will need.

Benchmark results are meant to provide the customer with guidelines for system performance characteristics and comparisons. Some of the benchmarks available for WebSphere Application Server are as follows:

- SPECjAppServer2004
- Trade (v2, 3 or 6)
- Apache DayTrader

However, you have to remember that your own application performance may vary depending on the quality of code and tuning applied. These applications, even when written to simulate some real-world scenario, have purposes that are different from ours, so the data from these benchmarks should be used with caution.

9.7.1 SPECjAppServer 2004 benchmark

The definition of this benchmark is found at http://www.spec.org/jAppServer2004. The Standard Performance Evaluation Corporation (SPEC) is a non-profit organization that provides a multi-tier, industry-standard benchmark for measuring the performance of Java EE technology based application servers. An end-to-end application exercises all major J2EE technologies implemented by compliant application servers.

Description of SPECjAppServer2004 benchmark

- The workload emulates an automobile manufacturing company and its associated dealerships.
- The dealers interact with the system using Web browsers (simulated by the driver) while the actual manufacturing process is accomplished through RMI (also driven by the driver).

Final metric – jAppServer Operations/sec (JOPS)

JOPS = DealerTransactions/sec + WorkOrders/sec

Performance characteristics

- The benchmark scales linearly with multiple application server instances (locally or horizontally).
- It heavily exercises all parts of the underlying infrastructure that make up the application environment: hardware, JVM software, database software, JDBC drivers, and the system network.
- A single instance on a single node matters the most. It shows the infrastructure's capability to scale under a highly stressed/contended environment.

9.7.2 IBM Trade Performance Benchmark Sample for WebSphere Application Server

The IBM Trade Performance Benchmark Sample for WebSphere Application Server (known as Trade 6) provides a suite of IBM developed workloads for characterizing the performance of the WebSphere Application Server. The workloads consist of an end-to-end Web application and a full set of primitives. The applications are a collection of Java classes, Java servlets, Java Server Pages, Web services, and Enterprise Java Beans built for open J2EE APIs. Together these provide versatile and portable test cases designed to measure aspects of scalability and performance. The IBM Trade Performance Benchmark Sample for WebSphere Application Server (Trade 6) can be downloaded from the following Web site:

http://www-306.ibm.com/software/webservers/appserv/was/performance.html

9.7.3 Apache DayTrader benchmark

Built on the model of an online stock trading system, DayTrader is the open source version of the IBM Trade Performance Benchmark Sample that was donated to the Apache Geronimo community in 2005. For a complete description of DayTrader, refer to the article found at:

http://cwiki.apache.org/GMOxDOC20/daytrader.html

9.8 Capacity planning

Wondering about how to plan and design an infrastructure deployment that is based on WebSphere middleware? We discuss some of the considerations for capacity planning so that the planned deployment environment will be able to address the need for the current demand as well as the future growth and needs of your enterprise. This section describes the WebSphere-specific components that you must understand in order to run a successful WebSphere infrastructure project.

Once the application design and scalability techniques have been decided, you need to determine the number of machines along with their capacity requirements for the project. It is given that the application design will evolve over time and capacity planning is usually done in the early stages of design. However, when planning for capacity, it is important that you have a static version of the application design with which to work. The better view you have of the application design, the better your sizing estimate will be. This would include all the components that will support the application, such as:

- Operating system
- Infrastructure component sizing
- Choice of hardware systems
- WebSphere Application Server sizing or capacity planning

You should also consider which hardware platforms you want to use. This decision is primarily dependent on your platform preference, which platforms have sizing information available, and which platforms WebSphere Application Server supports. Hardware decisions might also be driven by the availability of hardware that forces a limitation on the operating systems that can be deployed.

Next, determine whether you want to scale up or out. Scaling up means to do vertical scaling on a small number of machines with multiple processors. This can present significant single points of failure. Scaling out, however, means using a larger number of smaller machines.

This can generally be thought of as significantly more resilient, because it is unlikely that the failure of one small server is adequate to create a complete application outage. However, you would have to support and maintain many small machines. This is usually a decision of preference and cost for your environment. However, the reality is that application resiliency issues can change your view.

What you need to understand, though, is that the sizing estimates are solely based on your input, which means the better the input, the more accurate the estimation can be. Sizing work assumes an average standard of application performance behavior and an average response time is assumed for each transaction. Calculations based on these criteria are performed to determine the estimated number of machines and processors your application will require. If your enterprise has a user experience team, they might have documented standards for a typical response time that your new project will be required to meet.

If you need a more accurate estimation of your hardware requirements and you already have your application, consider benchmarking your own application on the actual hardware that you are considering. This will provide you with the best and accurate estimation.

Based on your estimation, you might have to update your production implementation design and the designs for the integration and development environments accordingly. Changes to the production environment should be incorporated into the development and testing environments if at all possible.

Benchmarking is the process used to take an application environment and determine the capacity of that environment through load testing. This determination enables you to make reasonable judgements as your environment begins to change. Using benchmarks, you can determine the current work environment capacity and set expectations as new applications and components are introduced.

Benchmarking is primarily of interest to two kinds of clients:

- Clients who already have an application and want to migrate to a new version of WebSphere or who want to evaluate the exact number of machines for their target deployment platform.
- Clients who sell products that are based on WebSphere and who want to provide sizing estimations for their products.

Many sophisticated enterprises maintain a benchmark of their application stack and change it after each launch or upgrade of a component. These customers usually have well-developed application testing environments and teams dedicated to the cause. For those that do not, an alternative is to do some mathematical analysis of the different available benchmarks and to try to map the complexity of the application slated for deployment to the one for which the actual benchmark data is available.

There are also third-party benchmark companies that provide such services. When choosing a service, make sure that the team that will perform your benchmark tests has adequate knowledge of the environment and a clearly defined set of goals. This helps reduce the costs of the benchmark tests and creates results that are much easier to quantify. If these techniques are applied without sufficient knowledge, you may end up with an under performing production system, thereby causing user dissatisfaction. For example, if Servlet or Java Server Page (JSP) processing in a customer application may be three times more processor intensive than the benchmark application, you will have to pay attention to these facts. Similarly, if your processing involves complex third-party transactions, your mileage will vary based on that as well. Response time, defined as time taken to serve a client request, is also a very relative term and has to be used in the right context. An acceptable response time of one second in one application may be totally unacceptable in other application contexts. However, this will provide a starting point for sizing the system.

Capacity planning solely depends on the choice of technology used, details of application complexity, and the methodology used for capacity planning. Wrong estimates about any one of them could lead to a system that would not meet your customer's expectations. All of these considerations have to be accurate and applied carefully to get the best estimation. Customers also need to forecast appropriately any growth in usage the application may experience in the near future. If this growth is not addressed, the application may meet the initial criteria, but when the load on application increases, it will fail to meet expectations. The capacity planning team needs to have an understanding of peak and average loads on the system. If the peak usage scenarios are not addressed, the system may have unexpected behaviors at those moments. The capacity to support peak loads can be achieved by using the different clustering and load balancing techniques available with WebSphere software.

Choice of hardware and its scalability also impacts capacity planning. For example, a system can be deployed with a minimal configuration of the system but have additional capability for expansion (that is, adding more memory or CPU) in the future to accommodate the additional workload requirements. Some systems may address the current needs, but if there is an increase in the load profile, the system cannot be expanded. In these situations, the architecture has to planned in such a way that systems can be added in parallel and removed without affecting the current deployment. This can be done by using one of the various technologies available with WebSphere:

- WebSphere Clustering
- Load Balancing through the Edge Component
- Use of Solaris virtualization, as discussed in Chapter 5, "Configuration of WebSphere Application Server in an advanced Solaris 10 Environment" on page 115
- Hardware clustering using Sun Clusters
- Extended Deployment technology within WebSphere Software

Systems have different characteristics that also need to be considered when choosing the right one for your deployment. A target deployment system may be suitable for a scenario that might not be appropriate in another application deployment due to its nature, technology, and its capability to expand. Some of the technology like LDoms are not available with all types of Sun systems, while some of the technologies like Solaris Container are available on all the systems running Solaris 10. Even when these characteristics are considered together, the system may not be perfect, but you should at least have a better performing and more stable application environment.

10

Problem determination

In general, this chapter discusses problem determination for WebSphere Application Server. However, there are several considerations that are specific to Solaris 10 that need to be addressed when troubleshooting, such as the Solaris JDK. Typically, these considerations are most important when troubleshooting JVM related problems. Therefore, this chapter discusses common WebSphere problems that are often platform neutral, as well as problems and tools that are specific to Solaris 10.

In this chapter, the following topics are discussed:

- Problem determination methodology
- Problem determination tools
- Common WebSphere Application Server problems
- Solaris JVM problem determination

10.1 Problem determination methodology

In this section, a basic methodology for performing problem determination will be presented. First, the causes and symptoms of common problems will be discussed, then a series of steps for problem determination and resolution will be presented. These steps include the following:

- Preparing for problems before they occur
- Organizing an investigation to identify the problem
- Providing relief options for users
- Conducting an initial and advanced investigation of problems

10.1.1 Causes of problems

When a problem occurs, your first inclination might be to call IBM Support so that they can provide a fix that resolves the problem. In some cases, contacting IBM is necessary. However, the experience of the WebSphere Application Server Support team has shown that a small percentage of client-reported problems are actually due to defects with WebSphere Application Server code. Most problems are caused by configuration issues, environment issues, application code defects, or a misunderstanding of WebSphere Application Server. Many of these problems can be resolved easily without calling IBM Support to open a problem management record (PMR). In addition, many issues can be resolved by following the problem determination procedures that are discussed in this book.

Each problem that you encounter has a different level of complexity and a different level of impact on your business. These factors determine how closely you follow the procedures in this book. For less complex problems, it might only be necessary to follow the most basic procedures. More complex problems can involve multiple components and maybe even multiple software products and systems. These problems require more time and effort with more thorough problem determination techniques. Obviously, the impact that the problem has on your business influences the urgency to resolve the problem and that also determines which problem determination techniques you follow.

10.1.2 Symptoms of common problems

Here are the common types of symptoms that you might see. Almost every symptom falls into one of these categories:

- Failure in installing or migrating WebSphere Application Server
- Failure in installing an application into WebSphere Application Server
- Difficulties in WebSphere Application Server system management or configuration
- Failure of an application or WebSphere Application Server process (for example, an application server, node agent, or deployment manager) to start
- Application problems
  - Application does not respond to incoming requests.
  - Application produces unexpected results (possibly errors or exceptions).
  - Application cannot connect to an external system or resource.
  - Application performs slowly or its performance degrades over time (JVM tuning problems).

10.1.3 General steps for problem determination

Figure 10-1 shows a flowchart of basic problem determination steps. A good problem determination methodology should begin with ensuring that you have system documentation and a diagnostic data collection plan to be used in the event that any problems occur.

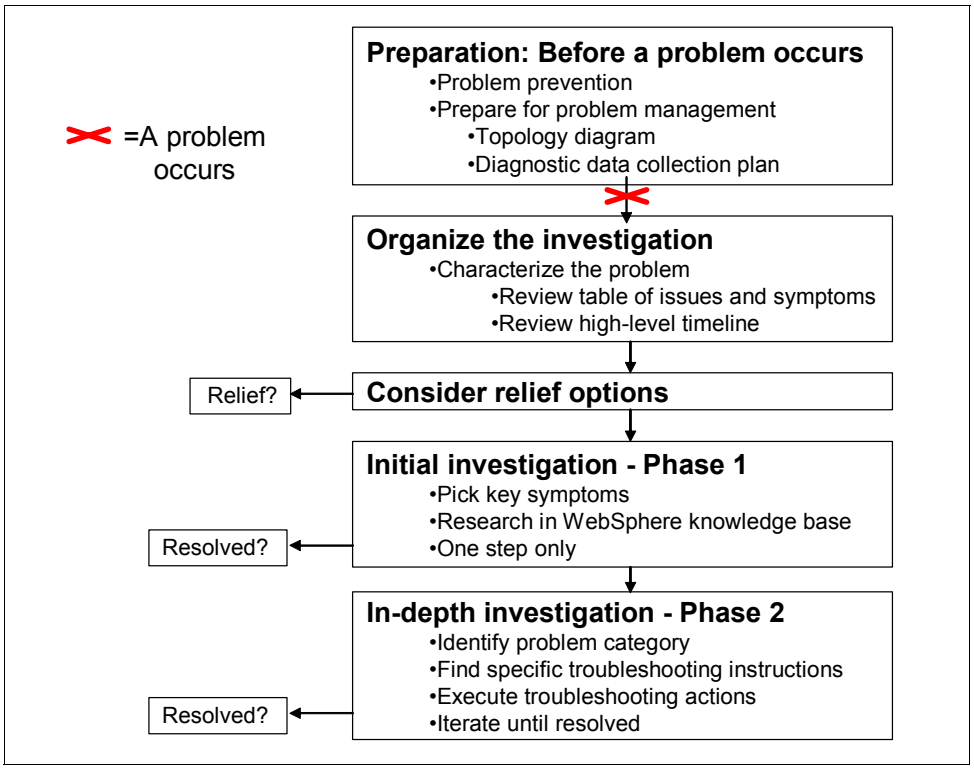


Figure 10-1 Steps for problem determination

After a problem occurs, two things need to be done, sometimes simultaneously: An investigation of the problem has to be organized, and you must find a way to provide relief to your users by reverting to safe conditions as soon as possible.

A large percentage of problems are common and well documented so that the identification of the problem and its resolution are relatively straight forward. Some problems are more complex and require a more in-depth analysis and, possibly, the application of several fixes, before the problem is completely resolved. In the following sections, each step in this methodology will be described in detail.

10.1.4 How to be prepared before problems occur

In any enterprise computing system, you can expect that some problems, large or small, will occur at times. In a best case scenario, the problems that you encounter will not be severe and will not result in critical business impact. However, it is a best practice to prepare and plan for the worst.

Applying WebSphere Application Server maintenance

Although many problems are caused by factors other than WebSphere Application Server code defects, the WebSphere Application Server Support team does find product defects when working through PMRs with clients. When a defect is found, the support team opens an authorized program analysis report (APAR). Each APAR has a unique identifier, a string that

contains two letters (either PQ or PK) and five numbers. You can search for a particular APAR or a problem symptom reported in an APAR on the WebSphere Application Server. The site is available at:

http://www-306.ibm.com/software/webservers/appserv/was/support

Fix packs for Version 6.1 are delivered on a regular basis, about once every three months. They only contain fixes for APARs, and they do not introduce any new features or functionality to the product. You can think of a Fix Pack as preventive maintenance. Each Fix Pack includes a list of defects, which lists every APAR that is fixed in that Fix Pack. It contains all of the APAR fixes that were included in the previous Fix Pack as well as fixes for APARs that have been opened since the last Fix Pack. The fix packs add a fourth number to your WebSphere Application Server version. For example, if you were to install Fix Pack 13 for WebSphere Application Server 6.1, you would be upgrading to WebSphere Application Server 6.1.0.13.

Fix packs do not contain upgrades to the Java Software Development Kit (SDK). They are tested with the latest Java SDK service release, but the upgrades to the Java SDK are delivered as separate fixes. You can also download the SDK fixes from the WebSphere Application Server Support site.

Because fix packs do not introduce new functionality to the product and because they have been fully regression tested, we recommend installing new fix packs as soon as they become available. However, we also recommend some level of testing with your applications.

Proactively installing fix packs as soon as they become available is an effective way to prevent problems from occurring. When you install a Fix Pack, you can be assured that you will not encounter any of the WebSphere Application Server code defects that are fixed in the Fix Pack. This saves the time and frustration of seeing one of these problems occur on your system.

For more information about the WebSphere Application Server V6.1 Update Strategy, you can review the Update Strategy document, which is available on the Support site:

http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg27009276

You can learn when fix packs are scheduled to be released by reviewing the Recommended Updates for WebSphere Application Server Base and Network at:

http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg27004980#ver61

To determine which fix packs have been installed on your system, you can usually just check the version of the product. This is available at the Welcome page of the administrative console and at the top of all SystemOut.log files for each WebSphere Application Server process. You can also use the **historyInfo** and **versionInfo** commands in the <was_home>/bin directory.

Checking the prerequisites

Another strategy for preventing problems is to ensure that all software and hardware in your environment meets the prerequisites of WebSphere Application Server V6.1. WebSphere Application Server has been tested with specific software and hardware configurations. It is known to work successfully in these configurations and to integrate well with the products with which it has been tested. You can find the software and hardware with which WebSphere Application Server has been tested and that it supports at:

http://www-306.ibm.com/software/webservers/appserv/doc/latest/prereq.html

If some of the software or hardware in your environment is not listed or if you are using older versions than the versions listed, you are at a greater risk for problems to occur. If you need to open a PMR, the support team will recommend moving to a supported configuration before proceeding with any other problem determination steps. Therefore, proactively ensuring that all of the software and hardware in your environment meets WebSphere Application Server's prerequisites is an important problem prevention and preparation technique.

System documentation

In the event that a problem occurs in your environment, it is possible that you will need to enlist the help of other people, either internal or external to your organization, to determine the root cause of the problem. When that happens, you will want everyone involved to understand thoroughly the details of the systems that are involved in your environment.

To this end, it is important to document the details of your configuration. Document all of the changes that have been made to your environment. In addition to that, you should maintain a high-level description of your basic topology. This is known as *system documentation*. System documentation is useful in the following circumstances:

- A problem occurs and you need to get assistance from others who might not be as familiar with your application and topology as you are. The system documentation allows you to bring them up to speed as quickly as possible.
- A problem occurs and you want to identify from which parts of your environment you should collect diagnostic data or monitor. Your system documentation shows the software components that are involved and the flow of your application, that is, how different software components are used when your application processes a request.

Your system documentation should consist of written documents and diagrams. Which information is included in the written documents and which is included in diagrams is a matter of preference. Overall, the information should be detailed and should show the specific versions and maintenance levels of the operating system and all software products involved, the hardware and network configuration, and specific host names and IP addresses of the systems that are involved.

A common and important component of system documentation is the topology diagram. It gives a quick overview of your system topology and application flow. Figure 10-2 shows an example.

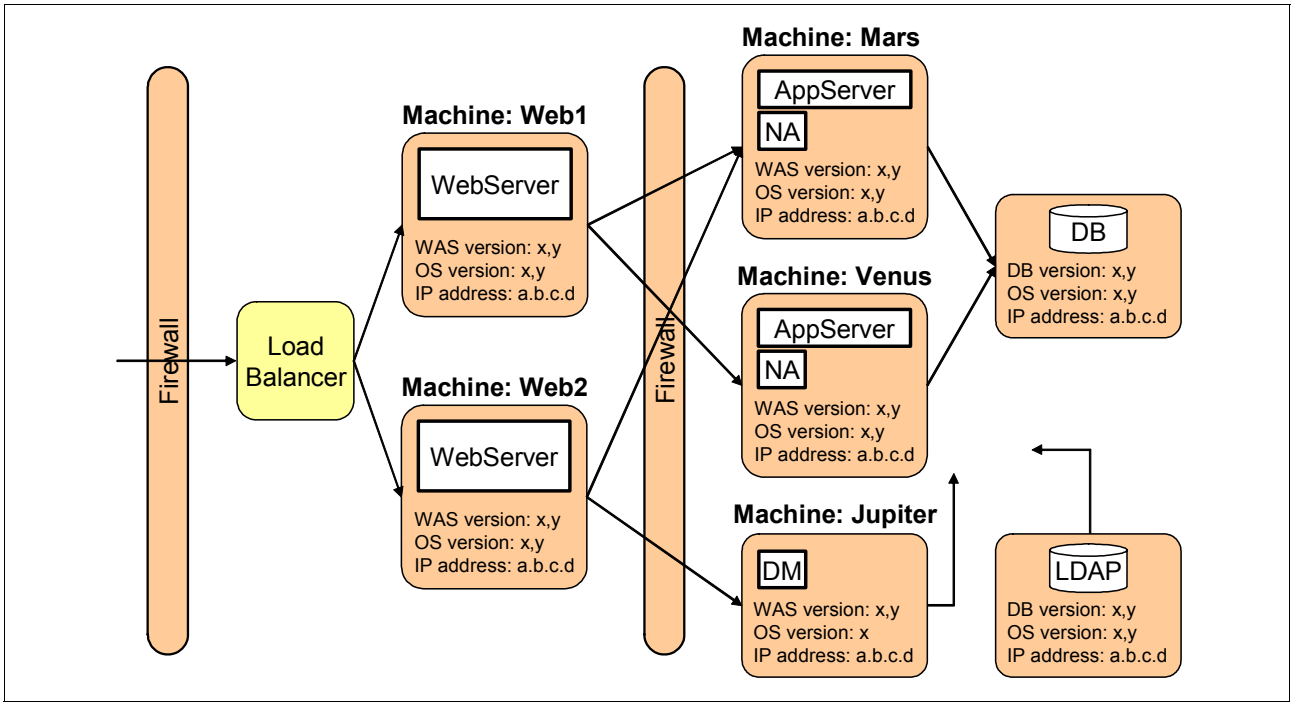


Figure 10-2 Example topology diagram

With a quick look at the diagram, you can see that HTTP requests are being load balanced between two Web servers and that applications are receiving HTTP requests from each of those Web servers. The applications are sending data to a DB2 database. An LDAP server is being used for the WebSphere security user repository.

In this example, specific software and hardware levels are indicated on each machine, but this information could also be included in accompanying written documentation in order to keep the diagram more readable if many more machines are involved.

Detailed system documentation is an integral part of your problem planning strategy that should not be overlooked.

Diagnostic data collection

Finally, to prepare for a problem occurrence, you should plan what diagnostic data to collect for various problem scenarios. In 10.1.2, “Symptoms of common problems” on page 396, several broad categories of problem symptoms were discussed. In subsequent sections, this document will elaborate on how to determine the cause of a problem for each of the problem symptoms. Information will also be included about what data to collect for different types of problems. It is a good idea to identify the most common problems that have occurred in the past in your environment and those that you believe might occur most often in the future. Then, you can form a diagnostic data collection plan so that you are prepared to collect the necessary data if a problem does occur.

There are some recommendations that apply to your diagnostic data collection plan regardless of the types of problems for which you are preparing. For example:

- Ensure that the clocks on all systems in your environment are synchronized. This helps in the analysis of diagnostic logs and traces. Often, a request is sent from one system to another, and it helps to match up the time stamps from both systems when analyzing the diagnostic data. If some systems are located in different time zones, make sure that this is documented in your system documentation (as discussed in “System documentation” on page 399).
- Configure WebSphere Application Server logging and tracing so that it captures a sufficient amount of data when a problem occurs. The WebSphere Application Server Support team has found that many times that the diagnostic data needed to determine the root cause of a problem has been overwritten before a client realizes that the problem has occurred. This type of situation can be prevented by increasing the amount of log and trace data that is saved before the files are rolled over. For both logging and tracing, you can configure file rotation properties for this purpose.

  You can find the instructions for configuring logging and tracing, including the file rotation properties, in WebSphere Application Server V6: Diagnostic Data, found at:

  http://www.redbooks.ibm.com/redpapers/pdfs/redp4085.pdf
- Plan to have extra disk space available on your system to store diagnostic data when problems occur. Many of the most severe problems, such as application server crashes, hangs, and out of memory conditions, require the largest amount of diagnostic data. If enough disk space is not available when a problem such as this occurs, you might need to reproduce the problem several times to collect the necessary data. To avoid this situation, we recommend that you have between 2 GB and 5 GB of extra disk space available on each system.
- After you resolve a problem, either delete or archive the diagnostic data that you collected for the problem. This will prevent the old diagnostic data from being confused with the new diagnostic data the next time that a problem occurs.
- Configure the thread monitor for hang detection. WebSphere Application Server V6.1 includes a thread monitor feature. This is also included in Version 6.0 and 5.1.1. The thread monitor is notified when the Web container, ORB, or asynchronous bean thread pools give work to a thread. By default, the thread monitor checks the status of all active threads every three minutes. If it finds a thread that has been active for more than ten minutes, it outputs a warning to the SystemOut log, similar to the one shown in Example 10-1.

Example 10-1 Thread monitor log message

```
WSVR0605W: Thread <threadname> has been active for <hangtime> and may be hung.
There are <totalthreads> threads in total in the server that may be hung.
```

- The thread monitor makes it easier to determine that a problem has occurred. If you see a WSVR0605W warning in your SystemOut log, you know that a thread has stopped responding. You can then perform further diagnostic steps to determine the cause of the hung thread. The thread monitor does not take any action to fix the problem beyond notifying you of the problem.

In preparing for diagnostic data collection, you might want to change the default thread monitor behavior. You can change the interval that the thread monitor checks the status of threads (this is *three minutes* by default) and the amount of time that a thread can be active before it is reported by the thread monitor (this is *ten minutes* by default). To alter these properties:

a. Log onto the administrative console.

b. Select your application server.

c. Under Server Infrastructure, select **Administration**.

d. Select **Custom Properties** and then click **New**.

e. Create these properties and provide the desired values for them:

   i. com.ibm.websphere.threadmonitor.interval (the interval that the thread monitor checks the status of threads)

   ii. com.ibm.websphere.threadmonitor.threshold (the amount of time that a thread can be active before it is reported by the thread monitor)

- Consider enabling verbose garbage collection on each application server. The performance impact of enabling verbose garbage collection is minimal, and the data is often useful when performance problems occur. To enable verbose garbage collection:

  a. Log onto the administrative console.

  b. Select your application server.

  c. Under Server Infrastructure, expand **Java and Process Management**, and then select **Process Definition**.

  d. In the resulting window, select **Java Virtual Machine** under Additional Properties.

  e. Select **Verbose garbage collection**.

  When verbose garbage collection is enabled, the output appears (on Solaris) in the native_stdout.log file for your application server.

- Use the WebSphere Application Server collector tool. The collector tool is run as an executable in the <was_home>/bin directory. It produces a Java archive (jar) file that contains all of the logs and XML configuration files from your system.

- Use the collection feature of the IBM Support Assistant. For details, see 10.2.1, “IBM Support Assistant portable collector tool” on page 408.

  The WebSphere Application Server Support team has compiled a comprehensive list of *MustGather* documents for different types of WebSphere Application Server problems. For more information, see *IBM - MustGather: Read first for all WebSphere Application Server products*, found at:

  http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg21145599

  The files listed in the MustGather documents are useful in determining the cause of your problem. You can use these documents to as part of your diagnostic data collection plan.

10.1.5 How to organize the investigation after a problem occurs

The following sections will describe how to organize an investigation.

Characterize the problem and identify problem symptoms

When you are first notified about the problem, you might only receive a vague, non-specific set of problem symptoms. You might be told that users cannot access your application at all or possibly that a specific action taken by your users is resulting in an error message.

You should collect specific details about the symptoms in order to form a detailed and thorough problem description. You can ask the following questions in order to get as specific a problem description as possible:

- What were the specific problem symptoms that were observed? Did an error occur? Did the application produce an unexpected result? Did the application fail to respond to an incoming request?
- What was the context under which the problem occurred? Did the user execute a specific function? Did the problem happen only after there was an unusually high workload on the system? Did it occur immediately after the application was restarted?
- How do you know when this particular problem occurs? Is there a something specific to watch for in order to recognize if the problem occurs again?
- How would you know that the problem was resolved? Would it be that an error message no longer occurred? Would the application behave differently? What specifically would confirm a problem resolution?
- Where did the problem occur? Did the problem occur only in your test environment, only in your production environment, or on both? Did it only occur on one system in your environment? Did it occur on multiple systems? Did it occur on every cluster member or only one?
- When did the problem occur? What was the time stamp of the error or unexpected behavior? Did the problem occur only once or many times? How often did the problem occur? Did it occur at certain intervals, or did it seem to occur at random times? Was there some event that occurred that might have triggered the problem? For example, did the user attempt a specific application function?
- Why might the problem have occurred? You might not be able to answer this right away. Was this the first time something specific was tried? Was there a recent change to the application code or the configuration of your environment? Does it happen in all of your environments or only one? If it only happens in one environment, how is this environment different from the others?
- Has the diagnostic data that is identified in your diagnostic data collection plan been collected? Does the data provide any other details about the problem or offer immediate clues as to why the problem occurred?

  Compile your answers in a document that gives as much specific detail about the problem occurrence as possible. We refer to this document as the *problem log*. Ensure that everyone who is involved in the problem determination process has access to the problem log and can update it with new information as it is uncovered. Also, ensure that the problem log lists the location of the diagnostic data.

  It is possible that the answers themselves will reveal the cause of the problem. For example, the original problem symptom could have been that the user received an error message when accessing a specific servlet in the application. You might have found that a new version of this servlet was just installed in the environment where the problem occurs but was not installed in other environments that are still functioning as expected. This could lead you to determine that an application code change caused the problem.

  On the other hand, the answer might require more investigation. If this is the case, you have developed a solid and complete problem description to use as the basis for more problem determination efforts.

10.1.6 Relief options and considerations

When any problem occurs, your first action should be to consider the business impact of the problem. Depending on the business impact, it might be necessary to take steps to limit the business impact before beginning your problem determination efforts. If the problem occurred in your test environment, the business impact is relatively low when compared to the potential impact of a production outage. In that case, making an effort to work around the problem while looking for a permanent solution is probably not necessary. You can use the test environment to execute all of the necessary problem determination steps.

However, if the problem occurred in production, you will probably want to consider how to quickly alleviate the problem symptoms so that customers and users will experience the least possible negative effects. In this section, we refer to this process as *reverting to safe conditions*. You will want to do this in parallel with your problem determination efforts.

Revert to safe conditions

Techniques for reverting to safe conditions are discussed here:

- Replace production with test environment.

  Make your test environment or another similar environment where the problem is not occurring in your temporary production environment. Configure your systems so that incoming requests are processed by an environment where the problem does not occur.

- Roll back to an older version of the application code.

  Install an older version of application code where the problem does not occur. If the problem started to occur after an application code change was introduced, it might be a good idea to go back to an earlier working version of the application.

- Revert to the baseline configuration.

  Change any recent configuration changes back to your baseline configuration. The baseline configuration should be a configuration that is fully tested and is known to be stable.

- Uninstall recent WebSphere Application Server updates.

  Remove any WebSphere Application Server maintenance that was recently installed before the problem occurred. This would temporarily resolve the problem if it is caused by a WebSphere Application Server code defect introduced by the latest maintenance package.

- Disable the problematic function of the application.

  Make the application function that produced the problem inaccessible to customers and users. You can post a notification on your Web site that the function is temporarily unavailable or under maintenance. You can provide an estimated time for when it will be available again.

Establishing safe conditions to which you can revert when a problem occurs and then reverting to those conditions when a production problem occurs is a good way to reduce the business impact of your problem. After you have evaluated whether reverting to safe conditions is necessary and taken the appropriate steps, you can begin the problem determination process.

10.1.7 Initial investigation: Phase 1 problem determination techniques

A large percentage of problems can be classified as *Phase 1* problems. These are well-known problems for which there is a fix readily available. All that is required is that the problem be correctly characterized, essential diagnostic data be collected, find the remedy, and apply it, as shown in Figure 10-3.

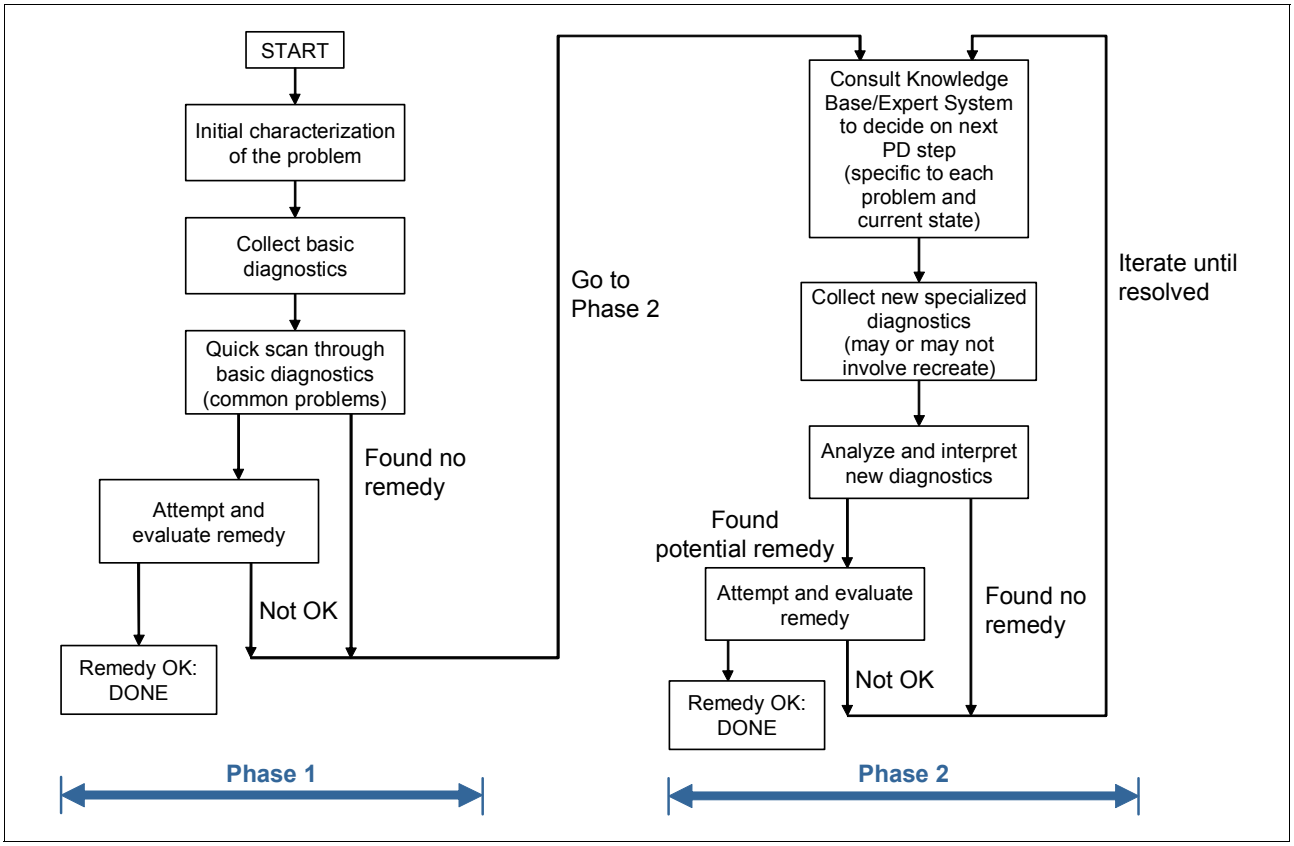


Figure 10-3 Phase 1 and Phase 2 flowcharts

As shown in Figure 10-3, it might be the case that the problem is more complex and the remedy that you apply does not fix the problem, or no remedy is found. In that case, you need to move on to a *Phase 2* investigation. The Phase 2 investigation will be discussed in 10.1.8, “Advanced investigation: Phase 2 problem determination techniques” on page 407.

Initial characterization of the problem

Problem characterization has already been discussed in “Characterize the problem and identify problem symptoms” on page 402. Therefore, *diagnostic data collection* will be discussed next.

Collecting diagnostic data

If you have gone through the problem identification process described earlier, then you have a specific issue to investigate, such as 100% CPU usage, server crash, hung server, Out of Memory error, and so on. You should next refer to IBM MustGather documentation at:

`http://www.ibm.com/support/docview.wss?rs=180&context=SSEQTP&uid=swg21145599`

These documents explain what data to collect and how to collect it for dozens of well-known problems.

Scan diagnostic data

Look for exceptions and in the server's SystemOut.log and SystemErr.log files. The error message itself will often give a reason why the exception was thrown. This information may be sufficient for solving the problem. For example, suppose you are trying to start a server, but the server fails to start. You look in the SystemOut.log and see a message like the following in Example 10-2. You can see that message describes why the server failed to start (`Could not register with Location Service Daemon`) and suggests a user action to perform (`Make sure the NodeAgent for this node is up an running.`).

Example 10-2 SystemOut.log error message

```
[timestamp] 0000000a WsServerImpl  E   WSVR0009E: Error occurred during startup
com.ibm.ws.exception.RuntimeError: com.ibm.ws.exception.RuntimeError:
com.ibm.ejs.EJSException:
Could not register with Location Service Daemon, which could only reside in the
NodeAgent.
Make sure the NodeAgent for this node is up an running.; nested exception is:
        org.omg.CORBA.ORBPackage.InvalidName:
LocationService:org.omg.CORBA.TRAN...
<...stack trace removed...>
```

If you are able to access the administrative console, it is helpful to look at the runtime error messages. Select **Troubleshooting** → **Runtime Messages** → **Runtime Error**. You will see a formatted message, as shown in Example 10-3.

Example 10-3 Admin console Runtime error message

```
Message
   Uncaught exception thrown in one of the service methods of the servlet:
/error.jsp.
   Exception thrown : java.lang.ClassCastException: java.lang.OutOfMemoryError
   at com.ibm._jsp._error._jspService(_error.java:131)
   at com.ibm.ws.jsp.runtime.HttpJspBase.service(HttpJspBase.java:85)
   at javax.servlet.http.HttpServlet.service(HttpServlet.java:856)
   at
com.ibm.ws.webcontainer.servlet.ServletWrapper.service(ServletWrapper.java:966)
   at com.ibm.ws.webcontainer.servlet.ServletWrapper.handleReques
Message type
   Runtime Error
Explanation
   Uncaught exception thrown in one of the service methods of the servlet.
User action
   Examine the server logs in order to determine the root cause of the problem.
   If there are no related messages or they do not help to resolve the problem,
   please contact IBM Support.

Message Originator
   com.ibm.ws.webcontainer.servlet.ServletWrapper
Source object type
   RasLoggingService
Timestamp
   Nov 30, 2007 6:29:23 AM GMT+10:00
Thread Id
   33
Node name
   edgeNode01
```

```
Server name
   server1
```

In general, Phase 1 problems once identified, have well documented solutions. If a quick scan of the log files does not provide explicit information or the log file exceptions and stack traces are too difficult to analyze, you might try searching the IBM APAR database for key words that relate to the problem you are having. You can use the Search feature the IBM Support Assistant to search all IBM documentation, APARs, Technotes, Information Centers, and so forth. For more information about the IBM Support Assistant, go to:

http://www.ibm.com/software/support/isa/

10.1.8 Advanced investigation: Phase 2 problem determination techniques

If the fix or remedy you applied during the Phase 1 investigation did not fix the problem or you could not find any information relating to the problem, then you need to proceed to a Phase 2 investigation as shown in Figure 10-3 on page 405. In general this more advanced phase of investigation requires you to research the available WebSphere documentation, identify one or more problems, and apply fixes in an iterative fashion.

A good place to begin your search of the WebSphere documentation is, again, the IBM Support Assistant. It might also be helpful to read the following problem determination IBM Redbooks publications:

- *WebSphere Application Server V6 Problem Determination for Distributed Platforms*, SG24-6798
- *WebSphere Application Server V6.1 Problem Determination: IBM Redpaper Collection*, SG24-7461

10.2 Problem determination tools

WebSphere provides numerous problem determination tools. Many troubleshooting tools can be accessed through the WebSphere Administrative Console. These tools include:

- Logs and Trace facility
- Class Loader Viewer
- Diagnostic Providers
- Tivoli Performance Viewer

Also there are numerous tools that can be run from the command line in the <was_home>/bin directory, such as:

- **dumpNameSpace.sh**
- **installver.sh**
- **ivt.sh**
- **EARExpander.sh**

All of these tools and many others are documented in the *WebSphere Application Server 6.1 Information Center* at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp

In the rest of this section, additional problem determination tools that are useful in a Solaris 10 environment will be discussed.

10.2.1 IBM Support Assistant portable collector tool

Typically, IBM Support Assistant is installed on a Windows or Linux desktop. If you want to use the collector feature of ISA to collect PID data on your Solaris machine, you can use the Service feature of the IBM Support Assistant to create a *portable collector*.

Use ISA to create a portable collector for WebSphere Application Server

Do the following steps:

1. Start ISA on your Windows or Linux desktop and select the **Service** feature.

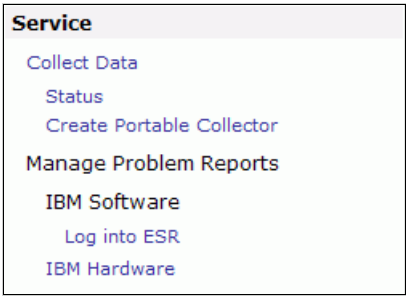


Figure 10-4 ISA Service menu

2. Click **Create Portable Collector** from the ISA Service menu.

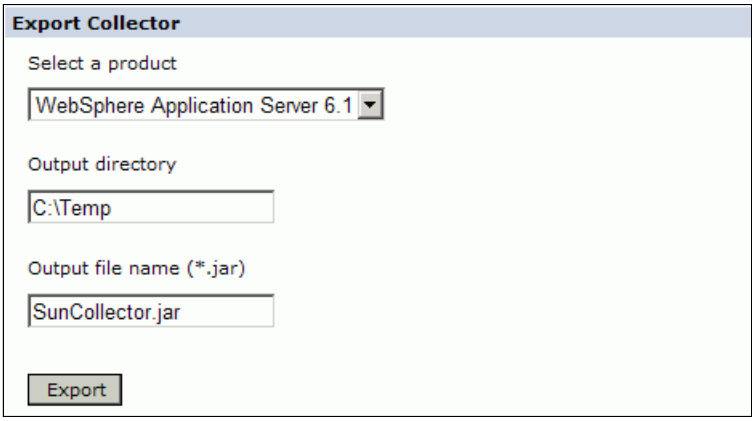


Figure 10-5 ISA Export Collector

3. Select your WebSphere product from the drop-down menu. Enter a local Output directory. Enter a descriptive name for the file. The file name *must* include the.jar extension, as shown in Figure 10-5
4. Click the **Export** button.

Figure 10-6 ISA Export Status

5. You should see a success message, as shown in Figure 10-6.

Once you create the portable collector, you can FTP the jar file to your Solaris machine.

Note: You should realize that though the portable collector file has a jar extension it is not to be executed with the **`java -jar`** command. Rather, the file is compressed like a zip file and must be extracted on your Solaris machine, as shown in "Run the portable collector on the Solaris machine".

Run the portable collector on the Solaris machine

Do the following steps:

1. Create a directory for the collector on the Solaris machine and FTP the SunCollector.jar file to that directory.
2. Extract the jar file using the **`jar xvf SunCollector.jar`** command.
3. Change permissions on the collector files to make them executable by issuing the **`chmod -R 755 *`** command.
4. You may need to define the JAVA_HOME variable as follows:

   ```
   export JAVA_HOME=/opt/IBM/WebSphere/AppServer/java
   ```

5. Run the collector by entering the **`./startcollector.sh`** command.

The collector will run in interactive mode prompting you to input information from the console. Example 10-4 shows what an interactive session with the collector might look like.

Example 10-4 Interactive collector session

```
edge-bash# ./startcollector.sh
logger params set to
Start collector tool ......
AUTOPD_HOME is /tmp
JAVA_HOME is /opt/IBM/WebSphere/AppServer/java/bin/
Enter the Output Filename/Path (must be a local path, and have a .zip file
extension):
(example: <pmrNumber>.<branchNumber>.<countryCode>.<desc>.zip):

1234.137.us.collectWas.zip

Enter yes to create the collection zip file at /tmp/1234.137.us.collectWas.zip

yes

Enter the number or title of the Automated Problem Determination tool collection
option you want or enter RECOVER to recover from a previous Automated Problem
Determination Tool collection or enter QUIT to end the tool
```

```
1: WebSphere Application Server

1

Enter the number or title of the Automated Problem Determination tool collection
option you want or enter RECOVER to recover from a previous Automated Problem
Determination Tool collection or enter QUIT to end the tool
1: General
2: Security and Administration
3: Runtime
4: Connector
5: HTTP
6: Service Oriented Architecture
7: WebSphere Application Server for z/OS
8: Return to Previous Menu

1

Enter the number or title of the Automated Problem Determination tool collection
option you want or enter RECOVER to recover from a previous Automated Problem
Determination Tool collection or enter QUIT to end the tool
1: General Problem
2: RAS Collector Tool
3: Analysis Report
4: Collect Product Information
5: Return to Previous Menu

1

************************************************
* Input Required
* WebSphere Application Server root directory</opt/IBM/WebSphere/AppServer>:
/opt/IBM/WebSphere/AppServer
************************************************
OPTIONS FOR COMPLETING THE INPUT DIALOG
1: OK<Continue the collection using the values you set during the INPUT DIALOG>
2: Cancel<Stop the collection>
1
************************************************
* Input profile name of your WebSphere Application Server
* WebSphere Application Server profile name:
* 1: Dmgr01
* 2: AppSrv01
2
************************************************
OPTIONS FOR COMPLETING THE INPUT DIALOG
1: OK<Continue the collection using the values you set during the INPUT DIALOG>
2: Cancel<Stop the collection>
1
************************************************
* Input the server name of your WebSphere Application Server.
* WebSphere Application Server server name:
* 1: server1
* 2: nodeagent
1
```

```
************************************************
OPTIONS FOR COMPLETING THE INPUT DIALOG
1: OK<Continue the collection using the values you set during the INPUT DIALOG>
2: Cancel<Stop the collection>
1
************************************************
* Input WebSphere Application Server Admin Information
* Admin User Name
wasadmin
* Administrator Password:
*
secret
************************************************
OPTIONS FOR COMPLETING THE INPUT DIALOG
1: OK<Continue the collection using the values you set during the INPUT DIALOG>
2: Cancel<Stop the collection>
1

************************************************
* Question Regarding How the Script Should Proceed
* You will be asked in a moment to choose how you want the
* script to enable tracing for the application server.  You
* must choose whether to enable tracing dynamically or by
* restarting the server.
* If you choose to enable tracing by restarting the server,
* the application server will be stopped, trace settings
* will be modified, and the application server will be
* restarted.
* If you choose to enable tracing without restarting the
* server, trace settings will be modified dynamically.
* When the collection has been completed, the startup trace
* specification (and the runtime trace if the server was
* running when the collection began) will be restored to
* their original values.
* Select one::
* 1: Enable Tracing by Restarting the Server
* 2: Enable Tracing without Restarting the Server
1
************************************************
OPTIONS FOR COMPLETING THE INPUT DIALOG
1: OK<Continue the collection using the values you set during the INPUT DIALOG>
2: Cancel<Stop the collection>
1
************************************************
* Proceeding with enabling tracing by restarting the server
************************************************
OPTIONS FOR COMPLETING THE INPUT DIALOG
1: OK<Continue the collection using the values you set during the INPUT DIALOG>
2: Cancel<Stop the collection>
2
```

<...Note: The interactive session was stopped as this point...>

Notice from the interactive session in Example 10-4 on page 409 that the results of the collection will be stored in the zip that you specified. In this example, the file is 1234.137.us.collectWas.zip. The zip file will contain a number of reports in HTML format and other information depending on the options and trace strings that you specify during the interactive session.

10.2.2 Heap Profiler (HPROF)

HPROF is a simple profiler agent shipped with JDK 5.0. It is a dynamically-linked library that interfaces to the VM using the Java Virtual Machine Tools Interface (JVM TI). It writes out profiling information either to a file or to a socket in ASCII or binary format. This information can be further processed by a profiler front-end tool.

HPROF is capable of presenting:

- CPU usage
- Heap allocation statistics
- Monitor contention profiles

In addition, it can also report complete heap dumps and states of all the monitors and threads in the JVM. In terms of diagnosing problems, HPROF is useful when analyzing:

- Performance
- Lock contention
- Memory leaks

HPROF is invoked as follows:

```
$ java -agentlib:hprof <ToBeProfiledClass>
```

A complete list of options, as shown in Example 10-5, is printed if the HPROF agent is provided with the help option:

```
$ java -agentlib:hprof=help
```

Example 10-5 HPROF options

```
HPROF: Heap and CPU Profiling Agent (JVMTI Demonstration Code)

hprof usage: java -agentlib:hprof=[help]|[<option>=<value>, ...]

Option Name and Value  Description                    Default
---------------------  -----------                    -------
heap=dump|sites|all    heap profiling                 all
cpu=samples|times|old  CPU usage                      off
monitor=y|n            monitor contention             n
format=a|b             text(txt) or binary output     a
file=<file>            write data to file             java.hprof[.txt]
net=<host>:<port>      send data over a socket        off
depth=<size>           stack trace depth              4
interval=<ms>          sample interval in ms          10
cutoff=<value>         output cutoff point            0.0001
lineno=y|n             line number in traces?         y
thread=y|n             thread in traces?              n
doe=y|n                dump on exit?                  y
msa=y|n                Solaris micro state accounting n
force=y|n              force output to <file>         y
```

```
verbose=y|n            print messages about dumps     y

Obsolete Options
----------------
gc_okay=y|n

Examples
--------
  - Get sample cpu information every 20 millisec, with a stack depth of 3:
      java -agentlib:hprof=cpu=samples,interval=20,depth=3 classname
  - Get heap usage information based on the allocation sites:
      java -agentlib:hprof=heap=sites classname

Notes
-----
  - The option format=b cannot be used with monitor=y.
  - The option format=b cannot be used with cpu=old|times.
  - Use of the -Xrunhprof interface can still be used, e.g.
       java -Xrunhprof:[help]|[<option>=<value>, ...]
    will behave exactly the same as:
       java -agentlib:hprof=[help]|[<option>=<value>, ...]

Warnings
--------
  - This is demonstration code for the JVMTI interface and use of BCI,
    it is not an official product or formal part of the J2SE.
  - The -Xrunhprof interface will be removed in a future release.
  - The option format=b is considered experimental, this format may change
    in a future release.
```

For additional information about HPROF and examples, read the *Java 2 Platform, Standard Edition 5.0 Troubleshooting and Diagnostic Guide*, found at:

http://java.sun.com/j2se/1.5/pdf/jdk50_ts_guide.pdf

10.2.3 Heap Analysis Tool (HAT)

The Heap Analysis Tool (HAT) is used to troubleshoot *unintentional object retention* (memory leaks). This term is used to describe a condition where an object is no longer needed by an application but is kept alive due to references from the rootset. This can happen, for example, if an unintentional static reference to an object remains after the object is no longer needed. Unintentional object retention is the Java Language equivalent of a memory leak. HAT can be used to browse the object topology in a heap snapshot. The tool allows a number of queries, for example, *show me all reference paths from the rootset to this object*. It is this query that is most useful for finding unnecessary object retention. HAT is not included in JDK5.0, but it can be downloaded from the following site:

http://hat.dev.java.net

HAT requires a heap dump in binary format as input. There are several ways to generate a heap dump:

- The application can be run with the HPROF profile, as shown in 10.2.2, “Heap Profiler (HPROF)” on page 412.

- As of Java SE 5.0 update 7, the -XX:+HeapDumpOnOutOfMemoryError command-line option tells the HotSpot VM to generate a heap dump when an OutOfMemoryError occurs. By default, the dump will be written to the working directory of the JVM. For WebSphere Application Server, the default directory will be <was_home>/profiles/<*profile_name*>. The file will be named java_pid<*pid*>.hprof, where <*pid*> is the process ID of the JVM. The default directory can be changed by using the -XX:HeapDumpPath= option. For example, -XX:HeapDumpPath=/disk1/dumps will cause the heap dump to be generated in the /disk1/dumps directory.

Note: The impact associated with using the -XX:+HeapDumpOnOutOfMemoryError is minimal, so using it on a production server should not impact performance.

- As of Java SE 5.0 update 14, the -XX:+HeapDumpOnCtrlBreak command-line option tells the HotSpot VM to generate a heap dump when a Ctrl-Break or SIGQUIT signal is received.

HAT queries

The following is a brief description of the different queries available with HAT.

- All Classes Query

  Shows all classes in a heap, excluding platform classes.

- Class Query

  Shows information about a specific class. Includes its superclass, subclasses, instance data members, and static data members.

- Object Query

  Provides information about a specific object. Within this query, you can navigate to the object's class and to the value of object members of the object. You can also see objects that refer to the current object.

- Instances Query

  Shows all instances of a given class.

- Roots Query

  Provides reference chains from the rootset to a given object. It shows a chain for each member of the rootset from which a specific object is reachable. The chains are calculated using a depth-first search, thus providing chains with minimal length. This query usually is the most useful query for troubleshooting memory leaks (unintentional object retention) since once you find a retained object, the query can tell you *why* it is being retained.

- Reachable Objects Query

  This query can be accessed from an Object query and shows the transitive closure of objects that are reachable from a specific object.

- Instance Counts for All Classes Query

  Shows counts of class instances for every class, excluding platform classes.

- All Roots Query

  Shows all members of the rootset.

- New Instances Query

  This query is only available if you invoke HAT with two heap dumps. Similar to the Instance query, except it can find new instances by comparing the recent heap dump with the older heap dump.

Using the HAT tool

Suppose an application throws an OutOfMemoryError exception and a heap dump called java_pid18014.hprof is generated. You can invoke HAT on the heap dump by issuing the following command:

```
#hat java_pid18014.hprof
```

The contents of the HAT analysis will become available on an HTTP server that uses port 7000 by default, but you can specify another port on the command line.

The output from the command is shown in Example 10-6.

Example 10-6 HAT command line output

```
edge-bash# hat java_pid18014.hprof

Started HTTP server on port 7000
Reading from java_pid18014.hprof...
Dump file created Mon Dec 03 06:47:14 GMT+10:00 2007
Snapshot read, resolving...
Resolving 619987 objects...
Chasing references, expect 1239 dots.....................
..................
Snapshot resolved.
Server is ready.
```

As you can see, an HTTP server is launched, which allows you to browse the different HAT queries described in “HAT queries” on page 414.

You can access the HTTP server using port 7000, as shown in Figure 10-7, and browse through all the queries.

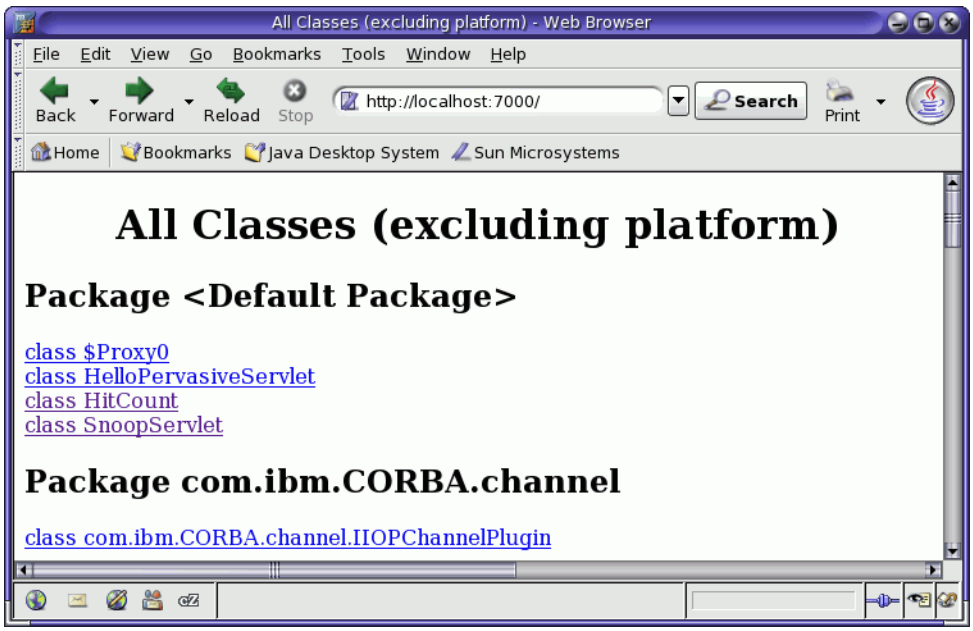


Figure 10-7 Browsing the HAT output

Figure 10-8 shows a portion of the rootset query that you can navigate to from the page shown in Figure 10-7 on page 415.

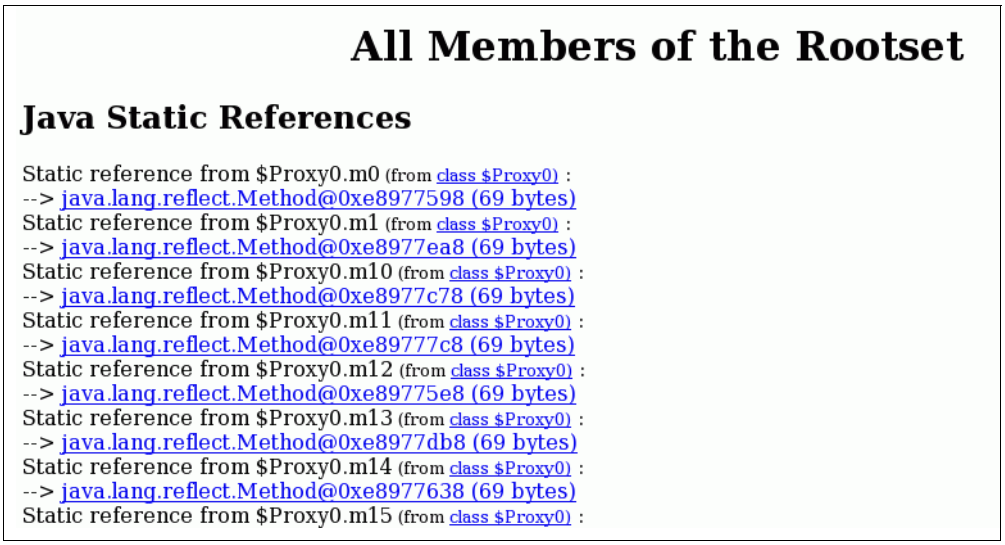


Figure 10-8 Rootset query

HAT can be a very useful tool for resolving JVM heap-related problems. For more details on how to use it, see the HAT documentation found at:

`http://hat.dev.java.net/doc/README.html`

10.2.4 Java Heap Analysis Tool (JHAT)

The Java Heap Analysis Tool (JHAT) is an enhanced version of HAT, but uses fewer resources and has less impact on performance. Some of the JHAT enhancements and improvements include the following:

- JHAT can parse incomplete or truncated heap dumps.
- JHAT can read heap dumps generated on 64-bit systems.
- JHAT supports the Object Query Language (OQL) for writing your own queries on the heap dump.

The JHAT utility is part of Java SE 6, but can read and analyze heap dumps created on Java SE 5.0 systems. If you have access to a machine with Java SE 6 installed, you can create a heap dump on the Java SE 5.0 system and transport it to the Java SE 6 system in order to parse and browse it with the JHAT utility.

For detailed information about JHAT, see the *Troubleshooting Guide for Java SE 6 with HotSpot VM*, found at:

`http://java.sun.com/javase/6/webnotes/trouble/TSG-VM/TSG-VM.pdf`

10.2.5 Solaris Operating System tools

The Solaris Operating System provides many native commands that are useful for monitoring and problem determination. Table 10-1 on page 417 provide a list of these tools and a brief description. Some of these tools are described in greater detail in 9.3.2, “Solaris system monitoring” on page 319. For additional details, refer to the man pages or the Solaris 10 Operating System documentation at `http://docs.sun.com/app/docs/coll/47.16`.

Table 10-1 Solaris Operating System tools

OS tool	Description
cpustat	Monitors system performance/behavior using CPU performance counters.
dtrace	Solaris 10: Dynamic tracing of Kernel functions, system calls and user-land functions. Allows arbitrary, safe scripting to be executed at entry/exit and other probe points.
libumem	User space slab allocator (Solaris 9 4/03 and later). It can be used to find and fix memory management bugs
iostat	Reports I/O statistics.
netstat	Displays the contents of various network-related data structures.
mdb	Modular debugger for kernel and user apps and crash dumps.
pfiles	Prints information about process file descriptors. Solaris 10 version prints the file name as well.
pldd	Prints shared objects loaded by a process.
pmap	Prints the memory layout of a process or core file: heap, data, text sections. Solaris 10 version clearly identifies stack segments with [stack] along with the thread ID.
prun	Sets the process to running mode (reverse of **pstop**).
prstat	Reports active process statistics.
psig	Lists the signal handlers of a process.
pstack	Prints the stack of threads of a given process or core file. Solaris 10 version can print Java method names for Java frames.
pstop	Stops the process (suspend).
ps	Lists all processes.
ptree	Prints the process tree starting at a given PID.
sar	System activity reporter.
truss	Traces entry/exit event for system calls, user-land functions, signals, and, optionally, stops the process at one of these events. Prints arguments as well.
vmstat	Reports system virtual memory statistics.
gcore	Utility to force a core dump of a process. The process continues after the core dump is written.

10.3 Common WebSphere Application Server problems

In this section, some of the more common WebSphere Application Server problem areas will be briefly described. There is plenty of documentation available in the WebSphere Application Server V6.1 Information Center and in several Redbooks concerning problem determination. Specific resources will be referenced in the subsequent sections.

10.3.1 Installation problem determination

Installation problems are well documented in the WebSphere Application Server V6.1 Information Center as well as *WebSphere Application Server V6 Problem Determination for Distributed Platforms*, SG24-6798. In this section, some common installation problems will be discussed. The installation logs that contain important diagnostic data for troubleshooting installation issues will also be examined and discussed.

Common installation problems

Common installation problems include the following.

- Launchpad or installation wizard fails to start

  If the launchpad or installation wizard fails to start, this is typically because you are not using a supported browser or the browser is not properly configured.

- Silent installation fails because of errors in the response file

  Syntax errors in the response file can cause a silent installation to fail. Also be aware that the information you provide in the response file is case sensitive.

- The installation process fails

  The installation process itself might fail because of low system resources such as disk space and virtual memory.

- Profile creation fails

  Creation of profiles might fail because of insufficient file system permissions and port conflicts.

- Running the Installation Verification Test (IVT)

  Running the IVT may fail to start the application server. This can happen for a number of different reasons. There may be port conflicts or any of the server's configuration files may be corrupt. Examine the server's SystemOut.log for details of a failed server start.

  If the server starts successfully, IVT tries validate server components, such as the Web container and EJB container. If any of these tests fail, then it may be because of a configuration problem. The console messages provided by IVT may be helpful for determining the problem. Also, examine the IVT client log at <was_home>/profiles/<*profile_name*>/logs/ivtClient.log.

- Running the Installation Verification Utility (IVU)

  The Installation Verification Utility (installver.sh) can be used to detect and diagnose initial installation issues. It computes the actual checksum value for the installed files and compares it to the shipped bill-of-materials list. IVU will report missing or changed files. Check the IVU default log file at <was_home>/logs/installver.log for any reported issues.

Installation files and diagnostic data

The following logs and traces are located in the directory <was_home>/logs/install/:

- The log.txt installation log

  This log contains information about all events that occur during installation. At the end of the log, you will see one of the following flags:

  - INSTCONFSUCCESS
  - INSTCONFPARTIALSUCCESS
  - INSTCONFFAILED

  If you see a partial success or failed message, try to determine what WebSphere component failed to install and look for additional messages about why the installation failed.

- trace.txt.gz and trace.xml.gz

  These are two compressed forms of the same installation information. As each component is installed, one of the flags listed above is written to the log. If an installation fails, these traces will show you exactly what component failed and a brief message saying why.

- installconfig.log.gz

  A log of the configuration events that occur during installation.

Profile creation logs

For each profile created, there is a profile creation log file located at <was_home>/logs/install/manageprofiles. Each profile's creation log will be named using the format <profile_name>_create.log, for example, Dmgr01_create.log.

If problems arise during profile creation, examine the creation log for failure messages.

In addition to the creation log, several log files are written to when the profile is created. These additional logs are located in the directory <was_home>/logs/install/manageprofiles/<profile_name>. The number and type of logs depend on the kind of profile (deployment manager, application server, or custom)

Some of the logs for a deployment manager profile include the following.

- collect_metadata.log
- createDefaultServer.log
- defaultapp_config.log
- filetransfer_config.log
- keyGeneration.log
- SetSecurity.log
- SIBDeployRA.log

Check these logs after installation to make sure that all components of the profile were configured successfully.

Recovering from a failed installation

If an installation fails, in most cases you can run the uninstall program. However, if the installation failed at an early stage before the uninstall program was created, you will have to manually uninstall. The process for manually uninstalling WebSphere on a Solaris system is well documented in the WebSphere Application Server V6.1 Information Center at http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tins_uninstallSolaris.html.

10.3.2 Database connection problem determination

This section will describe in detail how to troubleshoot database connection problems.

Introduction

The JCA specification provides a standard mechanism that allows modern J2EE applications to connect and use heterogeneous resources from various existing enterprise information systems (EIS) as well as modern relational database systems. Based on the JCA, WebSphere provides client applications to all system services regarding connection, transaction, and security management on behalf of the resource managers.

JCA involves four main components:

- Application server
- Application component
- Resource adapter
- EIS

It also specifies a requirement for packaging and deployment facilities for a resource adapter to plug into an application server. Figure 10-9 illustrates the contracts or relationships between these four components.

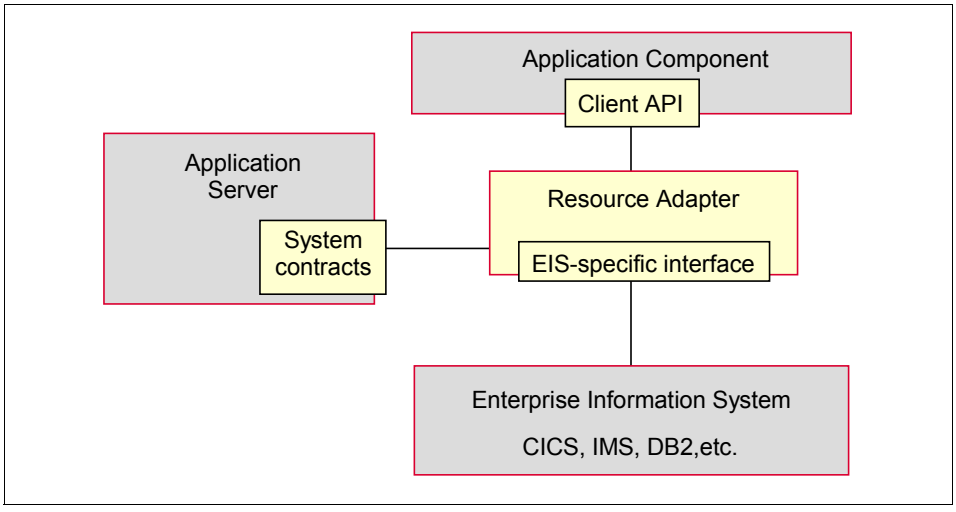


Figure 10-9 JCA overview

Based on the JCA specification, an EIS vendor can develop a standard resource adapter (RA) for its EIS to plug into any application sever that supports JCA. A resource adapter runs within the address space of an application server, while the EIS itself runs in a separate address space. For example, a DB2 database (EIS) and WebSphere Application Server each run in a separate machine. An application component is able to access the EIS through the resource adapter.

The relationships between the four major components can be described as follows:

- The contracts between the application component and the resource adapter are provided through some form of client API. The client API can either be specific to a particular type of resource adapter, for example, JDBC for relational data base, or a standard common client interface (CCI). The JCA recommends, but does not require, that a resource adapter implement the CCI. WebSphere Application Server V6 provides a relational resource adapter (RRA) that has an implementation for both the CCI and the traditional JDBC interfaces.
- The resource adapter and application server implement system contracts to provide the common mechanisms for connection, transaction, and security management.
- The contracts between the resource adapter and the EIS are specific to each underlying EIS. Thus, JCA does not impose any requirement on this proprietary relationship.

This section will discuss problems that are experienced during connection to enterprise information systems or databases using JCA.

Users with the following initial symptoms might be experiencing a JCA-related problem:

- Symptom: A JDBC call returns incorrect data.
- Symptom: Failure to connect to a new data source.
- Symptom: Failure to connect to an existing data source.
- Symptom: Failure to access a resource through JDBC.
- Symptom: Failure to access a non-relational resource.

Such symptoms can be observed in the WebSphere Application Server JVM logs in error messages with any of the following prefixes: WTRN, J2CA, WSCL, or DSRA.

Figure 10-10 shows the initial symptoms that you might experience when you have a JCA connection problem.

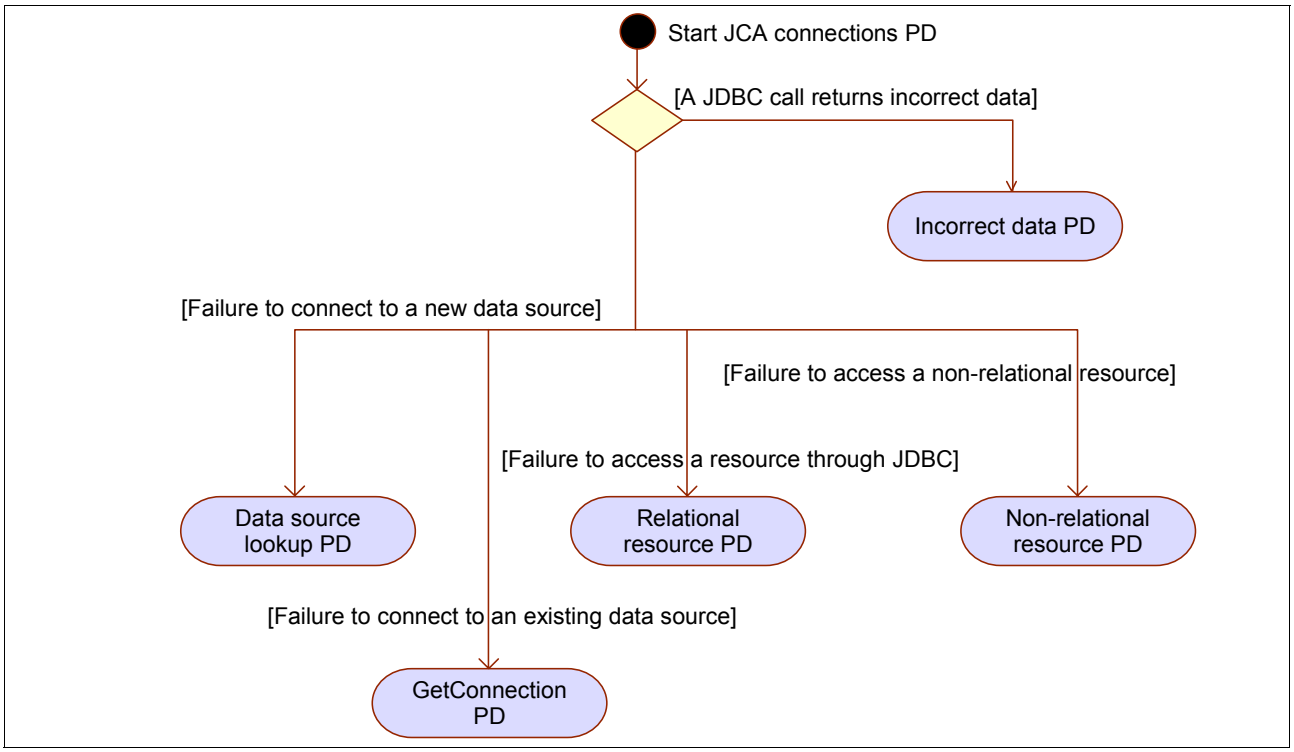


Figure 10-10 JCA connections - initial symptom analysis

Symptom: A JDBC call returns incorrect data

With this symptom, your application is running normally, but you receive incorrect data. For example, your database maintains a list of product inventory, but some product quantities or prices that are reported from your application are not in line with your expectations.

Rule out a database or JDBC driver problem

If you suspect data is being incorrectly returned to the application, try running the same query using the database's native client tools (such as Oracle SQL*Plus, Sybase ISQL, Informix dbaccess, DB2 Command Line Processor, or SQL Server Query Analyzer). Alternatively, try to run the same query in a stand-alone JDBC program using the JDBC driver directly. If the data returned from the client tool is the same as the data that your application returned when running in WebSphere Application Server, then it is likely that the database has a problem. In this case, you need to engage the database vendor for further help.

If the data in the database is correct, try to write a stand-alone JDBC program that runs the same query to see if the same symptom occurs. If the data returned from the stand-alone JDBC program is the same as the data that your application returned, then problem is with the JDBC driver. In this case, you need to contact the JDBC vendor for further help.

What to look for

If you have determined that the problem is not caused by the database or JDBC driver, you can continue the analysis by looking for the situation in which the error occurs. The best way to do this is to perform a number of runs through the application, both from a fresh restart and repeated runs within the same application session.

As shown in Figure 10-11, you might be in one of the following two situations:

- Sporadically incorrect data, which means that the data that is returned to you is correct sometimes but not always.

  If your application uses a prepared statement or callable statement, you can experience a problem with WebSphere's statement cache.

  A prepared statement is a precompiled SQL statement that is stored in a prepared statement object for parameterized queries. This object is used to run the given SQL statement efficiently multiple times. A callable statement is used to invoke stored procedures.

  In general, the more prepared statements and callable statements that your application has, the larger the cache should be. Be aware, however, that specifying a larger statement cache size than needed wastes application memory and does not improve performance.

  Determine the value for your cache size by adding the number of uniquely prepared statements and callable statements (as determined by the SQL string, concurrency, and the scroll type) for each application that uses this data source on a particular server. This value is the maximum number of possible prepared statements and callable statements that are cached on a given connection over the life of the server.

  You can try to disable the statement cache by setting Statement cache size to 0. This setting can be found in the administrative console by selecting **Resources** → **JDBC Providers** → **<JDBC_provider>** → **Data sources** → **<data_source>** → **WebSphere Application Server connection properties**.

  You should call IBM Technical Support to help you with this scenario.

- Consistently incorrect data is when the data that is returned to you is always incorrect. In this case, it is highly possible that your application is connected to the wrong database or resource manager. In this case, you might have a problem with your environment or configuration.

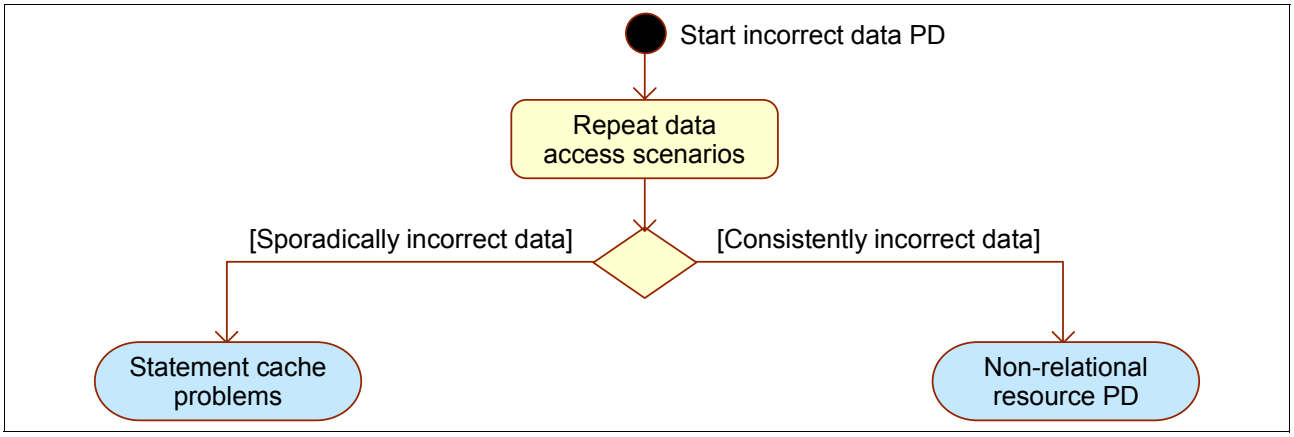


Figure 10-11 Incorrect data problem determination

10.3.3 Symptom: Failure to connect to a new data source

When you start a new working session in your application, the application first needs to look for a data source that provides the data. Any error at this early stage can be broadly identified as a failure on new data source.

Data to collect

All relevant error messages and exceptions needed for our analysis are available in SystemOut.log and SystemErr.log.

What to look for

To proceed with further analysis, you need to run a tool called TestConnection service. WebSphere Application Server provides a test connection service for testing connections to the data sources that you configure for database access. This test connection service can be activated in three different ways:

- Through the administrative console
- Using the **wsadmin** tool
- With a Java stand-alone program

For more information, see Test connection service in the WebSphere Information Center at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/cdat_testcon.html

Depending on the messages or exceptions produced by TestConnection, you might need to perform the additional actions shown in Figure 10-12.

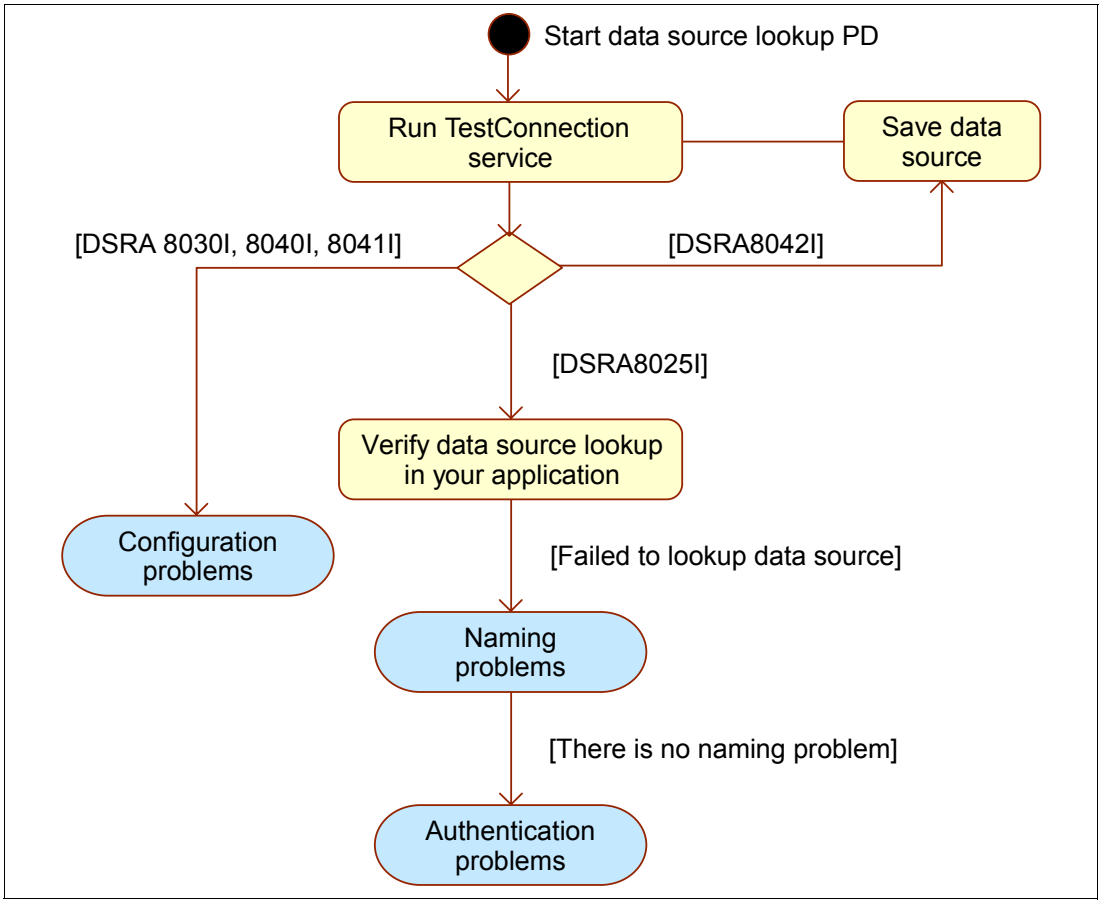


Figure 10-12 Data source lookup problem determination

Testing the connection results in one of the following outcomes:

- Message DSRA8042I

  This message indicates that the data source does not exist.

  If you created the data source but have not saved the configuration, the test connection will not find the data source. You need to save the data source and then re-run the TestConnection service.

- Messages DSRA8030I, DSRA8040I, or DSRA8041I

  These messages indicate that your connection is either failing or having problems. In this case, the TestConnection service produces additional DSRA messages in the range of 8000 to 8499 in the SystemOut.log.

- Message DSRA8025I

  This message indicates that the TestConnection service can connect to the data source successfully. There is nothing wrong with the data source configuration. So, the next step is to verify the data source lookup code in your application. One possible way to perform this verification is to run the application through a debugger with a breakpoint set after the lookup() method call on your java.naming.Context object. Another way is to create and run a small program with the data source lookup code cut and pasted from your application.

  If the data source lookup is successful, the new data source is OK, and you should go to "The next step" on page 428.

  If the data source lookup is not successful, you might have a naming or authentication problem.

  If there is no problem with naming, determine if you have a problem authenticating with the database. Refer to Chapter 6, "JCA connection problem determination", in *WebSphere Application Server V6 Problem Determination for Distributed Platforms*, SG24-6798 for information about this problem.

If you reach this point without identifying the problem, go to "The next step" on page 428.

Symptom: Failure to connect to an existing data source

In the situation where a current session with the application has been working or a new session is started with no errors on naming or authentication, you can assume that the application was successful in finding the data source (or connection factory). A failure to establish connectivity at this stage is broadly identified as a failure to get a new connection to an existing data source.

Data to collect

All relevant error messages and exceptions that are needed for our analysis are available in SystemOut.log and SystemErr.log.

What to look for

You need to evaluate the situation when there is a failure on getting a connection. One possible way to do this is to run the application through a debugger with a breakpoint set after the getConnection() method call on your javax.resource.cci.ConnectionFactory or javax.sql.DataSource object. Another way is to create and run a small program with the getting connection code cut and pasted from your application. You can run this test case many times and observe the outcomes, as shown in Figure 10-13.

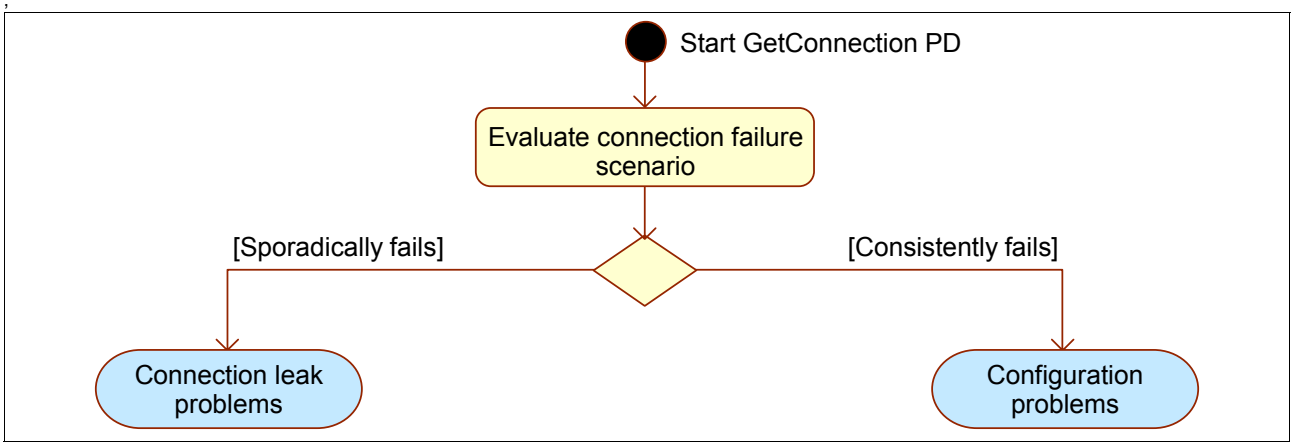


Figure 10-13 Problem determination path for failed connections

Examining the situation where there is a failure in getting a new connection can lead you to one of the following two cases:

- Connection consistently fails, which means that you might have a problem with your application configuration.
- Connection fails sporadically, which means that it is very likely that you have a connection leak.

Refer to Chapter 6, “JCA connection problem determination”, in *WebSphere Application Server V6 Problem Determination for Distributed Platforms*, SG24-6798 for more information.

If none of these symptoms apply, go to “The next step” on page 428.

Symptom: Failure to access a resource through JDBC

When an application uses JDBC to access a relational database and the access fails, you most likely receive error messages with the prefixes `WTRN` (transaction problem) or `DSRA` (data source resource adapter).

Data to collect

All relevant error messages and exceptions that are needed for our analysis are available in SystemOut.log and SystemErr.log.

What to look for

Examine the exceptions in SystemOut.log and SystemErr.log for more information, as shown in Figure 10-14 on page 427.

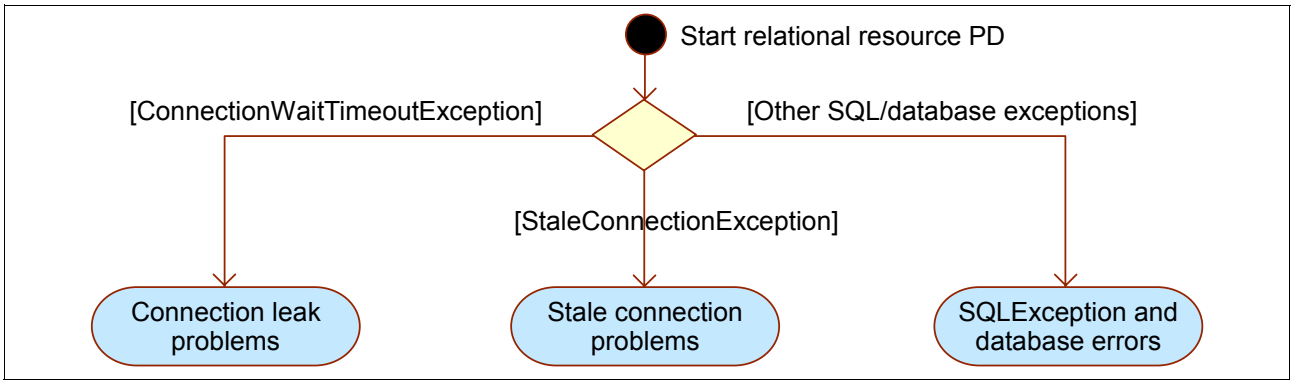


Figure 10-14 Relational resource problem determination

Depending on what you find, you may need to investigate the following items:

- ConnectionWaitTimeoutException
- StaleConnectionException
- Other SQL/database exceptions

Refer to Chapter 6, “JCA connection problem determination”, in *WebSphere Application Server V6 Problem Determination for Distributed Platforms*, SG24-6798 for more information.

If none of these symptoms apply, go to “The next step” on page 428.

Symptom: Failure to access a non-relational resource

When your application uses non-relational resource adapters, such as CICS® ECI, Siebel®, or SAP®, the failure to access a resource usually involves the XAResource that manages transactions or the connection factory that creates connections to the resource.

Data to collect

All relevant error messages and exceptions that are needed for our analysis are available in SystemOut.log and SystemErr.log.

What to look for

In most situations, you receive error messages with prefixes `WTRN` (transaction problem) or `J2CA` (general JCA connector problem). The errors can be classified into scenarios, as shown in Figure 10-15.

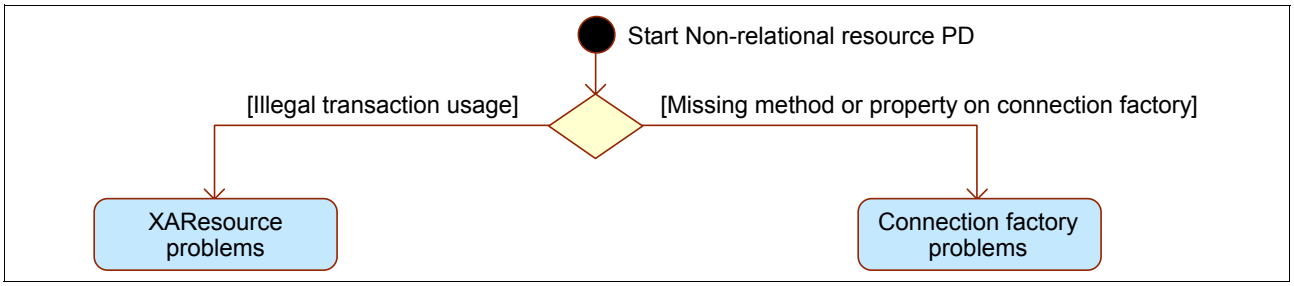


Figure 10-15 Non-relational resource problem determination

Depending on what you find, you may have to investigate the following items:

- Illegal transaction usage
- Missing method or property on connection factory

Refer to Chapter 6, “JCA connection problem determination”, in *WebSphere Application Server V6 Problem Determination for Distributed Platforms*, SG24-6798 for more information.

If none of these symptoms apply, go to “The next step” on page 428.

The next step

The symptoms and problem areas included in this section are some that you are more likely to experience. However, there are other things that can go wrong, or the cause of the problem might be related to something other than JCA components.

If, after going through this process, you still have an undiagnosed problem, we recommend that you go back to 10.1.5, “How to organize the investigation after a problem occurs” on page 402 and review the problem classifications to see if there are any other components that might be causing the problem.

10.3.4 Security problem determination

For security problem determination, refer to *Troubleshooting security configurations*, found at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/tsec_trouble.html

10.3.5 Application deployment problem determination

For application deployment problem determination, refer to *Troubleshooting deployment*, found at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/tsec_trouble.html

10.3.6 System management problem determination

For system management problem determination, refer to Chapter 4, “System management problem determination”, in *WebSphere Application Server V6 Problem Determination for Distributed Platforms*, SG24-6798.

10.3.7 Web container problem determination

For Web container problem determination, refer to Chapter 2, “WebSphere Application Server V6.1: Web container problem determination”, in *WebSphere Application Server V6.1 Problem Determination: IBM Redpaper Collection*, SG24-7461.

10.3.8 Web services problem determination

For Web services problem determination, refer to Chapter 3, “WebSphere Application Server V6.1: Web services problem determination”, in *WebSphere Application Server V6.1 Problem Determination: IBM Redpaper Collection*, SG24-7461.

10.3.9 Class loader problem determination

For class loader problem determination, refer to Chapter 4, “WebSphere Application Server V6.1: Class loader problem determination”, in *WebSphere Application Server V6.1 Problem Determination: IBM Redpaper Collection*, SG24-7461.

10.3.10 Workload management problem determination

For workload management problem determination, refer to the following chapters in *WebSphere Application Server V6.1 Problem Determination: IBM Redpaper Collection*, SG24-7461:

- Chapter 5, “WebSphere Application Server load balancing problem determination”
- Chapter 6, “Web server plug-in load balancing problem determination”
- Chapter 7, “EJB workload management problem determination”
- Chapter 8, “High availability manager problem determination”

10.3.11 JMS problem determination

For JMS problem determination, refer to Part 6, “JMS problem determination”, in *WebSphere Application Server V6.1 Problem Determination: IBM Redpaper Collection*, SG24-7461.

10.4 JVM problem determination: Hangs, crashes, and out of memory exceptions

This section focuses on typical JVM problems. In addition to describing the problems, the following discussion is geared toward identifying tools and techniques that are primarily relevant to the Solaris 10 environment.

10.4.1 Hanging and looping processes

Problems involving hanging processes or processes looping indefinitely are common. Hangs can occur for many reasons, but usually result from a deadlock in the code. The code may be:

- Application code
- API/library code

Occasionally, the hang may be due to a bug in the HotSpot Virtual Machine itself.

At times, what may appear to a hang at first turns out to be the JVM process consuming all available CPU cycles. CPU usage of 100% can be caused by a code defect that puts one or more threads into an infinite loop.

An initial step when diagnosing a hang is to determine the status of the JVM process. Is the process idle or is it consuming all available CPU cycles? Various operating system utilities can be used to answer this question.

If the process appears to be working and is consuming all available CPU cycles, then most likely that the problem is a looping thread rather than deadlocked threads.

On Solaris, use **prstat -L -p <pid>** to report the statistics for all LWPs in the target process. This information will enable you to identify the thread(s) that are consuming a large amount of CPU cycles. Example 10-7 shows the output of the **prstat** command run against a Java process.

Example 10-7 prstat output for JVM process

```
PID   USERNAME  SIZE  RSS  STATE  PRI   NICE TIME    CPU  PROCESS/LWPID
18014 root      624M  525M cpu13    9    0   0:02:01 1.5% java/34
18014 root      624M  525M sleep   59    0   0:05:40 0.0% java/134
18014 root      624M  525M sleep   59    0   0:05:15 0.0% java/54
18014 root      624M  525M sleep   59    0   0:05:16 0.0% java/107
```

Troubleshooting a looping process

When a JVM shows symptoms of looping, you should try to trigger a thread dump. Thread dump information usually indicates which thread is looping. If you can determine which thread is looping, then examine the stack trace for that thread. The trace might provide some hints as to where and possibly why the thread is looping.

Thread dumps can be triggered in the following ways:

- Send a SIGQUIT to the process with the following command:

  ```
  kill -QUIT <pid>
  ```

- If the application console (standard input/output) is available, use the following command:

  ```
  Crtl-\
  ```

For a WebSphere Application Server JVM, the thread dump will be written to the native_stdout.log file for the server.

Example 10-8 Thread dump

```
Full thread dump Java HotSpot(TM) Server VM (1.5.0_11-b03 mixed mode):

"WebContainer : 2" daemon prio=10 tid=0x004ca9d0 nid=0x88 runnable [0xc997f000..
0xc997f9f0]
        at com.ibm.io.async.AsyncLibrary.aio_getioev2(Native Method)
        at com.ibm.io.async.AsyncLibrary.getCompletionData2(AsyncLibrary.java:57
5)
        at com.ibm.io.async.ResultHandler.runEventProcessingLoop(ResultHandler.j
ava:524)
        at com.ibm.io.async.ResultHandler$2.run(ResultHandler.java:873)
        at com.ibm.ws.util.ThreadPool$Worker.run(ThreadPool.java:1469)
<..stack trace truncated...>

"WebContainer : 1" daemon prio=10 tid=0x00f8cfd8 nid=0x87 waiting for monitor en
try [0xc9b7e000..0xc9b7f970]
        at com.ibm.issf.atjolin.badapp.BadAppServlet.sneezyMethod(BadAppServlet.
java:338)
        - waiting to lock <0xf4f1bf28> (a java.lang.Object)
        - locked <0xf4f1bf30> (a java.lang.Object)
        at com.ibm.issf.atjolin.badapp.BadAppServlet.doPost(BadAppServlet.java:2
59)
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:763)
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:856)
        at com.ibm.ws.webcontainer.servlet.ServletWrapper.service(ServletWrapper
```

```
.java:989)
        at com.ibm.ws.webcontainer.servlet.ServletWrapper.handleRequest(ServletW
rapper.java:501)
        at com.ibm.ws.wswebcontainer.servlet.ServletWrapper.handleRequest(Servle
tWrapper.java:464)
        at com.ibm.ws.webcontainer.webapp.WebApp.handleRequest(WebApp.java:3276)
        at com.ibm.ws.webcontainer.webapp.WebGroup.handleRequest(WebGroup.java:2
67)
<..stack trace truncated...>

"WebContainer : 0" daemon prio=10 tid=0x00939ed0 nid=0x86 waiting for monitor en
try [0xc9c7e000..0xc9c7f8f0]
        at com.ibm.issf.atjolin.badapp.BadAppServlet.dopeyMethod(BadAppServlet.j
ava:321)
        - waiting to lock <0xf4f1bf30> (a java.lang.Object)
        - locked <0xf4f1bf28> (a java.lang.Object)
        at com.ibm.issf.atjolin.badapp.BadAppServlet.doPost(BadAppServlet.java:2
57)
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:763)
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:856)
        at com.ibm.ws.webcontainer.servlet.ServletWrapper.service(ServletWrapper
.java:989)
        at com.ibm.ws.webcontainer.servlet.ServletWrapper.handleRequest(ServletW
rapper.java:501)
        at com.ibm.ws.wswebcontainer.servlet.ServletWrapper.handleRequest(Servle
tWrapper.java:464)
<..stack trace truncated...>

    <..Remainder of thread dump truncated...>
```

Interpreting the thread dump output

The output consists of the header and stack trace for each thread. The Java threads are printed first, and these are followed by information about VM internal threads.

For each Java thread, there is a header line with information about the thread, and this is followed by the thread stack. The header line prints the *thread name*, indicates if the thread is a *daemon thread*, and also prints the Java *thread priority*, as shown in Example 10-8 on page 430.

In the header, the *tid* is the thread ID that is the address of a thread structure in memory. The *nid* is the ID of the native thread and this is followed by the *thread state*. The thread state indicates what the thread is doing at the time of the thread dump. Table 10-2 shows the possible states that can be printed.

Table 10-2 Thread states

Thread State values
allocated
initialized
runnable
waiting for monitor entry
waiting on condition

in Object.wait()
sleeping

When trying to diagnose a looping thread, a good place to start is the thread stacks of the threads that are in the *runnable* state. It is often necessary to get a sequence of thread dumps in order to determine which threads appear to be continuously busy.

The **jstack** utility can also be used to obtain a stack thread if you cannot trigger a thread dump for any reason. Also try using **jstack** if the thread dump does not show signs that a Java thread is looping. When using **jstack**, look for threads that are in the following states:

- IN_JAVA
- IN_NATIVE
- IN_VM

These are the most likely states for threads that are looping. Again, it is a good idea to run **jstack** several times to better identify the threads that are looping. If you find a thread looping indefinitely while in the IN_VM state, this might indicate a HotSpot VM bug.

Troubleshooting a hung process

If the application is responding very slowly or not responding at all, and if the process seems to be idle, try to trigger a thread dump. Also, as mentioned in “Diagnostic data collection” on page 400, examine the server’s SystemOut.log for messages from the Thread Monitor similar to those shown in Example 10-9.

Attention: All of the examples shown in this section were generated by running a J2EE application called BadApp.ear. In particular, this application was used to create a hung threads scenario when it was accessed from two different browsers simultaneously. The BadApp.ear file, including source code, is available for download (see Appendix C, “Additional material” on page 457 for more information).

Example 10-9 Thread Monitor messages

```
[...Timestamp...] 0000001b ThreadMonitor W   WSVR0605W: Thread "WebContainer : 1"
(0000002c) has been active for 623599 milliseconds and may be hung.
There is/are 1 thread(s) in total in the server that may be hung.

[...Timestamp...] 0000001b ThreadMonitor W   WSVR0605W: Thread "WebContainer :
0"(0000002b) has been active for 631957 milliseconds and may be hung.
There is/are 2 thread(s) in total in the server that may be hung.
```

The messages identify the threads that are potentially hung based on a configurable threshold, which is 600000 milliseconds by default.

To trigger the thread dump, use the methods described in “Troubleshooting a looping process” on page 430.

Finding a deadlock

If the hung process is responsive enough to generate a thread dump, then the output will be printed to the application server’s native_stdout.log file. HotSpot VM also executes a deadlock detection algorithm. If a deadlock is detected, it will be printed along with the stack trace of the threads involved in the deadlock. Example 10-10 on page 433 shows how the deadlock output will appear in the thread dump.

Example 10-10 Deadlock detected in thread dump data

```
Found one Java-level deadlock:
============================
"WebContainer : 1":
  waiting to lock monitor 0x00275188 (object 0xf4f1bf28, a java.lang.Object),
  which is held by "WebContainer : 0"
"WebContainer : 0":
  waiting to lock monitor 0x002755c0 (object 0xf4f1bf30, a java.lang.Object),
  which is held by "WebContainer : 1"

Java stack information for the threads listed above:
===================================================
"WebContainer : 1":
        at com.ibm.issf.atjolin.badapp.BadAppServlet.sneezyMethod(BadAppServlet.
java:338)
        - waiting to lock <0xf4f1bf28> (a java.lang.Object)
        - locked <0xf4f1bf30> (a java.lang.Object)
        at com.ibm.issf.atjolin.badapp.BadAppServlet.doPost(BadAppServlet.java:2
59)
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:763)
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:856)
        at com.ibm.ws.webcontainer.servlet.ServletWrapper.service(ServletWrapper
.java:989)
        at com.ibm.ws.webcontainer.servlet.ServletWrapper.handleRequest(ServletW
rapper.java:501)
        at com.ibm.ws.wswebcontainer.servlet.ServletWrapper.handleRequest(Servle
tWrapper.java:464)
        at com.ibm.ws.webcontainer.webapp.WebApp.handleRequest(WebApp.java:3276)
        at com.ibm.ws.webcontainer.webapp.WebGroup.handleRequest(WebGroup.java:2
67)
        at com.ibm.ws.webcontainer.WebContainer.handleRequest(WebContainer.java:
811)
        at com.ibm.ws.wswebcontainer.WebContainer.handleRequest(WebContainer.jav
a:1455)
        at com.ibm.ws.webcontainer.channel.WCChannelLink.ready(WCChannelLink.jav
a:113)
        at com.ibm.ws.http.channel.inbound.impl.HttpInboundLink.handleDiscrimina
tion(HttpInboundLink.java:454)
        at com.ibm.ws.http.channel.inbound.impl.HttpInboundLink.handleNewInforma
tion(HttpInboundLink.java:383)
        at com.ibm.ws.http.channel.inbound.impl.HttpInboundLink.ready(HttpInboun
dLink.java:263)
        at com.ibm.ws.tcp.channel.impl.NewConnectionInitialReadCallback.sendToDi
scriminators(NewConnectionInitialReadCallback.java:214)
        at com.ibm.ws.tcp.channel.impl.NewConnectionInitialReadCallback.complete
(NewConnectionInitialReadCallback.java:113)
        at com.ibm.ws.tcp.channel.impl.AioReadCompletionListener.futureCompleted
(AioReadCompletionListener.java:165)
        at com.ibm.io.async.AbstractAsyncFuture.invokeCallback(AbstractAsyncFutu
re.java:217)
        at com.ibm.io.async.AsyncChannelFuture.fireCompletionActions(AsyncChanne
lFuture.java:161)
        at com.ibm.io.async.AsyncFuture.completed(AsyncFuture.java:136)
        at com.ibm.io.async.ResultHandler.complete(ResultHandler.java:195)
        at com.ibm.io.async.ResultHandler.runEventProcessingLoop(ResultHandler.j
```

```
ava:743)
        at com.ibm.io.async.ResultHandler$2.run(ResultHandler.java:873)
        at com.ibm.ws.util.ThreadPool$Worker.run(ThreadPool.java:1469)
"WebContainer : 0":
        at com.ibm.issf.atjolin.badapp.BadAppServlet.dopeyMethod(BadAppServlet.j
ava:321)
        - waiting to lock <0xf4f1bf30> (a java.lang.Object)
        - locked <0xf4f1bf28> (a java.lang.Object)
        at com.ibm.issf.atjolin.badapp.BadAppServlet.doPost(BadAppServlet.java:2
57)
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:763)
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:856)
        at com.ibm.ws.webcontainer.servlet.ServletWrapper.service(ServletWrapper
.java:989)
        at com.ibm.ws.webcontainer.servlet.ServletWrapper.handleRequest(ServletW
rapper.java:501)
        at com.ibm.ws.wswebcontainer.servlet.ServletWrapper.handleRequest(Servle
tWrapper.java:464)
        at com.ibm.ws.webcontainer.extension.InvokerExtensionProcessor.handleReq
uest(InvokerExtensionProcessor.java:407)
        at com.ibm.ws.webcontainer.webapp.WebApp.handleRequest(WebApp.java:3276)
        at com.ibm.ws.webcontainer.webapp.WebGroup.handleRequest(WebGroup.java:2
67)
        at com.ibm.ws.webcontainer.WebContainer.handleRequest(WebContainer.java:
811)
        at com.ibm.ws.wswebcontainer.WebContainer.handleRequest(WebContainer.jav
a:1455)
        at com.ibm.ws.webcontainer.channel.WCChannelLink.ready(WCChannelLink.jav
a:113)
        at com.ibm.ws.http.channel.inbound.impl.HttpInboundLink.handleDiscrimina
tion(HttpInboundLink.java:454)
        at com.ibm.ws.http.channel.inbound.impl.HttpInboundLink.handleNewInforma
tion(HttpInboundLink.java:383)
        at com.ibm.ws.http.channel.inbound.impl.HttpInboundLink.ready(HttpInboun
dLink.java:263)
        at com.ibm.ws.tcp.channel.impl.NewConnectionInitialReadCallback.sendToDi
scriminators(NewConnectionInitialReadCallback.java:214)
        at com.ibm.ws.tcp.channel.impl.NewConnectionInitialReadCallback.complete
(NewConnectionInitialReadCallback.java:113)
        at com.ibm.ws.tcp.channel.impl.AioReadCompletionListener.futureCompleted
(AioReadCompletionListener.java:165)
        at com.ibm.io.async.AbstractAsyncFuture.invokeCallback(AbstractAsyncFutu
re.java:217)
        at com.ibm.io.async.AsyncChannelFuture.fireCompletionActions(AsyncChanne
lFuture.java:161)
        at com.ibm.io.async.AsyncFuture.completed(AsyncFuture.java:136)
        at com.ibm.io.async.ResultHandler.complete(ResultHandler.java:195)
        at com.ibm.io.async.ResultHandler.runEventProcessingLoop(ResultHandler.j
ava:743)
        at com.ibm.io.async.ResultHandler$2.run(ResultHandler.java:873)
        at com.ibm.ws.util.ThreadPool$Worker.run(ThreadPool.java:1469)
Found 1 deadlock.
```

In J2SE 5.0, deadlock detection works only with locks that are obtained using the *synchronized* keyword. Therefore, deadlocks that arise through the use of the java.util.concurrency package will not be detected.

If the deadlock detection algorithm detects a deadlock, then you must examine the output in more detail so that the deadlock can be understood. In Example 10-10 on page 433, you can see that thread `"WebContainer : 1"` is waiting to lock object `0xf4f1bf28,` which is held by `"WebContainer : 0",` and `"WebContainer : 0"` is waiting to lock monitor object `0xf4f1bf30,` which is held by `"WebContainer : 1".`

Additional details in the stack traces should provide helpful information about the deadlock.

Not finding a deadlock

If you are investigating a hang condition and the thread dump reveals that no deadlocks are found, then verify the following:

- The application or its libraries are *not* using the concurrency package (see “Finding a deadlock” on page 432 for restrictions on deadlock detection).
- Is there a code bug that causes a thread waiting on a monitor to never be notified?

If you suspect the second situation, it could be a timing issue or a general logic bug.

To investigate this issue further:

- Examine each of the threads in the thread dump.
- Examine each thread that is blocked in Object.wait().
- Check caller frame in the stack trace to see what class and method is invoking the wait() method.

The line number information provides direction as to what code you should examine. In most cases, you will need to understand the application logic or library is to pursue this issue any further. In general, you will need to know how the synchronization works in the application and in particular, the details and conditions under which monitors are notified.

No available thread dump

If the JVM does not respond to a **`Ctrl-\`**, then it is possible that the JVM may be deadlocked or hung for some other reason. In that case, use the **`jstack`** utility to obtain a thread dump. This also applies in the case where the application is not accessible or the output is directed to an unknown location.

With the **`jstack`** output, examine each of the threads in the BLOCKED state. The top frame can sometimes indicate why the thread is blocked (Object.wait or Thread.sleep, for example), and the rest of the stack should give an indication as to what the thread is doing. This is particularly true when the source has been compiled with line number information (the default) and you can cross reference the source code, as shown in Example 10-11.

Example 10-11 jstack output for hung threads

```
Thread t@135: (state = BLOCKED)
 - com.ibm.issf.atjolin.badapp.BadAppServlet.sneezyMethod() @bci=24, line=338
(Interpreted frame)

-com.ibm.issf.atjolin.badapp.BadAppServlet.doPost(javax.servlet.http.HttpServletRe
quest, javax.
servlet.http.HttpServletResponse) @bci=108, line=259 (Interpreted frame)
 - javax.servlet.http.HttpServlet.service(javax.servlet.http.HttpServletRequest,
javax.servlet.ht
```

```
tp.HttpServletResponse) @bci=139, line=763 (Interpreted frame)
 - javax.servlet.http.HttpServlet.service(javax.servlet.ServletRequest,
javax.servlet.ServletResp
onse) @bci=30, line=856 (Interpreted frame)
<...output truncated...>

Thread t@134: (state = BLOCKED)
 - com.ibm.issf.atjolin.badapp.BadAppServlet.dopeyMethod() @bci=24, line=321
(Interpreted frame)
-com.ibm.issf.atjolin.badapp.BadAppServlet.doPost(javax.servlet.http.HttpServletRe
quest, javax.
servlet.http.HttpServletResponse) @bci=101, line=257 (Interpreted frame)
 - javax.servlet.http.HttpServlet.service(javax.servlet.http.HttpServletRequest,
javax.servlet.ht
tp.HttpServletResponse) @bci=139, line=763 (Interpreted frame)
 - javax.servlet.http.HttpServlet.service(javax.servlet.ServletRequest,
javax.servlet.ServletResp
onse) @bci=30, line=856 (Interpreted frame)
-com.ibm.ws.webcontainer.servlet.ServletWrapper.service(javax.servlet.ServletReque
st, javax.ser
vlet.ServletResponse, com.ibm.ws.webcontainer.webapp.WebAppServletInvocationEvent)
@bci=247, line
=989 (Interpreted frame)
<...output truncated...>
```

If a thread is BLOCKED and the reason is not obvious, use the -m option to get a mixed stack. From the mixed stack output, it is usually possible to identify why the thread is blocked. For example, if a thread is blocked trying to enter a synchronized method or block, then you will see frames like ObjectMonitor::enter near the top of the stack, as shown in Example 10-12.

Example 10-12 jstack mixed stack output

```
----------------- t@134 -----------------
0xff2c6acc      ___lwp_cond_wait + 0x4
0xfea9e5e4      void ObjectMonitor::EnterI(Thread*) + 0x2b8
0xfeadd99c      void ObjectMonitor::enter2(Thread*) + 0x2c8
0xfe96eba8      void InterpreterRuntime::monitorenter(JavaThread*,BasicObjectLoc
k*) + 0x1e8
0xf8016d6c      * com.ibm.issf.atjolin.badapp.BadAppServlet.dopeyMethod() bci:24
 line:321 (Interpreted frame)
0xf8005764      * com.ibm.issf.atjolin.badapp.BadAppServlet.doPost(javax.servlet
.http.HttpServletRequest, javax.servlet.http.HttpServletResponse) bci:101 line:2
57 (Interpreted frame)

----------------- t@135 -----------------
0xff2c6acc      ___lwp_cond_wait + 0x4
0xfea9e5e4      void ObjectMonitor::EnterI(Thread*) + 0x2b8
0xfeadd99c      void ObjectMonitor::enter2(Thread*) + 0x2c8
0xfe96eba8      void InterpreterRuntime::monitorenter(JavaThread*,BasicObjectLoc
k*) + 0x1e8
0xf8016d6c      * com.ibm.issf.atjolin.badapp.BadAppServlet.sneezyMethod() bci:2
4 line:338 (Interpreted frame)
0xf8005764      * com.ibm.issf.atjolin.badapp.BadAppServlet.doPost(javax.servlet
.http.HttpServletRequest, javax.servlet.http.HttpServletResponse) bci:108 line:2
59 (Interpreted frame)
```

Other tools for troubleshooting hung processes

- Thread Analyzer

  The Thread Analyzer tool is described in 9.4.6, “Tools for performance tuning” on page 364. It is also useful for analyzing thread dump data. With the Thread Analyzer you can import multiple thread dumps taken at subsequent time intervals and perform a comparative analysis of the threads over time.

- Solaris **pstack** utility

  Another tool to mention in the context of hung processes is the **pstack** utility on Solaris. On Solaris 10 and JDK5.0 the output of **pstack** is similar to the output from **jstack -m**. As with **jstack**, the Solaris 10 implementation of **pstack** prints the fully qualified class name, method name, and bci. It will also print line numbers for the cases where the source was compiled with line number information (the default).

10.4.2 Crashes

When a crash, or fatal error, occurs, the JVM process will terminate. If an application server crashes on a WebSphere node, the node agent will attempt to restart the server. If the server can be restarted successfully, it might not be immediately apparent that the server crashed. However, if you notice performance degradation on a server and you examine the server’s SystemOut.log, you might notice that the server process is terminating and is being restarted periodically.

There many possible reasons for a crash. A crash can arise due to a bug in any of the following:

- The HotSpot VM
- A system library
- A J2SE library/API
- Application native code
- The operating system

External factors may also be involved. For example, a crash can arise due to resource exhaustion in the operating system.

Crashes caused by bugs in the HotSpot VM or J2SE library code should be rare. In the event that you do encounter a crash, then this section provides suggestions on how to examine the crash. In some cases, it may be possible to workaround a crash until the cause of the bug is diagnosed and fixed.

The first step with any crash is to locate the fatal error log. The fatal error log is a file named *hs_err_pid<pid>.log* (where <*pid*> is the process ID of the process). Normally the file is created in the working directory of the process. For WebSphere Application Server, the location is <was_home>/profiles/<*profile_name*>, for example, /opt/IBM/WebSphere/AppServer/profiles/AppSrv01.

However due to constraints on disk space, directory permissions, or other reasons, the file may instead be created in the temporary directory of the operating system. On Solaris, the temporary directory is /tmp.

In this section, we describe the format of the fatal error log, describe how the error log can be analyzed, and in a few cases, provide suggestions on how the issue may be worked around.

Format of the fatal error log

The fatal error log contains information retrieved at the time the fatal error occurred. If possible, the following information will be obtained from the process and written to the log file:

- The operating exception or signal that caused the fatal error
- Version and configuration information
- Details on the thread that caused the fatal error and its stack trace
- The list of running threads and their state
- Summary information about the heap
- The list of native libraries loaded
- Command-line arguments
- Environment variables
- Details about the operating system and CPU

Important: In some cases, only a sub-set of this information is written to the error log. This can happen when a fatal error is of such severity that the error handler is unable to recover and report all details.

The fatal error log in JDK 5.0 consists of the following sections:

- Header
- Thread
- Process
- System

Note: The fatal error log used in the following examples is from the crash of an application server Java process.

Header

The header gives a brief description about the problem that caused the crash. Example 10-13 shows a header from a fatal error log.

Example 10-13 Header section of fatal error log

```
#
# An unexpected error has been detected by HotSpot Virtual Machine:
#
#  SIGSEGV (0xb) at pc=0xff2c65c8, pid=19203, tid=1
#
# Java VM: Java HotSpot(TM) Server VM (1.5.0_11-b03 mixed mode)
# Problematic frame:
# C  [libc.so.1+0xc65c8]
```

The header in Example 10-13 tells you the following information:

- JVM terminated because of an unexpected error.
- The signal type is SIGSEGV.
- The program counter (pc) that issued the signal.
- The JVMs process ID (PID).

- The thread ID (TID)
- The type of JVM (Client or Server). It is Server in this example.
- Function frame that caused the crash: C [libc.so.1+0xc65c8].

The “C” in the function frame line signifies a native C frame. Table 10-3 shows other possible frame types.

Table 10-3 Function frame types

Frame	Description
C	Native C frame
J	Other frames, including compiled Java frames
j	Interpreted Java frames
V	VM frames
v	VM generated stub frame

Thread

Information about the thread that crashed is shown in this section. Example 10-14 shows the thread section from a fatal error log.

Example 10-14 Thread section of fatal error log

```
---------------  T H R E A D  ---------------

Current thread (0x00039418):  JavaThread "P=832795:0=0:CT" [_thread_blocked, id=1]

Registers:
 O0=0x0000005b O1=0x00000000 O2=0xffbfbf58 O3=0x00000000
 O4=0x009a41bf O5=0x00000000 O6=0xffbfbe98 O7=0xff2b965c
 G1=0x000000b7 G2=0xffbfc250 G3=0x00000002 G4=0xff02d02c
 G5=0x00008374 G6=0x00000000 G7=0xff3a2000 Y=0x083126e9
 PC=0xff2c65c8 nPC=0xff2c65cc


Top of Stack: (sp=0xffbfbe98)
0xffbfbe98:   ff3a2000 00000000 00000000 00000001
0xffbfbea8:   0020c49b 00000000 00000001 10624dd3
0xffbfbeb8:   00000000 00000000 ffbfbf58 00000000
0xffbfbec8:   00000000 7ffffd78 ffbfbef8 ff261b34
0xffbfbed8:   00000000 f4e18758 f4e17e80 ffbfc188
0xffbfbee8:   ffbfc33c 00000000 d8167dc0 ffbfc3cc
0xffbfbef8:   0020c49b 0fdf3b14 03f7cec5 04189360
0xffbfbf08:   0020c49b 00000000 083126e9 10624dd3

Instructions: (pc=0xff2c65c8)
0xff2c65b8:   81 c3 e0 08 01 00 00 00 82 10 20 b7 91 d0 20 08
0xff2c65c8:   0a bd 73 be 01 00 00 00 81 c3 e0 08 01 00 00 00

Stack: [0xffb7e000,0xffc00000),  sp=0xffbfbe98,  free space=503k
Native frames: (J=compiled Java code, j=interpreted, Vv=VM code, C=native code)
C  [libc.so.1+0xc65c8]
C  [libc.so.1+0x61b3c]  poll+0x84
```

```
V  [libjvm.so+0x19214c]
```

The Thread section in Example 10-14 on page 439 tells you the following information:

- Thread type: Java Thread
- State: _thread_blocked

The thread types are JavaThread, VMThread, CompilerThread, JVMPIDaemonthread, GCTaskThread, WatcherThread, and ConcurrentMarkSweepThread.

Important thread states are shown in Table 10-4.

Table 10-4 Important thread states

Thread state	Description
_thread_uninitialized	The thread is not created. This should never happen unless there has been memory corruption.
_thread_new	The thread has been created, but it has not yet been started.
_thread_in_native	The thread is running native code. There is probably a bug in native code.
_thread_in_vm	The thread is running VM code.
_thread_in_Java	The thread is running (either interpreted or compiled) Java code.
_thread_blocked	The thread is blocked.
..._trans	If you see any of the above states but they are followed by "_trans", it means the thread is changing to a different state.

For more details on threads, refer to *Java 2 Platform, Standard Edition 5.0 Troubleshooting and Diagnostic Guide*, found at:

http://java.sun.com/j2se/1.5/pdf/jdk50_ts_guide.pdf

Process

The process section contains information about the whole process, including the thread list and memory usage of the process.The thread list includes threads known to the VM. Included are all Java threads and some VM internal threads, but any native threads created by the user application that have not attached to the VM are not included. Example 10-15 shows an example of the process section.

Example 10-15 Process section

```
---------------  P R O C E S S  ---------------
Java Threads: ( => current thread )
  0x0143bee8 JavaThread "Non-deferrable Alarm : 3" daemon [_thread_blocked,
id=149]
  0x012a9bd0 JavaThread "Non-deferrable Alarm : 2" daemon [_thread_blocked,
id=145]
  0x004ca9d0 JavaThread "WebContainer : 2" daemon [_thread_in_native, id=136]
  0x00f8cfd8 JavaThread "WebContainer : 1" daemon [_thread_blocked, id=135]
  0x00939ed0 JavaThread "WebContainer : 0" daemon [_thread_blocked, id=134]
```

```
  0x0057b078 JavaThread "Deferrable Alarm : 3" daemon [_thread_blocked, id=131]
  0x010cd6e0 JavaThread "HAManager.thread.pool : 1" daemon [_thread_blocked,
id=130]
  0x010cd518 JavaThread "HAManager.thread.pool : 0" daemon [_thread_blocked,
id=129]
  0x02945af8 JavaThread "TCPChannel.DCS : 2" daemon [_thread_in_native, id=128]
<...this section truncated...>

Other Threads:
  0x00272728 VMThread [id=34]
  0x0053dac0 WatcherThread [id=44]
```

The Process section in Example 10-15 on page 440 tells you the following:

- Current thread (=>)
- Thread type: JavaThread, for example
- Names: "WebContainer : 2", "HAManager.thread.pool : 1", and so on
- State: _thread_blocked, _thread_in_native, and so on

For more details on process, refer to *Java 2 Platform, Standard Edition 5.0 Troubleshooting and Diagnostic Guide*, found at:

http://java.sun.com/j2se/1.5/pdf/jdk50_ts_guide.pdf

System

The final section in the fatal error log contains system information. The type of output you see depends on the operating system. In general, you will see the operating system version, CPU information, and a summary of the memory configuration. Example 10-16 is from Solaris 10.

Example 10-16 System section

```
                ---------------  S Y S T E M  ---------------

OS:                       Solaris 10 8/07 s10s_u4wos_12b SPARC
           Copyright 2007 Sun Microsystems, Inc.  All Rights Reserved.
                        Use is subject to license terms.
                           Assembled 16 August 2007

uname:SunOS 5.10 Generic_120011-14 sun4v  (T2 libthread)
rlimit: STACK 8192k, CORE infinity, NOFILE 65536, AS infinity
load average:0.32 0.37 0.29

CPU:total 32 has_v8, has_v9, has_vis1, is_sun4v, is_niagara1

Memory: 8k page, physical 16646144k(10405424k free)

vm_info: Java HotSpot(TM) Server VM (1.5.0_11-b03) for solaris-sparc, built on Dec
15 2006 01
:18:11 by unknown with unknown Workshop:0x550
```

Types of crashes identifiable in the fatal error log

The following crashes usually can be identified as such by examining the fatal error log.

- Crash in native code

  The header will show that the SIGSEGV comes from a thread running in a native library. Or a JavaThread may crash while in the "_thread_in_vm" state.

- Stack overflow crash

  While a stack overflow in Java code will usually become manifest as a java.lang.StackOverflowError being thrown by the thread, a stack overflow in JNI™ code is fatal and will cause the process to terminate. Look for the following in the fatal error log:

  - There is an EXCEPTION_STACK_OVERFLOW in the header.
  - The thread state is _thread_in_native, indicating that the thread is executing JNI code.
  - Native frames indicates that a recursive function is executing.

- Crash in a HotSpot compiler thread

  If the "Current thread" is a JavaThread named either CompilerThread0, CompilerThread1, or AdapterCompiler, then the crash is usually due to a bug in the compiler.

- Crash in compiled code

  A crash in compiled code typically means that you have encountered a compiler bug that has resulted in incorrect code being generated. Look for the following in the fatal error log:

  - The problematic frame is marked "J" (compiled Java frame).
  - An incomplete stack trace with the message `error occurred during error reporting...` exists.

- Crash in the VM thread

  The VM thread is a special thread in the HotSpot VM. It performs VM management tasks, such as garbage collection. If the VM_Operation suggests that the operation is a garbage collection, then it is possible that you have encountered a problem, such as heap corruption. Look for the following in the fatal error log:

  - `Current Thread` is the VMThread.
  - `VM_Operation` in the Thread section.

Implementing workarounds for crashes

If a critical application is crashing, and the crash appears to be caused by a bug in the HotSpot VM, then you probably need to find and implement a temporary workaround as soon as possible. If the crash arises with an application that is deployed with the most up-to-date release of 5.0, then the crash should always be reported to Sun Microsystems. For complete details on finding and implementing workarounds for the crashes described above, refer to the *Java 2 Platform, Standard Edition 5.0 Troubleshooting and Diagnostic Guide*, found at:

`http://java.sun.com/j2se/1.5/pdf/jdk50_ts_guide.pdf`

10.4.3 Troubleshooting memory leaks

Application developers must often deal with applications that terminate with an java.lang.OutOfMemoryError exception. This exception is thrown when there is not enough space to allocate an object on the heap. When an allocation failure occurs, a garbage collection will be performed in order to make room for the object. The OutOfMemoryError exception is not thrown unless the following are true:

- The garbage collection cannot make any additional space available to allocate the new object.
- The heap cannot be further expanded.

An OutOfMemoryError does not necessarily mean the application is leaking memory. The problem might be that the specified heap size is not sufficient for the application.

Another memory related problem can occur when applications terminate because there is no available virtual memory on the operating system. In this section, we will discuss OutOfMemoryError conditions and how to troubleshoot memory leaks in applications.

Diagnosing an OutOfMemoryError

The first step in diagnosing an OutOfMemoryError error is to determine what the error means. Is the Java heap full or is the native heap full? Here are some error messages that you might see when the OutOfMemoryError is thrown. Example 10-17 shows a different OutOfMemoryError error that may be found in the native_stderr.log file.

Example 10-17 OutOfMemoryError message

```
[Time stamp] [Thread ID] SystemErr R java.lang.OutOfMemoryError: Java heap space

[Time stamp] [Thread ID] SystemErr R java.lang.OutOfMemoryError: PermGen space

[Time stamp] [Thread ID] SystemErr R java.lang.OutOfMemoryError: Requested array
size exceeds VM limit
```

- Java heap space

  If you see *Java heap space*, this means that an object could not be allocated.

  - This does not necessarily mean that an application is leaking memory. It may be that the configured or default maximum heap size is too small.
  - However, this may well be a symptom that an application is leaking memory. Memory leaks are described as conditions where an application has objects that are no longer needed but is unintentionally holding references to them so that they cannot be garbage collected. As the application continues to create more of these objects, the heap eventually runs out of memory. The HAT tool (see “Heap Analysis Tool (HAT)” on page 413) can be used to view all reachable objects and understand which references are keeping each one alive.
  - Another cause of this error might be the excessive use of finalizers by the application.

- PermGen space

  - This means that the *permanent generation* is full. As described earlier (see “Java SE 5.0 Sun HotSpot JVM garbage collectors” on page 353), this is the area of the heap where the JVM stores its metadata. If an application loads too many classes, then you may need to increase the size of the permanent generation. Specify the command-line option –XX:MaxPermSize=*n*, where *n* specifies the size.

- Requested array size exceeds VM limit
  - This means that the application tried to allocate an array that is larger than the heap size. If, for example, an application tries to allocate an array of 512 MB, but the maximum heap size is 256 MB, then this error will be thrown. Usually if you see this message, it is probably because the heap size is too small or there may be a bug in the code that tries to create an array that is unnecessarily large.

Troubleshooting memory leaks in Java code

If you are seeing OutOfMemoryError exceptions in the log files and you suspect the application code has a memory leak, then begin your investigation by taking one or more heap dumps. Refer to 10.2.3, "Heap Analysis Tool (HAT)" on page 413 for a discussion of how to use the -XX:+HeapDumpOnOutOfMemoryError command-line option and how to use HAT to process the heap dump.

If you have a heap dump that was generated at the time that the OutOfMemoryError happened, you can use HAT to process the dump and create a snapshot of the heap, which you can then examine in any browser.

To make effective use of the information that HAT provides, you need to have some knowledge of the application and, additionally, some knowledge about the libraries/APIs that the application uses. Generally the HAT information can help answer two important questions:

- What is keeping an object alive?
- Where was this object allocated?

Analyze the HAT information to determine why the object is being kept alive

For any suspected object instances, check the objects listed in the section named *"References to this object"* to see which objects directly reference these objects.

You can also use a *"roots query*" to provide you with the reference chains from the root set to the given object. Reference chains show a path from a root object to this object. These chains can quickly show how an object is reachable from the root set.

Two roots queries are available: One *excludes* weak references (roots), and the other *includes* them (allRoots). A weak reference is a reference object that does not prevent its referent from being made finalizable, finalized, and then reclaimed. If an object is only referred to by a weak reference, it usually is not considered to be retained, because the garbage collector can collect it as soon as it needs the space.

HAT sorts the rootset reference chains by the type of the root, in this order:

- Static data members of Java classes.
- Java local variables. For these roots, their thread is also shown. Since a thread is a Java object, the link is active (clickable). Therefore, you can navigate to the name of the thread.
- Native static values.
- Native local variables. These roots are also identified with their thread.

Analyze the HAT information to determine where the object was allocated

For object instances, the section entitled "Objects allocated from" shows the allocation site as a stack trace so you can determine where the object was created.

If the leak cannot be identified using a single object dump, then another approach is to collect a series of dumps and to focus on the objects created in the interval between each dump. HAT provides this capability using the -baseline option, where you can provide two dumps for HAT to process.

If you start HAT with two heap dumps, the “Show instance counts for all classes” query includes another column that shows the count of new objects for that type. An instance is considered new if it is in the second heap dump, and there is no object of the same type with the same ID in the baseline. If you click a new count, then HAT lists the new objects of that type. For each instance, you can view where it was allocated, which objects these new objects reference, and which other objects reference the new object.

The baseline option can be very useful if the objects that need to be analyzed are created during the time between the successive dumps.

Other tools for troubleshooting OutOfMemoryError issues

- The **`jmap -histo`** command

  As mentioned in “Using the jmap command” on page 368, a heap histogram can be obtained by running the **`jmap -histo <pid>`** command against the process. The output generated will show the total size and instance count for all class types in the heap. If several histograms is generated (two or three minutes apart), then it may be possible to see a trend that may be helpful in analyzing the problem.

- Memory Dump Diagnostic for Java (MDD4J)

  MDD4J can be downloaded and run in the IBM Support Assistant. This tool can analyze heap dumps in different formats, including text and phd. WebSphere Application Server on Solaris 10 can be configured to generate heap dumps in text format. Figure 10-16 shows the Analysis Summary for a heap dump that was generated after an OutOfMemoryError exception was thrown.

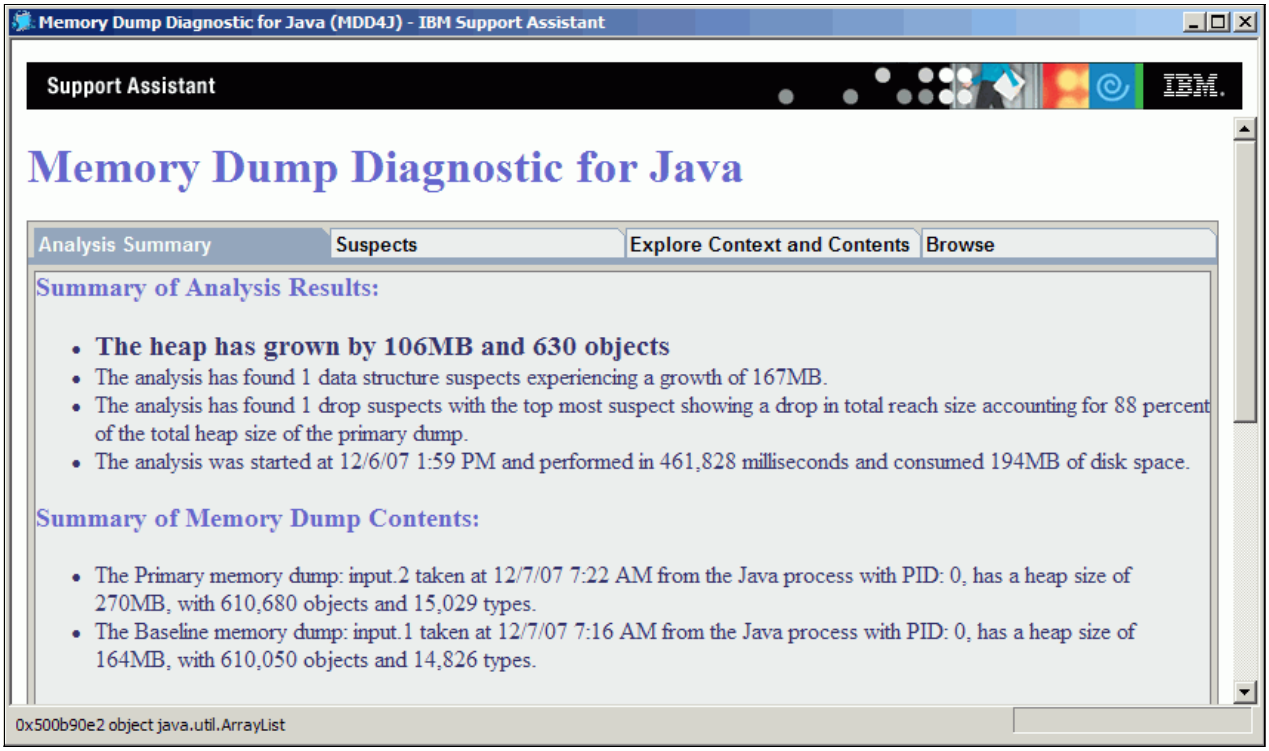


Figure 10-16 MDD4J heap dump analysis summary

Troubleshooting leaks in native code

There are several techniques used to find and isolate native code memory leaks. For a detailed discussion of these techniques, refer to the *Java 2 Platform, Standard Edition 5.0 Troubleshooting and Diagnostic Guide*, found at:

`http://java.sun.com/j2se/1.5/pdf/jdk50_ts_guide.pdf`

A

WebSphere Application Server V6.0.2 considerations

This appendix describes the key differences between WebSphere Application Server versions 6.0.2 and 6.1. It also highlights some of the changes in V6.0.2 that WebSphere Administrators need to know when working with this version compared to older versions of WebSphere Application Server, such as v5.x.

Although we suggest customers upgrade to the latest versions of WebSphere Application Server (such as V6.1) for good reasons like performance improvements, there are many reasons they may not be able to do so immediately. Therefore, in this appendix, we address some of the key issues with WebSphere Application Server V6.0.2 and Solaris 10 so that customers like you have a better experience with WebSphere Application Server on Solaris 10.

- What is new from previous release
- Java performance tuning guidelines
- Differences with silent product installation between versions 6.0.2 and 6.1
- Security considerations

What is new in WebSphere Application Server V6.0.2

WebSphere Application Server V6.0.2 offers many new and exciting improvements to WebSphere products. Whether you are new to the product, or making the transition from a prior release, there will be lot of new features to help you make the transition.

WebSphere Application Server is a proven, high-performance transaction engine that can help you build, run, integrate, and manage dynamic business applications. It excels as the foundation for a service-oriented architecture (SOA) with these main benefits:

- Simple, rapid development and deployment
- Secure and scalable SOA runtime
- Extensible communication services
- Effective application management
- Improved platform consistency

More details about what is new in this version can be found at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/topic/com.ibm.websphere.base.doc/info/aes/ae/welc_newinrelease.html

All of these can be combined with Solaris 10 to take the advantage of Solaris 10 capability. This version supports J2EE 1.4 specification and works with JDK 1.4.2, which poses a different set of challenges for WebSphere Application Server admins.

One of the important changes to note here is that we do not have to deal with the MQ as a separate component, which has been integrated into WebSphere Application Server itself, so we do not need to set any /etc/system parameters, which we used to with the previous version. This adds much simplicity.

Also, note that this is the first release in which all platforms have been made available at the same time with the same feature-functionality. Common code and delivery enhance the mobility of applications. You can now deploy your applications on the platforms best suited to your needs without delay or trade-off. Starting with this release, WebSphere Application Server became available on Sun Solaris 10 on X86 based system. However, on those systems, it was only available as 64-bit, which gives you additional capability to exploit larger heap space and a choice of a Solaris 10 hardware platform.

Java performance tuning guidelines

The JDK version supported with WebSphere Application Server V6.0.2 is 1.4.2, which has a different characteristics than JDK 1.5. The newer JDK includes the Java Virtual Machine that is better optimized and delivers better performance, as discussed in Chapter 9, “WebSphere Application Server performance tuning on Solaris 10” on page 313. The Sun JDK bundled with WebSphere Application Server V6.0.2 slightly differs from the Sun JDK (which you can download from the official Sun Web site at: http://java.sun.com), namely Object Request Broker (ORB), Security, and the XML Module, which has been replaced by IBM’s own implementation. However, the base JVM itself remains the same. Therefore, you can apply any standard tuning and optimization for Sun JDK 1.4.2 to WebSphere Application Server V6.0.2.

Note: Some of the features we describe in 9.4, “Java Virtual Machine (JVM) Performance Management” on page 346 are specific to JDK 1.5; therefore, the following recommendations apply to WebSphere Application Server V6.1 only:

- JVM Ergonomics (Self tuning)
- JVM Arguments, such as:
  - -XX:+ParallelOldGC
  - -XX:+AggressiveOpts

You can use the following JVM tuning guidelines for WebSphere Application Server V6.0.2. If you want to put in the ergonomics feature’s tuning for 1.5, you need to explicitly apply the proper arguments for the JVM.

Some of the JVM options, which a WebSphere Application Server admin can tweak to get better performance, are:

- -server (needs to be specified explicitly due to an absence of JVM Ergonomics)
- -Xmn<*size-of-heap-for-young-generation*>m
- -XX:-ScavengeBeforeFullGC
- -XX:MaxTenuringThreshold
- -XX:+UseParallelGC
- -XX:ParallelGCThreads (number of CPU or processing cores on multi core systems)
- -XX:+AggressiveHeap
- -XX:PermSize

The details of Garbage Collection for JDK 1.4.2 is available at:

`http://java.sun.com/docs/hotspot/gc1.4.2`

Attention: The WebSphere Application Server performance and scalability of a single instance between V6.0.2 and V6.1 are not directly comparable due to major enhancements in JDK 1.5 over 1.4.2. What WebSphere Application Server V6.1 can perform and produce with a single instance on a system, you may have to use two instances of WebSphere Application Server V6.0.2 to achieve the similar performance and better utilization.

Silent installation considerations

The silent installation procedure has changed between versions. Version 6.1 silent installation can provide a profile creation of one deployment manager and one federated cell in the same binary installation. Also, profile creation is configured in the same response file where product binary installation options are configured.

Now you do not have to configure two different response files for installation and profile creation. Refer to Table A-1 for more information about the options that replace V6.0.x profile creation response files and options in those response files.

Table A-1 Silent installation options summary

V6.0 option	V6.1 option
-W ndsummarypanelInstallWizardBean.launchPCT (responsefile.nd.txt)	-OPT createProfile

V6.0 option	V6.1 option
-W pctresponsefilelocationqueryactionInstallWizardBean.fileLocation (responsefile.nd.txt)	N/A
-W profilenamepanelInstallWizardBean.profileName (responsefile.pct.NDdmgrProfile.txt)	-OPT PROF_dmgrProfileName
-W profilenamepanelInstallWizardBean.isDefault (responsefile.pct.NDdmgrProfile.txt)	-OPT PROF_isDefault
-P installLocation (responsefile.pct.NDdmgrProfile.txt)	-OPT PROF_profilePath
-W nodehostandcellnamepanelInstallWizardBean.nodeName (responsefile.pct.NDdmgrProfile.txt)	-OPT PROF_nodeName
-W nodehostandcellnamepanelInstallWizardBean.hostName (responsefile.pct.NDdmgrProfile.txt)	-OPT PROF_hostName
-W nodehostandcellnamepanelInstallWizardBean.cellName (responsefile.pct.NDdmgrProfile.txt)	-OPT PROF_cellName
-W pctdmgrprofileportspanelInstallWizardBean.WC_adminhost -W pctdmgrprofileportspanelInstallWizardBean.WC_adminhost_secure -W pctdmgrprofileportspanelInstallWizardBean.BOOTSTRAP_ADDRESS -W pctdmgrprofileportspanelInstallWizardBean.SOAP_CONNECTOR_ADDRESS -W pctdmgrprofileportspanelInstallWizardBean.SAS_SSL_SERVERAUTH_LISTENER_ADDRESS -W pctdmgrprofileportspanelInstallWizardBean.CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS -W pctdmgrprofileportspanelInstallWizardBean.CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS -W pctdmgrprofileportspanelInstallWizardBean.ORB_LISTENER_ADDRESS -W pctdmgrprofileportspanelInstallWizardBean.CELL_DISCOVERY_ADDRESS -W pctdmgrprofileportspanelInstallWizardBean.DCS_UNICAST_ADDRESS (responsefile.pct.NDdmgrProfile.txt)	-OPT PROF_defaultPorts -OPT PROF_startingPort (-OPT PROF_nodeStartingPort) This option is used if you are creating a deployment manager profile and one federated cell in the same machine.

V6.0 option	V6.1 option
-W profiletypepanelInstallWizardBean.selection (responsefile.pct.NDdmgrProfile.txt)	-OPT profileType
-W setnondmgrcellnameinglobalconstantsInstallWizardBean.value (responsefile.pct.NDstandAloneProfile.txt)	-OPT PROF_cellName
-W pctfederationpanelInstallWizardBean.federateLater (responsefile.pct.NDmanagedProfile.txt)	-OPT PROF_federateLater
-W pctfederationpanelInstallWizardBean.hostname (responsefile.pct.NDmanagedProfile.txt)	-OPT PROF_dmgrHost
-W pctfederationpanelInstallWizardBean.port (responsefile.pct.NDmanagedProfile.txt)	-OPT PROF_dmgrPort
-W profiletypepanelInstallWizardBean.selection (responsefile.pct.NDmanagedProfile.txt)	N/A

Refer to the InfoCenter for the correct and appropriate values for these options at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp

Security considerations

This section outlines some of the major changes from WebSphere Application Server V6.0 to WebSphere Application Server V6.1.

The following security-related items are new for WebSphere Application Server V6.1:

- Administrative security can be enabled out of the box. Access to the administrative system and its data is now protected by default.
- Simplified security configuration and administration. The administrative console security panels are simplified, and new wizards and a configuration reporting tool are provided.
- Automatically generated server IDs. You no longer need to specify a server user ID and password during security configuration, unless using a mixed cell environment.
- Federate various repositories, so you can manage them as one. The inclusion of virtual member manager in this release provides a single model for managing organizational entities. You can configure a realm that consists of identities in the file-based repository that is built into the system in one or more external repositories or in both the built-in, file-based repository and in one or more external repositories.
- WebSphere key and certificate management has been simplified.
- Interoperability with other vendors of WS-Security. The product now supports the WS-I Basic Security Profile 1.0, which promotes interoperability by addressing the most common problems encountered from implementation experience to date.
- Separate Web authentication and authorization. Now, Web authentication can be performed with or without Web authorization. A Web client's authenticated identity is available whether or not Web authorization is required.
- Enhanced control over Web authentication behavior.

- Portlet URL security, enabling direct access to portlet URLs just like servlets.
- Larger variety of administrative roles.
- Fine-grained administrative role authorization. In prior releases, users granted administrative roles could administer all of the resource instances under the cell. Now, the product is more fine-grained, meaning that access can be granted to each user per resource instance.
- Hardware cryptographic device support for Web services security.

B

Sun Solaris 8 Migration Assistant

For customers with earlier Solaris 8 environments, the Solaris 8 Migration Assistant helps you quickly and easily consolidate one or many of your existing Solaris 8 systems onto a newer, more cost effective, and more powerful Sun SPARC systems, such as the CoolThreads™ and M-series servers that are running Solaris 10. In other words, you host your Solaris 8 environment inside a Solaris 10 container without requiring application migration. Customers can then transition these environments to "native" Solaris 10 containers at their own pace while removing the dependency on Solaris 8 hardware support.

In addition, it also gives the customer the capability of running different containers with different Solaris 8 updates or different sets of patches. This may be essential, as you may have one vendor’s application that is only supported on one specific Solaris 8 and its patch level while another vendor’s application requires Solaris 8 with a different kernel patch level.

The Sun Solaris 8 Migration Assistant provides easier transition to Solaris 10.

Solaris 8 Migration Assistant

The Solaris 8 Migration Assistant is a combination of software and services offering that helps customers accelerate the migration of application environments using older WebSphere Application Server, such as V5.x, and Sun Solaris 8 to new systems running Solaris 10.

This is a part of the virtualization technology offering from Sun to aid customers who have applications that have not migrated to Solaris 10 and customers who are willing to migrate to Solaris 10 in near future. Previously, the Solaris 10 containers only supported applications that were already running on Solaris 10. As a result, the containers could not be used for applications that still needed a Solaris 8 environment. To assist the customer with these types of situations, Sun developed a technology called BrandZ. BrandZ architecture was released by Sun in October 2007. Complete details on BrandZ can be found at:

http://www.opensolaris.org/os/community/brandz/

BrandZ is a framework that extends the Solaris Containers infrastructure to create Branded Containers, which are containers that contain non-native operating environments like Solaris 8 in a Solaris 10 environment. Solaris 8 is the non-native operating environment and Solaris 10 is the native operating environment. The Solaris 8 Migration assistant utilizes the BrandZ technology in Solaris to allow Solaris 8 applications to run in a Solaris 10 container. While many Solaris 8 applications have been already migrated to Solaris 10, there are still many applications running old Solaris 8 applications and in the process of migrating to Solaris 10. Some of these application may be nearing their end of life (EOL) very soon due to vendor support or perhaps a customer wants to move to a newer version of software or hardware that provides new features, cost effectiveness, or better scalability and reliability.

Note: Solaris 8 end-of-support milestones are still in effect: Vintage I support ends March 31, 2009; end of service life is March 31, 2012. See http://sun.com/solaris/8 for details.

Overview of Solaris 8 migration assistant

The main components of the Migration Assistant are:

- Software Component
- Service Component

The Migration Assistant software can capture the entire environment of the Solaris 8 source system that will be migrated and then transfer this environment to a Solaris Container running on the target Solaris 10 system. However, we should note that this is provided for migration purposes and should not be planned to be used for the long term. Sun also offers additional services to customers going through the migration cycle:

http://sun.com/service/serviceplans/software/overview.xml

The services part of the Migration Assistant helps customers from the initial feasibility study to the actual implementation, and ultimately, support for the technology.

Solaris 8 Migration Assistant can be downloaded from:

http://sun.com/download/products.xml?id=470c4a45

And the documentation can be found here:

http://docs.sun.com/app/docs/coll/1759.1?l=es

How it works

The Solaris 8 Migration Assistant consists of two software components:

- Physical to virtual (P2V) conversion tool
- Solaris 8 Container.

The P2V tool helps to move the application and its environment from the Solaris 8 system to the Solaris 10 system and converts it so it can run on this new platform.

The second component, the Solaris 8 Container, is the runtime environment for the migrated Solaris 8 environment.

When the existing Solaris 8 environment is transferred to the destination system and placed in a Solaris 8 Container, the installer adds a few patches to the Solaris 8 image to make it work correctly on the new system.

This installation does not install a new version of Solaris 8. It creates a Solaris Container that will hold the Solaris 8 bits that were copied from the source system and puts those bits into the Container ready to run on the new system.

The Solaris 8 Container is a new type of Solaris Container that is based on the previously mentioned BrandZ architecture.

Usages with WebSphere Application Server V5.1

In the Redbooks test environment, we tested and found no issues running WebSphere Application Server V5.1 in this container.

The operating system requirements are the same as a Solaris 8 operating environment. For example, System V related IPC parameters can be specified the same way as it used to be for Solaris 8. In a nutshell, once the Solaris 8 container is created and deployed, WebSphere Application Server V5.1 can be deployed as though it is on a stand-alone Solaris 8 system.

You can keep this Solaris 8 Container to host the WebSphere Application Server V5.1 application environment until the application environment is transitioned to a newer environment, such as WebSphere Application Server V6.1. When the system transition is completed, you can take the Solaris 8 container out of your production environment.

Similarly, the Solaris 8 Migration Assistant can help deploy other WebSphere products. We strongly recommend that you perform tests and analysis of your application environments for each product to ensure that there are no technical issues.

C

Additional material

This book refers to additional material that can be downloaded from the Internet as described below.

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 at:

ftp://www.redbooks.ibm.com/redbooks/SG247584

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG247584.

Using the Web material

The additional Web material that accompanies this IBM Redbooks publication includes the following files:

File name	*Description*
FileRegistrySample.zip	Zipped sample java source code and properties files.
BadApp.zip	Contains the badappWithSource.ear file and instructions for installing and using it.

How to use the File Registry Sample material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder.

Use the following procedure to compile and execute the sample Java code:

1. Implement all the methods in the interface except for the CreateCredential method, which is implemented by WebSphere Application Server. The FileRegistrySample.java file is provided for reference.

   Attention: The sample provided is intended to familiarize you with this feature. Do not use this sample in an actual production environment.

2. Build your implementation.

To compile your code, you need the app_server_install_rootBase/plugins/com.ibm.ws.runtime_6.1.0.jar and the app_server_install_rootBase/plugins/com.ibm.ws.security.crypto_6.1.0.jar files in your class path. Refer to Example C-1 for more information.

Example: C-1 Sample command to compile the java file

```
%install_root%/java/bin/javac -classpath
%install_root%app_server_install_rootBase/plugins/com.ibm.ws.runtime_6.1.0.jar;
%install_root%app_server_install_rootBase/plugins/com.ibm.ws.security.crypto_6.1.0
/FileRegistrySample.java
```

3. Copy the class files that are generated in the previous step to the product class path.

   The preferred location is the %was_install_root%/lib/ext directory. Copy these class files to all of the product process class paths.

4. Follow the steps in 8.2.1, “Custom registry interface” on page 274 to configure your implementation using the administrative console. This step is required to implement custom user registries.

What to do next

If you enable security, make sure that you complete the remaining steps:

1. Save and synchronize the configuration and restart all of the servers.
2. Try accessing some J2EE resources to verify that the custom registry implementation is correct.

How to use the BadApp problem determination material

BadApp has no EJBs; it consists of one HTML page (index.html), one servlet BadAppServlet, and one JSP BadAppJSP.

To install BadApp (badappWithSource.ear), follow the normal WebSphere Application Server procedures and use all the defaults during installation. Once it is installed and running, the URL for the BadApp main page is `http://localhost:9083/BadAppWebProject/index.html` (The host name and the port may vary depending on your installation and how many versions of WebSphere Application Server you have installed at the time.)

The source is included in the ear file so that you can examine it to see how the hang condition and out of memory problems are built into the code.

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

For information about ordering these publications, see “How to get Redbooks” on page 462. Note that some of the documents referenced here may be available in softcopy only.

- *Optimizing Operations with WebSphere Extended Deployment V6.1*, SG24-7422
- *WebSphere Application Server Network Deployment V6: High Availability Solutions*, SG24-6688
- *WebSphere Application Server V6 Problem Determination for Distributed Platforms*, SG24-6798
- *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392
- *WebSphere Application Server V6 System Management & Configuration Handbook*, SG24-6451
- *WebSphere Application Server V6.1: System Management and Configuration*, SG24-7304

Sun Blueprints

These Sun publications are also relevant as further information sources:

- *Solaris Containers--What They Are and How to Use Them*, found at:

  http://sun.com/blueprints/0505/819-2679.pdf
- *Solaris Containers Technology Architecture Guide*, found at:

  http://sun.com/blueprints/0506/819-6186.pdf
- *Service Management Facility (SMF) in the Solaris 10 OS*, found at:

  http://sun.com/blueprints/0206/819-5150.pdf
- *Beginners Guide to LDoms: Understanding and Deploying Logical Domains for Logical Domains 1.0 Release*, found at:

  http://sun.com/blueprints/0207/820-0832.html
- *Limiting Service Privileges in the Solaris 10 Operating System*, found at:

  http://sun.com/blueprints/0505/819-2680.pdf
- *Developing and Tuning Applications on UltraSPARC T1 Chip Multithreading Systems*, found at:

  http://sun.com/blueprints/0107/819-5144.html

Online resources

These Web sites are also relevant as further information sources:

- Albert Leigh's blog:

  http://blogs.sun.com/sunabl/
- Dileep Kumar's blog:

  http://blogs.sun.com/dkumar/
- Sun Solaris How - To Guides:

  http://www.sun.com/software/solaris/howto_guides.jsp
- WebSphere Application Server Information Center

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

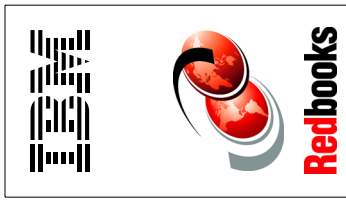
U

V

W

Y

Z



IBM WebSphere Application Server V6.1 on the Solaris 10 Operating System

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

IBM WebSphere Application Server V6.1 on the Solaris 10 Operating System

Learn detailed deployment and configuration strategies

Leverage Solaris 10 virtualization and other new features

Optimize WebSphere Application Server performance on Solaris 10

This IBM Redbooks publication tells the reader how to run WebSphere Application Server V6.1 on the Sun Solaris 10 Operating System. While WebSphere Application Server is platform-independent, each platform it runs on requires some additional platform-specific knowledge and configuration in order to ensure it operates efficiently and at maximum capability.

Our primary focus is to explain the installation, configuration, and best practices for deployment and performance tuning of WebSphere Application Server in the Solaris environment. We take you through all the steps required to run WebSphere Application Server on Solaris 10, including installing WebSphere Application Server and Solaris, tuning the operating system and application server, security, administrative tasks, problem determination, and advanced topologies.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-7584-00 ISBN 0738485993